نظریه زبانها و اتوماتا

پیتر لینز

جعفر تنها – محمد سلیمانی تبار

سرشناسه	: لینز، پیتر	Linz, Peter
عنوان و پدیدآور	: [illegible] / پیتر لینز ترجمه جعفر تنها، محمد سلیمانی‌تبار	
مشخصات نشر	: تهران: نص، ۱۳۸۸.	
مشخصات ظاهری	: ۳۹۲ ص	
شابک	: ۶۰۰۰۰ ریال	ISBN: 978-964-410-212-7
وضعیت فهرست نویسی	: فیپا.	
یادداشت	: عنوان اصلی	An introduction to formal languages and autumata , 4th. Ed,c 2006.
عنوان دیگر	: مقدمه‌ای بر نظری زبانها و ماشینها	
عنوان دیگر	: نظریه زبانها و ماشینها	
موضوع	: زبان‌های صوری	
موضوع	: نظریه ماشین	
شناسه افزوده	: تنها، جعفر، ۱۳۵۵، مترجم	
شناسه افزوده	: سلیمانی تبار، محمد، ۱۳۵۱، مترجم	
رده‌بندی کنگره	: الف ۱۳۸۸ ۷م۹ ل QA۲۶۷/۳	
رده‌بندی دیویی	: ۰۰۵/۱۳۱	
شماره کتابشناسی ملی	: ۱۷۸۵۸۶۵	

موسسه علمی فرهنگی

نظریه زبان‌ها و اتوماتا

پیتر لینز

جعفر تنها، محمد سلیمانی تبار

چاپ اول : تابستان ۸۸

تیراژ: ۲۰۰۰

ناشر: «نص»

چاپ و صحافی : سازمان چاپ و انتشارات وزارت فرهنگ و ارشاد اسلامی

طراحی، آماده‌سازی: موسسه علمی فرهنگی «نص»

قیمت: ۶۰۰۰ تومان

دفتر: تهران، میدان انقلاب، خ اردیبهشت، بن بست مین، شماره ۶ ضلع جنوب شرقی میدان انقلاب، شماره ۳/۱۵۰۰

تلفن: ۶۶۴۱۲۳۸۵-۶۶۴۶۵۶۷۴-۶۶۹۵۳۸۸۳ فاکس: ۶۶۹۵۷۶۹۰ ص.پ. ۸۶۳-۱۳۱۲۵ تلفن: ۶۶۴۰۵۳۷۲

وب سایت: www.nass.ir - ایمیل: info@nasspub.com

شابک: ۷-۲۱۲-۴۱۰-۹۶۴-۹۷۸ ISBN: 978-964-410-212-7

تقدیم

وجود عالم محسوس و تمامی حرکات و تغییرات آن نشأت گرفته از سنت‌هایی است که در عالمی ورای این عالم، به دست **الله**، آن یگانه هستی بخش آفریده شده‌اند و شناخت این سنت‌های تغییرحسی، لازمه همیشگی قدرت یافتن این بشر خاکی بر سایر پدیده‌های عالم هستی می‌باشد.
این اثر قطره‌ای از تلاش آدمی برای ورود به بیتی ازآن منظومهٔ پر رمز و راز نهفته در بطن عالم است.

به امید درک رازهای پنهانی قصیدهٔ آسمانی عشق حق.

این اثر را به پیشگاه پاک **پدر، مادر و همسرم** تقدیم می‌کنم.

نعمت‌هایی که مرا خوب درک می‌کنند و من از درک آنها عاجز و ناتوانم،
وجود آنها برای من همیشه نعمت و افتخار ولی وجود من برای آنها همیشه رنج و مشقت،
از دستان دنیا دیده آنها سجدگاهی می‌سازم و بر آن نماز شکر می‌خوانم.

پیش گفتار مؤلف

کتاب حاضر را می‌توان یک منبع آموزشی هرچند مقدماتی برای زبان‌های صوری، اتوماتا، محاسبه‌پذیری و دیگر موضوعات مربوطه درنظرگرفت. این سرفصل‌ها بخش عمده‌ای از نظریه‌محاسبات را تشکیل می‌دهند که امروزه بعنوان یک استاندارد آموزشی در برنامه درسی اغلب شاخه‌های کامپیوتر گنجانده می‌شود. برای درک مطالب کتاب، باید با برخی از زبان‌های برنامه‌نویسی سطح بالا خصوصاً C ، C++ و Java و همچنین اصول بنیادی ساختمان‌داده‌ها و الگوریتم‌ها آشنایی داشته. بعلاوه، ضروری است قبل از مطالعه این کتاب، دانشجویان درس ریاضیات‌گسسته، شامل مباحث نظریه مجموعه‌ها، توابع، روابط، منطق و اصول استدلال‌ریاضی را با موفقیت گذرانده باشند. هرچند امروزه این سرفصل‌های پیشنیاز در برنامه آموزشی اغلب رشته‌های کامپیوتری تدریس می‌شود.

از جمله مهمترین اهداف مطالعه نظریه محاسبات می‌توان به (الف) آشناسازی دانشجویان با اصول و قواعد دانش کامپیوتر، (ب) تدریس مطالب مفید و آموزنده جهت ارتقاء سطح یادگیری در دوره‌های آموزشی بعدی و (ج) افزایش توان دانشجویان در ارائه استدلال‌های صوری، دقیق و صحیح ریاضی اشاره کرد. روش ارائه مطالب در این کتاب دو هدف را برآورده می‌کند و می‌توان اثبات کرد که در بدست آوردن هدف سوم نیز موفق است. جهت تفهیم ایده‌ها و درک بهتر مطالب، اصل زیربنایی در تهیه این کتاب، ایجاد انگیزه‌های شهودی و تشریح نظریه‌ها از طریق مثال‌ها می‌باشد. برای دانشجویان، استفاده از استدلال‌های ساده بر استدلال‌های کوتاه نکته‌دار، که دارای مفاهیم مشکل می‌باشند، ترجیح داده شده است. در این کتاب تعاریف و قضایا به دقت بیان و غالباً از ارائه جزئیات خسته‌کننده و تکراری صرفنظر شده است. این کار اهداف آموزشی مدنظر نویسنده را بهتر تأمین می‌کند، چون بسیاری از اثبات‌های کتاب، کاربردهای مکرر استقراء یا برهان خلف با تفاوت‌های جزئی هستند. ارائه

کامل این استدلال‌ها، مانع از جریان عادی مطالب آموزشی کتاب می‌شود. از اینرو، تکمیل برخی از اثبات‌ها به خواننده واگذار شده است. هرچند این کار باعث ناامیدی آن دسته از دانشجویانی می‌شود که همواره در پی یادگیری راحت و بدون دردسر مطالب هستند، اما کسب مهارت‌های ریاضی با رونویسی از کتاب حل‌مسئله حاصل نمی‌شود، بلکه باید با دقت بر روی نکته مسئله، نیازهای اثبات را در ذهن تثبیت نمود. این مهارت‌ها اکتسابی است و تصور می‌شود که روش کلی‌گویی اثبات‌ها در این کتاب، نقطه شروع بسیار خوبی برای اینگونه تمرین‌های مهارتی می‌باشد.

اغلب دانشجویان رشته‌های کامپیوتر واحد درسی نظریه زبان‌ها و اتوماتا را غیرلازم، انتزاعی و در نتیجه، فاقد نتایج عملی می‌دانند. برای تغییر نظر این گروه از مخاطبین، تأکید اصلی کتاب بر روی آموزش از طریق حل مسأله استوار بوده تا نظر مثبت آنها جلب و در عین حال اصلاح شود. هدف اصلی از تعقیب روش حل یک مسأله آن است که دانشجویان اساساً مطالب را از طریق مثال‌های تشریحی گویا در رابطه با مسئله و ارتباط آن با قضایا و تعاریف، بیاموزند. برخی از مثال‌ها حاوی نکات مهمی بوده که باید بیشتر مورد توجه قرار گیرند. تمریناتی که بعنوان کار در منزل در انتهای هر بخش آمده، بخش عمده‌ای از بار آموزشی و فرآیند یادگیری کتاب را بر عهده دارند. این تمرینات با تشریح و تفسیر مطالب آموزشی کتاب، توان حل مسأله دانشجویان را در سطوح مختلف به چالش می‌کشند. برخی از تمرینات نسبتاً ساده و در واقع مکمل مطالب نیمه‌تمام متن بوده و دانشجویان باید آنها را تکمیل کرده یا یکی دو مرحله به پیش ببرند. تمرینات ستاره‌دار، بسیار مشکل بوده و فقط دانشجویان بسیار با انگیزه یا مستعد قادر به انجام آنها می‌باشند. البته بهترین گزینه، ترکیب هردو گروه تمرینات می‌باشد.

در برنامه‌های آموزشی دانشگاه‌های مختلف تلاش می‌شود تا ضمن گریز از افتادن به دام نظریه‌محوری، امکان کاربرد عملی مطالب آموزشی فراهم شود. امیدواریم کتاب حاضر هر دوجنبه نظری و عملی را تحت پوشش قرار دهد. بعلاوه، بهتر است اساتید محترم دانشجویان خود را، از سطح انتزاع مورد انتظار از آنها، برای تمرینات اثبات محور مطلع کنند. توصیه می‌شود که دانشجویان، به محض برخورد با عباراتی نظیر "اثبات کنید" یا "نشان دهید" ابتدا نحوه ایجاد اثبات‌های دقیق در مثال‌ها و قضایای کتاب را به خاطر آورده و سپس، استدلال دقیقی را ارائه کنند.

کتاب و مطالب آن برای یک ترم تحصیلی تهیه شده است. برخی بخش‌های کتاب را که با علامت * مشخص شده، می‌توان به اختیار و بدون آسیب به بقیه مطالب حذف کرد. بازهم تأکید می‌کنیم که کل مطالب مهم بوده و باید فراگرفته شوند. تغییرات انجام شده در ویرایش‌های قبلی جزئی بود. اما در این ویرایش سعی شده تا با بررسی کاستی‌های گذشته، برخی مطالب مهم و جدید گنجانده شود. همچنین تلاش شده تا در ارائه مطالب مشکل از روش‌های ساده‌ای استفاده شود. متأسفانه در برخی موارد همچون لم تزریق مشکلات یادگیری ذاتی هستند. هرچند این مطالب در استدلال‌های دقیق چندان مفید نیست، اما در اثبات‌های صوری بسیار مفید خواهد بود.

پیتر لینز

پیش گفتار مترجمین

خداوند منان را شکر می‌گوییم که با اعطای نعمت حیات و عنایت او، توفیق آن را یافتیم تا کتابی تحت عنوان مقدمه‌ای بر نظریه زبانها و اتوماتا نوشته پیتر لینز را ترجمه و به دانشجویان علاقمند تقدیم کنیم. با توجه به نیاز مبرم دانشجویان رشته علوم و مهندسی‌کامپیوتر و مهندسی فناوری‌اطلاعات برای یادگیری نظریه زبانها و اتوماتا، بر آن شدیم، که تجربه چندین ساله خود در زمینه تدریس این درس در قالب ترجمه‌ای شیوا از کتابی که یکی از بروزترین و بهترین منابع می‌باشد در اختیار دانشجویان عزیز قرار دهیم. می‌توان گفت که کتاب حاضر به جرات بهترین منبع برای واحد درسی نظریه زبانها و اتوماتا (نظریه زبان‌ها و ماشین‌ها) در مقطع کارشناسی می‌باشد. این کتاب سعی دارد که هر دو جنبه درس یعنی جنبه نظری و عملی را تحت پوشش خود قرار دهد. که به اعتقاد ما تا حدود زیادی نیز در این راه موفق بوده است.

این کتاب ترجمه آخرین ویرایش کتاب اصلی می‌باشد که به همراه تمرینات جدید و کامل‌تر، دسته‌بندی بهتری برای تمرین‌ها توسط مولف محترم صورت گرفته است و همچنین با پاسخ به تعدادی از تمرین‌ها به صورت انتخابی سعی در تکمیل آموزش دارد، مولف سعی کرده برای درک بهتر مطالب بسیاری از مسائل و اثبات ها را به صورت الگوریتم و به زبان ساده ارائه کند. که تا حدود زیادی نیز موفق عمل کرده است. کتاب اصلی دارای چهارده فصل می‌باشد که از فصل دوازده تا چهارده مربوط به درس بعدی یعنی نظریه محاسبات می‌باشد بنابراین ما از فصل یک تا یازده را ترجمه کردیم تا سرفصل کاملی را برای درس نظریه زبان‌ها و اتوماتا پوشش داده باشیم. همچنین برای تکمیل فرآیند آموزش، تعدادی سوال چهارگزینه‌ای که برگزیده‌ای از آزمون‌های دانشگاه‌ها می‌باشند در انتهای کتاب قرار داده‌ایم. و امید است در چاپ‌های بعدی، این مجموعه سوالات چهارگزینه‌ای را کامل‌تر کنیم.

در پایان لازم است از تلاش‌ها و کمک‌های سرکار خانم مهندس ایمانی‌مهر که در بخشی از ویراستاری علمی، ما را یاری کردند نهایت تشکر و قدردانی را داشته باشیم. همچنین از سرکار خانم نیلوفر عزتی که ما را در تهیه مطالب کتاب از منبع اصلی و ترجمه جمله به جمله براساس قوانین ترجمه یاری کردند، و تمامی عزیزانی که با کمک‌های معنوی خود در پیمودن این راه ما را یاری کرده‌اند، کمال تشکر را داشته باشیم.

بر خویش وظیفه می‌دانیم قدردانی خود، از مدیریت محترم انتشارات نص جناب آقای مهندس زارع را صمیمانه ابراز کنیم که حسن ظن ایشان به جامعه دانشگاهی و دقت فراوان ایشان در تک‌تک مراحل از حروفچینی تا چاپ و صحافی موجب شد تا این اثر با کمترین کاستی ارائه گردد. همچنین از کادر محترم انتشارات نص، به خصوص خانم‌ها رفیعی، محمدی و خانجانی که دلسوزانه ما را در این راه یاری کردند، تشکر می‌کنیم.

هرچند به جرات می‌توانیم ادعا کنیم که برخی از اشکالات موجود در منبع زبان اصلی را با مولف محترم در میان گذاشتیم و در این ترجمه رفع کرده‌ایم ولی برای کتابی با این حجم فراوان فرمول، شکل و نشانه‌ها نمی‌توان ادعا کرد که بدون خطا باشد، از اینرو انتظار داریم به زینت نقد و بررسی همکاران عزیز، دانش‌پژوهان و دانشجویان مزین گردد، این عزیزان می‌توانند با ارسال یادداشت و نظرات خود برای غنی‌سازی هرچه بیشتر مطالب و پربارتر کردن این کتاب، ما را یاری کنند تا در چاپ‌های بعدی مورد استفاده قرار گیرد، در پایان از درگاه خداوند منان مسئلت داریم این خدمت خالصانه را، در راه رضای خود از ما بپذیرد و ما را یاری نموده.و نیرویی مرحمت فرماید تا بتوانیم در خدمت به دانش‌پژوهان عزیز، به پاره‌ای از آنچه آرزو داریم، تحقق بخشیم.

جعفر تنها
tanha@pnu.ac.ir

محمد سلیمانی تبار
soleimanitabar@pnu.ac.ir

تابستان ۸۸

فهرست مطالب

فصل ۱

مقدمه‌ای بر نظریه محاسبات

موضوع مورد بحث در این کتاب، یعنی نظریه محاسبات، سرفصل‌های متنوعی از جمله نظریه اتوماتاها، گرامرها و زبانهای صوری، محاسبه‌پذیری و پیچیدگی را شامل می‌شود. این موضوعات درمجموع پایه نظری علوم کامپیوتر را تشکیل می‌دهند. به بیان ساده‌تر، اتوماتاها، گرامرها و محاسبه‌پذیری را در مجموع می‌توان مطالعه توانایی‌های کلی کامپیوترها تلقی کرد؛ در حالیکه پیچیدگی بیشتر به جنبه‌های عملی اشاره دارد. در این کتاب، تقریباً در همه‌جا روی بخش اول تمرکز کرده‌ایم. به همین دلیل، اتوماتاهای مختلف را مطالعه کرده، نحوه ارتباط آنها با زبان‌ها و گرامرها را مورد بررسی قرار داده و راجع به قابلیت‌های کامپیوترهای دیجیتالی و همچنین محدودیت‌های آنها بحث خواهیم کرد. اما باید به‌خاطر داشته باشید که این نظریه، علیرغم کاربردهای متعدد خود، ذاتاً انتزاعی و ریاضی است.

علوم کامپیوتر یک رشته عملی محسوب می‌شود. به همین دلیل، اهالی این رشته اغلب ترجیح می‌دهند خود را درگیر حل مسائل مفید و ملموس کنند تا پیش‌بینی‌های نظری. این امر خصوصاً در مورد دانشجویان علوم کامپیوتر صدق می‌کند، بطوریکه اغلب آنها با کاربردهای مشکل این علم در دنیای واقعی درگیر هستند و فقط در صورتی به سراغ مسائل نظری می‌روند که آن مسائل، به آنها در یافتن راه حل‌های مناسب کمک کند. در واقع بدون این کاربردها، کامپیوترها تبدیل به دستگاه‌هایی بدون جذابیت و کسل‌کننده می‌شوند. اما همین دیدگاه پذیرفته شده، ممکن است این سؤال را هم مطرح کند که "پس چه لزومی به دانستن نظریه‌ها وجود دارد؟"

اولین پاسخ این است که نظریه‌ها، مفاهیم و اصولی را می‌سازند که به ما در درک ماهیت کلی این رشته کمک خواهند کرد. حوزه علوم کامپیوتر شامل طیف گسترده‌ای از موضوعات خاص، از طراحی ماشین تا برنامه‌ریزی آن، می‌باشد. استفاده از کامپیوترها در دنیای واقعی مستلزم رعایت جزئیاتی است

که آگاهی از آنها باعث کاربرد موفقیت آمیز کامپیوترها می‌شود. همین ظرافت، علوم کامپیوتر را تبدیل به رشته‌ای بسیار متنوع و گسترده کرده است. در این میان، برخی اصول زیربنایی و مشترک هم وجود دارند. برای مطالعه این اصول بنیادی، در این کتاب مدل‌های انتزاعی از کامپیوترها و محاسبات ارائه نموده‌ایم. مدل‌های ارائه شده ویژگی‌های مهم سخت‌افزاری و نرم‌افزاری را به تصویر می‌کشند که جزء جدایی‌ناپذیر بسیاری از ساختارهای پیچیده و ویژه‌ای هستند که در حین کار با کامپیوتر با آنها مواجه خواهیم شد. حتی در صورتی که این مدل‌ها برای به‌کارگیری فوری در دنیای واقعی بسیار ساده و پیش پا افتاده به نظر برسند، برداشت‌های ناشی از مطالعه آنها اساس برخی پیشرفت‌ها را ایجاد می‌کند. البته این روش در علوم کامپیوتر بی‌سابقه نیست. مدل‌سازی، یکی از ضروریات هر رشته علمی محسوب می‌شود؛ بعلاوه، سودمندی بسیاری از رشته‌ها، غالباً وابسته به وجود قوانین و نظریات ساده، اما قدرتمند است.

دومین پاسخ این است که ایده‌های مطرح شده در این کتاب کاربرد مستقیم و مهمی در طراحی دیجیتالی، زبان‌های برنامه‌سازی، کامپایلرها و ... دارد. مفاهیم مورد بحث، مانند یک تاروپودی ظریف، تقریباً در سرتاسر بافت زیبای قالی علوم کامپیوتر، از سیستم‌های عامل گرفته تا شناسایی الگو، کشیده شده است.

پاسخ سوم، که امیدواریم خواننده را کاملاً متقاعد کند، اینکه موضوع مورد بحث از نظر ذهنی محرک و جالب است. بعلاوه، مسائل چالشی و معمادار متعددی را مطرح می‌کند که ممکن است مدتها ذهن شما را به خود معطوف کند؛ و این همان حل مسئله به معنای واقعی کلمه است.

در این کتاب، نگاهی به برخی مدل‌هایی خواهیم داشت که ویژگی‌های اساسی تمامی کامپیوترها و کاربردهای آنها را مطرح می‌کنند. برای مدل‌سازی سخت‌افزار کامپیوتر، ایده **اتومات** (جمع آن را **اتوماتا می‌گوئیم**) را مطرح کرده‌ایم. اتومات ساختاری است که تمام ویژگی‌های لازمه یک کامپیوتر دیجیتالی امروزی را دارا می‌باشد. اتومات، ورودی را پذیرفته و خروجی را تولید می‌کند. همچنین ممکن است دارای یک حافظه موقت بوده و تصمیماتی راجع به تبدیل ورودی به خروجی در صورت نیاز اتخاذ نماید. زبان صوری، مدل انتزاعی از ویژگی‌های عمومی زبان‌های برنامه‌سازی است. در تعاریف این کتاب یک زبان صوری مجموعه‌ای از سمبل‌ها و تعدادی قانون است که می‌توان از آنها برای ترکیب سمبل‌ها و ایجاد موجودیت‌هایی به نام جمله استفاده نمود. هرچند برخی از زبان‌های صوری مورد مطالعه در این کتاب ساده‌تر از زبان‌های برنامه‌سازی هستند، ولی در بسیاری از ویژگی‌های اساسی با هم مشترک می‌باشند. بررسی زبان‌های صوری امکانی را برای یادگیری زبان‌های برنامه‌سازی در اختیار خواننده قرار می‌دهد. سپس، با ارائه تعریف دقیقی از اصطلاح الگوریتم، مفهوم محاسبه مکانیکی را مدون کرده و انواع مسائلی را مطالعه می‌کنیم که این ابزار مکانیکی برای حل آنها مناسب است (یا نیست). در جریان مطالعه، ارتباط نزدیک بین این انتزاع‌ها را نشان داده و نتایج احتمالی آنها را بررسی خواهیم نمود.

در فصل یک، با بررسی مفصل این نظریات اساسی، سعی می‌کنیم تا زمینه برای کارهای بعدی

فراهم شود. در بخش ۱-۱، برخی نظریات اساسی در ریاضیات را که ممکن است بعداً مورد استفاده قرار گیرند، مطالعه می‌کنیم. هرچند همواره در کشف ایده‌ها از شهود استفاده می‌کنیم، نتایج براساس استدلال‌های دقیق استوار هستند. از آنجایی‌که این‌کار تا حدی نیازمند دانش ریاضی است، خواننده باید درک مناسب و کافی از اصطلاحات و لغات بکار رفته و نتایج ابتدایی نظریه مجموعه‌ها، توابع و روابط داشته باشد. هر چند به دفعات از ساختارهای درختی و گراف استفاده شده، خواننده غالباً به چیزی بیش از دانستن تعریف گراف جهت‌دار و برچسب‌دار نیاز نخواهد داشت. شاید ضروری‌ترین ویژگی برای موفقیت، توان دنبال‌کردن اثبات‌ها و درک جزئیات استدلال ریاضی مناسب باشد. این کار مستلزم آشنایی با روشهای اصلی اثبات استنتاج، استقراء و اثبات با استفاده از برهان خلف است. البته فرض می‌کنیم که خواننده این پیش‌زمینه‌ها را دارد. بخش ۱-۱ برخی از نتایج اصلی مورد استفاده را مطرح و زمینه‌ای برای ایجاد تعریف صوری مجموعه‌ها از طریق نشانه‌های ریاضی را، برای مباحث بعدی فراهم می‌کند.

در بخش ۱-۲، نگاه اولیه‌ای به مفاهیم کلیدی زبان‌ها، گرامرها و اتوماتا خواهیم داشت. این مفاهیم به صورت‌های مختلف و متعدد در سرتاسر کتاب تکرار می‌شوند. در بخش ۱-۳، برخی کاربردهای ساده این نظریات کلی را ارائه می‌دهیم تا خوانندگان عملاً با کاربردهای گسترده این مفاهیم در علوم کامپیوتر آشنا شوند. بحث در این دو بخش بیشتر شهودی است. در ادامه، این نکات را روشن‌تر خواهیم کرد، اما فعلاً هدف ما بیشتر شفاف‌سازی مفاهیم مورد بحث است.

مجموعه علائم و مقدمات ریاضی

مجموعه‌ها

مجموعه گروهی از اعضاء است که ساختاری غیر از عضویت ندارند. می‌گوییم x متعلق به مجموعه S است و می‌نویسیم $x \in S$. بالعکس، عبارت $x \notin S$ به این معناست که x متعلق به مجموعه S نیست. مجموعه‌ها را می‌توان با بیان توصیفی از اعضای آن در کروشه مشخص کرد. به عنوان مثال، مجموعه اعداد صحیح ۰، ۱و۲ را به صورت

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

نمایش می‌دهیم. در صورتی که مجموعه همه اعضای مجموعه‌ای مشخص بوده و تعداد آن زیاد باشد برای جلوگیری از اتلاف وقت از نقطه‌چین در توصیف مجموعه استفاده می‌شود. بنابراین، $\{a,b,\ldots,z\}$ به معنای تمام حروف کوچک انگلیسی و $\{2,4,6,\ldots\}$ بیانگر مجموعه تمام اعداد صحیح زوج و مثبت است. در صورت لزوم، می‌توان از تعاریف مشخص‌تر نیز استفاده کرد و برای مثال اخیر نوشت:

$$S = \{i : i > 0 \text{ , } i \text{ زوج است}\}. \qquad (۱-۱)$$

عبارت بالا را به این صورت می‌خوانیم: " S مجموعه تمام i های زوج بزرگتر از صفر است" که البته بطور ضمنی صحیح بودن عدد i را هم در خود دارد.

عملگرهای معمول بر روی مجموعه‌ها شامل اجتماع ($\cup$)، اشتراک ($\cap$) و تفاضل ($-$) است که به صورت زیر تعریف می‌شوند:

$$S_1 \cup S_2 = \{x : x \in S_1 \text{ or } x \in S_2\},$$
$$S_1 \cap S_2 = \{x : x \in S_1 \text{ and } x \in S_2\},$$
$$S_1 - S_2 = \{x : x \in S_1 \text{ and } x \notin S_2\}.$$

عملگر مهم دیگر **مکمل‌گیری** است. مکمل مجموعه S بصورت $\overline{S}$ نشان داده شده و شامل تمام عناصر غیرموجود در S است. برای درک بهتر موضوع، مجموعه جهانی U، شامل تمام اعضاء ممکنه، را معرفی می‌کنیم. در صورت مشخص بودن U، آنگاه

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

مجموعه تهی یا مجموعه پوچ مجموعه‌ای است که هیچ عضوی نداشته و با $\emptyset$ نمایش داده می‌شود. از تعریف مجموعه می‌توان نتیجه گرفت که

$$S \cup \emptyset = S - \emptyset = S,$$
$$S \cap \emptyset = \emptyset,$$
$$\overline{\emptyset} = U,$$
$$\overline{\overline{S}} = S.$$

همانطور که می‌دانید قوانین مفید زیر، که بنام **قانون دمورگان** خوانده می‌شوند، کاربرد گسترده‌ای داشته و در موارد متعدد در این کتاب هم مورد استفاده قرار خواهند گرفت.

$$\overline{S_1 \cup S_2} = \overline{S}_1 \cap \overline{S}_2, \qquad (۱-۲)$$

$$\overline{S_1 \cap S_2} = \overline{S}_1 \cup \overline{S}_2, \qquad (۱-۳)$$

اگر تمام اعضای S_1 عضو S باشند، مجموعه S_1 **زیرمجموعه** S خوانده می‌شود و می‌نویسیم

$$S_1 \subseteq S.$$

اگر $S_1 \subseteq S$، اما یکی از اعضای S در S_1 موجود نباشد، می‌گوییم که S_1 **زیرمجموعه محض** S است و می‌نویسیم

$$S_1 \subset S$$

اگر S_1 و S_2 هیچ عضو مشترکی نداشته باشند، یعنی $S_1 \cap S_2 = \emptyset$، می‌گوییم که دومجموعه جدا از هم هستند.

یک مجموعه اگر حاوی تعداد متناهی از اجزاء باشد، **مجموعه متناهی** و در غیر اینصورت **مجموعه نامتناهی** نامیده می‌شود. اندازه یک مجموعه متناهی برابر با تعداد اعضاء موجود در آن است و بصورت $|S|$ نمایش داده می‌شود.

مجموعه‌ها معمولاً تعداد زیادی زیرمجموعه دارند. مجموعه تمام زیرمجموعه‌های مجموعه مفروض S، **مجموعه توانی** S خوانده شده و به صورت 2^S نشان داده می‌شود. توجه کنید که 2^S مجموعه‌ای است که اعضای آن هر کدام، خود نیز مجموعه هستند.

مثال ۱-۱

اگر S مجموعه $\{a,b,c\}$ باشد، آنگاه مجموعه توانی آن

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

خواهد بود. در اینجا $|S| = 3$ و $|2^S| = 8$ است. این نمونه‌ای از یک نتیجه‌گیری کلی است؛ اگر S متناهی باشد، آنگاه

$$|2^S| = 2^{|S|}.$$

■

در اکثر مثال‌های این کتاب، اعضاء یک مجموعه، دنباله‌ای از زوج مرتب‌های اعضای مجموعه‌های دیگر هستند. این دست مجموعه‌ها حاصل **ضرب دکارتی** مجموعه‌های دیگر خوانده می‌شوند. برای نمایش حاصل ضرب دکارتی دو مجموعه، که خود مجموعه‌ای از زوج‌های مرتب است، می‌نویسیم

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

مثال ۱-۲

فرض کنیم $S_1 = \{2,4\}$ و $S_2 = \{2,3,5,6\}$. آنگاه

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

یادآوری می‌کنیم که ترتیب نوشتن اجزای یک زوج مهم است. مثلاً، زوج $(4,2)$ متعلق به $S_1 \times S_2$ است، ولی زوج $(2,4)$ در آن وجود ندارد. حاصل ضرب دکارتی را می‌توان روی بیش از دو مجموعه نیز تعریف نمود:

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

■

مجموعه‌ها را می‌توان از طریق جداسازی به چند زیرمجموعه مجزا تقسیم کرد. فرض کنید که S_1، S_2، ... و S_n زیرمجموعه‌های مجموعه مفروض S بوده و شرایط زیر برقرار باشد:

زیرمجموعه‌های S_1، S_2، ... و S_n دو به دو جدا از هم هستند و

$S_1 \cup S_2 \cup \ldots \cup S_n = S$ ،
هیچکدام از S_i ها تهی نیست.

آنگاه S_1، S_2، ... و S_n یک افراز(پارتیشن) از S خوانده می‌شوند.

توابع و روابط

تابع قانون یا ضابطه‌ای است که به هریک از اعضاء یک مجموعه، عضو منحصر به فردی از مجموعه دیگر را نسبت می‌دهد. اگر f بیانگر یک تابع باشد، آنگاه مجموعه اول **دامنه** f و مجموعه دوم **برد** آن خوانده می‌شود و می‌نویسیم

$$f : S_1 \to S_2$$

به این معنا که دامنه f زیرمجموعه‌ای از S_1 و برد f زیرمجموعه‌ای از S_2 است. اگر همه اعضای S_1، دامنه f باشند، f **تابع تام** روی S_1 و در غیر اینصورت f **تابع جزئی** خوانده می‌شود.

در بسیاری از کاربردها، دامنه و برد توابع مورد استفاده، در مجموعه اعداد صحیح مثبت قرار دارند. بعلاوه، به دلیل طولانی بودن استدلال‌های توابع، ما غالباً فقط به رفتار آنها علاقه‌مند هستیم. در این موارد باید درکی از سرعت رشد در اختیار داشته و از یک مجموعه نشانه‌های مرتبه رشد رایج استفاده کنیم. فرض کنید که $f(n)$ و $g(n)$ دو تابع با دامنه‌ای از زیرمجموعه اعداد صحیح مثبت باشند. اگر عدد ثابت مثبت c وجود داشته باشد که برای همه n های بزرگ

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

می‌گوییم که **مرتبه رشد** f حداکثر برابر g است و می‌نویسیم

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

اگر

$$|f(n)| \ge c|g(n)|,$$

آنگاه **مرتبه رشد** f حداقل برابر g است و می‌نویسیم

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

در نهایت، اگر اعداد صحیح و مثبت c_1 و c_2 وجود داشته باشند، بطوریکه

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

می‌گوییم f و g دارای مرتبه بزرگی یکسانی بوده و می‌نویسیم

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

در بحث مرتبه، با رشد n می‌توان از ضرایب ثابت و عبارت‌های با مرتبه پایین‌تر صرف‌نظر نمود.

مثال ۱-۳

قرض کنید

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

آنگاه

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

در نماد مرتبه، علامت = نباید تساوی تلقی شود، چون نمی‌توان بنا نماد مرتبه مانند عبارات معمولی برخورد کرد. نتیجه‌گیری‌هایی از قبیل

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

قابل قبول نبوده و ممکن است منجر به اشتباهات زیادی شود. همانطور که در فصل‌های بعد خواهیم دید، همین استدلال‌های مرتبه، در صورتی که به نحو مناسبی مورد استفاده قرار گیرند، می‌توانند بسیار مفید باشند.

■

برخی از توابع را می‌توان بصورت مجموعه‌ای از زوج‌ها نمایش داد، مثلاً

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

که در آن x_i عضوی از دامنه تابع و y_i مقدار متناظر آن در برد تابع است. برای اینکه چنین مجموعه‌ای، تابعی را تعریف کند، هریک از x_i ها مجاز هستند که حداکثر یک مرتبه به عنوان عضو اول زوج استفاده شوند. در غیر اینصورت، مجموعه **رابطه** خوانده می‌شود. همانطور که می‌دانید، روابط جامع‌تر از توابع هستند. مثلاً در یک تابع، هریک از اعضای دامنه فقط به یک عضو در برد مربوط می‌شود؛ اما در روابط ممکن است در برد چندین عضو به ازای یک عضو دامنه وجود داشته باشند.

هم‌ارزی یکی از انواع رابطه‌هاست که در واقع صورت کلی از مفهوم تساوی (تشابه) است. برای بیان رابطه هم‌ارزبودن زوج (x, y) می‌نویسیم

$$x \equiv y.$$

روابطی که بوسیله ≡ نمایش داده می‌شوند در صورتی هم ارز خوانده می‌شوند که از سه قانون زیر تبعیت کنند:

- قانون انعکاسی

$x \equiv x$ برای تمام x ها؛

- قانون تقارنی

اگر $x \equiv y$، آنگاه $y \equiv x$؛

- قانون انتقالی (تراگذاری)

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

مثال ۱-۴

روی مجموعهٔ اعداد صحیح غیرمنفی، میتوان رابطه

$$y \equiv x$$

را در صورتی تعریف کرد که اگر و تنها اگر

$$y \bmod 3 = x \bmod 3.$$

در اینصورت $۵ \equiv ۲$ ، $۱۲ \equiv ۰$ و $۳۶ \equiv ۰$. این رابطه به دلیل داشتن سه خاصیت انعکاسی، تقارنی و انتقالی یک رابطه هم‌ارزی است.

■

اگر S مجموعه‌ای باشد که بتوان رابطهٔ هم‌ارزی را روی آن تعریف کرد، آنگاه می‌توان از این هم‌ارزی برای پارتیشن‌بندی مجموعه به **کلاس‌های هم‌ارزی** استفاده کرد. هر کلاس هم ارزی شامل کلیه اعضای هم‌ارز است.

گراف‌ها و درخت‌ها

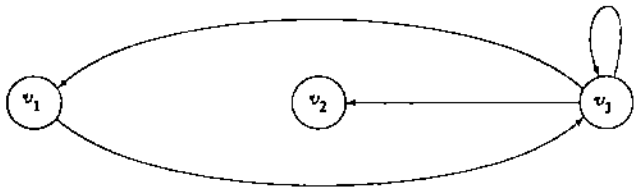
گراف، یک ساختار شامل دو مجموعه متناهی، یعنی مجموعه $V = \{v_1, v_2, \ldots, v_n\}$ از رئوس و مجموعه $E = \{e_1, e_2, \ldots, e_m\}$ از یال‌ها، می‌باشد و هر یال بوسیله زوجی از رئوس V ، نمایش داده می‌شود؛ به عنوان مثال

$$e_i = (v_j, v_k)$$

یالی از v_j به v_k است. e_i یال خروجی برای رأس v_j و یال ورودی برای رأس v_k می‌باشد. از آنجایی‌که به ازای هر یال یک جهت (از v_j به v_k) مشخص می‌کنیم، گراف‌های مورد بحث ما در واقع جهت‌دار هستند. یال‌ها و رئوس گراف‌ها را می‌توان برچسب‌دار (وزن‌دار) کرد. این برچسب‌ها در واقع نام یا اطلاعات دیگری در مورد اجزای گراف هستند.

گراف‌ها را می‌توان توسط نمودارهایی نمایش داد که در آنها، رئوس به صورت دوایر و یال‌ها به صورت خطوط جهت‌دار برای اتصال رئوس به هم بکار برده می‌شوند. در شکل ۱-۱، گرافی را با

رئوس $\{v_1, v_2, v_3\}$ و یال‌های $\{(v_1, v_2), (v_2, v_3), (v_3, v_1), (v_3, v_3)\}$ مشاهده می‌کنید.



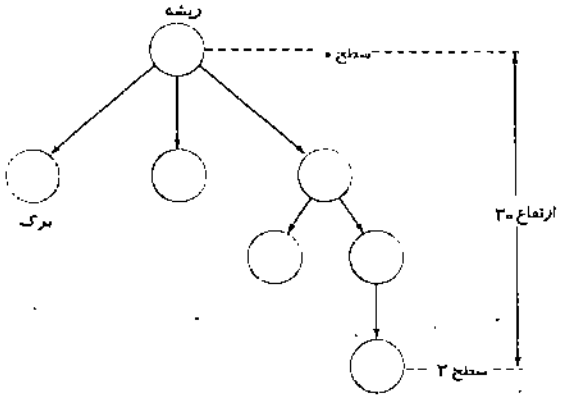
شکل ۱-۱

دنباله یال‌های $(v_1, v_2), \ldots, (v_1, v_1), (v_1, v_2), (v_2, v_3)$، یک قدم از v_i به v_n خوانده می‌شود. طول هر قدم تعداد کل یال‌های پیموده شده در طی مسیر از رأس شروع به رأس پایان است. قدمی که در آن هیچ یال تکرار نشود اصطلاحاً مسیر نام دارد. یک مسیر در صورتی ساده است که هیچ رأسی در آن تکرار نشود. قدم از v_i به خود آن، بدون تکرار هیچیک از یال‌ها، یک حلقه با پایه v_i خوانده می‌شود. حلقه ساده، حلقه‌ای است که در آن هیچ رأسی غیر از پایه تکرار نشود. در شکل ۱-۱، $(v_1, v_2), (v_2, v_3), (v_3, v_1)$ یک مسیر ساده از v_1 به v_2 است. دنباله یال‌های $(v_1, v_2), (v_2, v_3), (v_3, v_1), (v_1, v_2)$ یک حلقه است، ولی ساده نیست. در صورت برچسب‌دار بودن یال‌های گراف، می‌توان راجع به برچسب قدم صحبت کرد. این برچسب، دنباله برچسب یال‌ها در سرتاسر مسیر است. کلام آخر اینکه، یک یال از یک رأس به خود آن طوقه نامیده می‌شود. در شکل ۱-۱ یک طوقه روی رأس v_3 وجود دارد.

در بسیاری از موارد، می‌توان برای پیدا کردن تمام مسیرهای ساده بین دو رأس مفروض (یا تمام دورهای ساده روی یک رأس) از الگوریتم استفاده کرد. اگر مسأله کارآیی را اولویت اصلی خود در انتخاب الگوریتم قرار ندهیم، می‌توان از روش مشخص زیر استفاده کرد. با شروع از یک رأس فرضی، مثلاً v_i، تمام یال‌های خروجی $\ldots, (v_i, v_1), (v_i, v_2), (v_i, v_3)$ را لیست می‌کنیم. در اینصورت، تمام مسیرهای با طول یک و مبدأ v_i را در اختیار خواهیم داشت. برای تمام رئوس $v_1, v_2, \ldots$ که به این ترتیب بدست آمدند، تمام یال‌های خروجی را، بجز یال‌هایی که رأس بعدی آنها جزء رئوسی است که نام آنها در مسیرهای ساخته شده وجود دارد، لیست می‌کنیم. به این ترتیب، کلیه مسیرهای ساده با طول دو ازمبدأ v_i را پیدا خواهیم نمود. اینکار را تا پیدا کردن تمام مسیرهای ممکن ادامه می‌دهیم. به دلیل متناهی بودن تعداد رئوس، در نهایت لیستی از تمام مسیرهای ساده با مبدأ v_i را در اختیار خواهیم داشت. سپس، می‌توان از بین آنها به راحتی رئوس منتهی به رأس مورد نظر را انتخاب کرد.

درخت – که نوع خاصی از گراف محسوب می‌شود– یک گراف جهت‌دار بدون دور با رأس مشخصی به نام ریشه است، بطوریکه دقیقاً یک مسیر از ریشه به هریک از رئوس دیگر آن وجود دارد. این تعریف به طور ضمنی بیان می‌کند که ریشه فاقد یال‌های ورودی بوده و بعلاوه، برخی رئوس فاقد یال‌های خروجی خواهند بود. این رئوس **برگهای** درخت نامیده می‌شوند. در صورتیکه یک یال از v_i به v_j وجود داشته باشد، آنگاه v_i **پدر** v_j و v_j **فرزند** v_i خوانده می‌شود. **سطح** متناظر با هر رأس، برابر

با تعداد یال‌هایی است که در مسیر موجود از ریشه تا آن رأس وجود دارند. ارتفاع درخت برابر با ماکزیمم تعداد سطح هر یک از رئوس است. با بررسی شکل ۱-۲ این مفاهیم را بهتر درک خواهید کرد.



شکل ۱-۲

در برخی موارد لازم است که برای هر یک از گره‌های موجود در یک سطح از ترتیب خاصی استفاده کنیم. در این موارد موضوع درخت‌های مرتب مطرح می‌شود. جزئیات بیشتر درمورد گراف‌ها و درخت‌ها را می‌توانید در بسیاری از منابع ریاضیات گسسته(ساختمان‌های گسسته) مطالعه کنید.

روشهای اثبات

یکی از شروط مهم برای مطالعه و درک مفاهیم این کتاب، توانایی پی‌گیری اثبات‌ها می‌باشد. در استدلال‌های ریاضی، ما از قوانین پذیرفته شده منطق استنتاجی استفاده کرده و به همین دلیل بسیاری از اثبات‌ها صرفاً دنباله‌ای از مراحل این منطق هستند. بطورکلی، از دو روش اثبات، به نام **اثبات با استفاده از استقراء** و **اثبات با استفاده از برهان خلف** استفاده می‌کنیم. در ادامه، مختصراً راجع به این دو روش بحث خواهیم کرد.

در روش استقراء، صحت تعدادی از عبارات را می‌توان از صحت چند مورد خاص نتیجه گرفت. فرض کنید که بخواهیم صحت دنباله‌ای از عبارات $P_1, P_2, P_3, \ldots$ و P_n را ثابت کنیم. بعلاوه، فرض کنید که شرایط زیر برقرار است:

۱. به ازای برخی از $k \geq 1$، می‌دانیم که $P_1, P_2, P_3, \ldots$ و P_k صحیح هستند.

۲. مسأله به گونه‌ای است که به ازای هر $n \geq k$، صحت $P_1, P_2, P_3, \ldots$ و P_n دلالت بر صحت P_{n+1} دارد.

بنابراین می‌توانیم از استقراء برای نمایش صحت هریک از عبارات این دنباله استفاده کنیم.

در روش اثبات براساس استقراء، به این ترتیب بحث می‌کنیم: از شرط یک می‌دانیم که اولین k عبارت، صحت دارند. سپس شرط دو به ما می‌گوید که P_{k+1} هم باید صحت داشته باشد. اما حالا که می‌دانیم اولین $k+1$ عبارت صحت دارند، می‌توانیم مجدداً از شرط دو برای ادعای صحت P_{k+2} استفاده کنیم و تا آخر با همین الگوی استدلال ادامه می‌دهیم. از همین دنباله استدلال‌ها، می‌توان در مورد هر عبارتی استفاده کرده و هر یک را اثبات نمود.

جملات آغازین $P_1, P_2, P_3, \ldots$ و P_k پایه استقراء نامیده می‌شوند. مرحله‌ای که P_n را به P_{n+1} متصل می‌کند گام استقراء نامیده می‌شود. گام استقراء بطورمعمول به کمک **فرض استقراء** یعنی صحیح بودن $P_1, P_2, P_3, \ldots$ و P_n، ساده‌تر می‌شود. سپس استدلال می‌کنیم که صحت این عبارات، صحت P_{n+1} را تضمین می‌کند. در استدلال‌های استقرایی صوری، هرسه بخش را بطور آشکار اثبات می‌کنیم.

مثال ۱-۵

درخت دودویی، درختی است که در آن، هیچ پدری نمی‌تواند بیش از دو فرزند داشته باشند. اثبات کنید که یک درخت دودویی با ارتفاع n حداکثر 2^n برگ دارد.

اثبات: اگر حداکثر تعداد برگهای یک درخت دودویی با ارتفاع n را با $l(n)$ نشان دهیم، بنابراین باید نشان دهیم که $l(n) \leq 2^n$.

پایه : چون یک درخت با ارتفاع صفر نمی‌تواند گره‌ای غیر از ریشه داشته باشد، بنابراین $l(0) = 1 = 2^0$؛ پس حداکثر یک برگ دارد.

فرض استقرایی:

$$l(n) \leq 2^i \quad \text{به ازای } i = 0, 1, \ldots, n. \qquad (۱-۴)$$

گام استقراء: برای بدست آوردن یک درخت دودویی با ارتفاع $n+1$ از درختی با ارتفاع n، می‌توانیم حداکثر دو برگ را در محل هریک از برگهای قبلی ایجاد کنیم. بنابراین،

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

حال، با استفاده از فرض استقراء داریم که

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

بنابراین، با توجه به اینکه ادعای ما برای هر n صحت دارد باید برای $n+1$ نیز صحت داشته باشد. بدلیل آنکه n می‌تواند هر عددی باشد، این عبارت برای تمام n ها صحت خواهد داشت.

■

توجه کنید که نشانهٔ ■ در سرتاسر کتاب به معنای پایان اثبات و حل است.

درک استدلال استقرایی هرچند ممکن است تا حدودی مشکل باشد، اما ارتباط نزدیک بین استقراء و بازگشت را در برنامه‌نویسی نشان می‌دهد. به عنوان مثال، تعریف بازگشتی تابع $f(n)$ ، که در آن n هرعدد صحیح مثبت است، غالباً از دو قسمت تشکیل می‌شود. یک قسمت شامل تعریف $f(n+1)$ در جملات $f(1), f(2), \ldots, f(n-1)$ و $f(n)$ است که همان گام استقراء می‌باشد. بخش دوم، "فرار" از بازگشتی، با تعریف غیربازگشتی $f(k)$ برای $f(1), f(2), \ldots$ محقق شده و متناظر با پایه استقراء است. این قسمت هم مانند استقراء، به ما امکان می‌دهد تا صرفاً با داشتن چند مقدار آغازین و بهره‌گیری از ماهیت بازگشتی مسأله، راجع به نمونه‌های مشابه یک مسأله نتیجه‌گیری کنیم.

گاهی اوقات یک مسأله در ظاهر بسیار پیچیده به نظر می‌رسد، اما با کمی، تغییر نگاه، همه چیز ساده خواهد شد. در اغلب موارد نگاه به مسأله به صورت بازگشتی تا حد زیادی باعث سادگی آن می‌شود.

مثال ۱-۶

مجموعه $l_1, l_2, \ldots, l_n$ متشکل از خطوط مستقیم دو به دو متقاطع، صفحه را به تعدادی ناحیه جداگانه تقسیم می‌کنند. یک خط، صفحه را به دو ناحیه تقسیم می‌کند، دو خط چهار ناحیه بوجود می‌آوردند، سه خط هفت ناحیه و الی آخر. این‌کار تا سه خط به راحتی قابل انجام است، اما با افزایش تعداد خطوط ارائه یک شکل شهودی نیز پیچیده و نامفهوم می‌شود. اجازه دهید این مسأله را به روش بازگشتی حل کنیم.

براساس شکل ۱-۳، مشاهده می‌کنید که با افزودن خط جدید l_{n+1} به n خط موجود، ناحیه واقع در سمت چپ l_1 و همچنین ناحیه واقع در سمت چپ l_1 به دو ناحیه جدید تقسیم شده و این‌کار تا آخرین خط ادامه پیدا می‌کند. در نتیجه، هریک از n تقاطع، یک ناحیه جدید ایجاد کرده و همچنین خط l_{n+1} ، ناحیه واقع در سمت راست l_n را نیز تقسیم و یک ناحیه جدید دیگر ایجاد می‌شود. بنابراین، اگر $A(n)$ را تعداد نواحی ایجاد شده بوسیله n خط در نظر بگیریم، مشاهده می‌کنیم که

$$A(n+1) = A(n) + n + 1, \quad n = 1, 2, \ldots.$$

و $A(1)=2$ از همین رابطه بازگشتی ساده می‌توان نتیجه گرفت که $A(2)=4$، $A(3)=7$ و $A(4)=11$ و الی آخر.

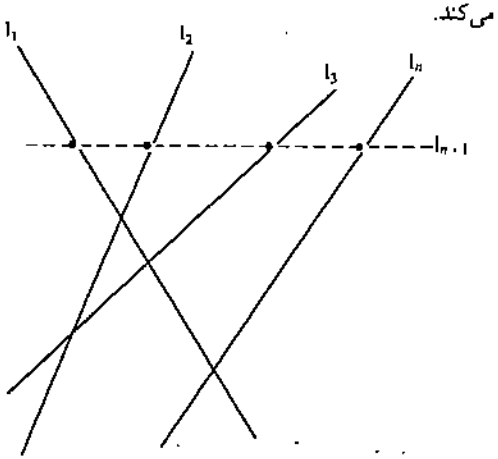
برای ارائه فرمول برای $A(n)$ و اثبات درستی آن، از روش استقراء استفاده می‌کنیم. اگر بپذیریم که

$$A(n) = \frac{n(n+1)}{2} + 1,$$

آنگاه

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

گام استقراء را تأیید می‌کند.



شکل ۱-۳

هرچند در این مثال در تشخیص پایه، فرض و گام استقراء چندان صوری عمل نکردیم، این ویژگی‌ها ضروری هستند. برای جلوگیری از صوری شدن بیش از حد استدلال‌های بعدی، غالباً استفاده از روش مثال دوم توصیه می‌شود. با این وجود، اگر در پی‌گیری یا ساخت مراحل اثبات مشکل دارید، بهتر است که از روش مشخص‌تر مثال ۱-۵ استفاده کنید.

■

اثبات از طریق برهان خلف روش قدرتمند دیگری است که، معمولاً در مواردی کاربرد دارد که روشهای دیگر قابل استفاده نباشند. فرض کنید بخواهیم صحت عبارت مفروض P را اثبات کنیم. بنابراین ابتدا فرض می‌کنیم که P نادرست بوده و همین فرض نادرست بودن P را دنبال می‌کنیم. اگر به این نتیجه برسیم که فرض ابتدایی مسئله نادرست بوده، بنابراین P حتماً درست است. مثال زیر یکی از نمونه‌های ابتدایی و بسیار خوب در این زمینه است.

مثال ۱-۷

عدد گویا، عددی است که بتوان آنرا به صورت $\sqrt{2} = \frac{n}{m}$، نسبت دو عدد صحیح n و m بیان کرد، بطوریکه n و m هیچ ضریب مشترکی نداشته باشند. عدد حقیقی که گویا نباشد، غیرگویا نامیده می‌شود. نشان دهید که $\sqrt{2}$ غیرگویا است.

مشابه تمام اثبات‌های دیگر براساس برهان خلف، ابتدا نقیض چیزی را که می‌خواهیم اثبات کنیم درست فرض می‌کنیم. در اینجا فرض می‌کنیم که $\sqrt{2}$ عدد گویا بوده و می‌توان آنرا به صورت (۱-۵) نوشت.

$$\sqrt{2}=\frac{n}{m}, \qquad (۵-۱)$$

که در آن، n و m، اعداد صحیح بدون عامل ضربی مشترک هستند. (۱-۵) را می‌توانیم به صورت

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

نیز بنویسیم. بنابراین n^2 باید زوج باشد و این به آن معناست که n نیز زوج می‌باشد. لذا می‌توان نوشت که $n=2k$ یا

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

و

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

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

■

این مثال نمونه‌ای از اثبات بوسیله برهان خلف بود. با ایجاد یک فرض خاص برای حکم، ما به سمت خلاف فرض یا واقعیت شناخته شده دیگری که از ابتدا وجود دارد، هدایت می‌شویم. در صورت صحت منطقی تمام مراحل استدلال باید نتیجه بگیریم که فرض اولیه ما، اشتباه بوده است، پس حکم درست است.

تمرین‌ها

۱. با استفاده از استقراء روی اندازه S نشان دهید که اگر S یک مجموعه متناهی باشد، آنگاه $|2^S|=2^{|S|}$

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

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

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

۴. فرض کنید رابطه بین دو مجموعه فقط و فقط هنگامی با S_1 $S_1\equiv S_2$ تعریف می‌شود که $|S_1|=|S_2|$. نشان دهید که این رابطه هم ارزی است.

۵. قوانین دمورگان و تساوی‌های (۱-۲) و (۱-۳) را اثبات کنید.

۶. در برخی موارد لازم است که از نشانه‌های اجتماع و اشتراک مشابه نشانهٔ جمع $\sum$ استفاده کنیم. مثلاً،

$$\bigcup_{P\in\{i,j,k,\dots\}} S_P=S_i\cup S_j\cup S_k\cdots$$

را با مجموعه نشانه‌های مشابه اشتراک چند مجموعه تعریف می‌کنیم. براساس این مجموعه علائم، قوانین کلی دمورگان به صورت

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

و

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

نوشته می‌شوند. این موجودیت‌ها را درحالی ثابت کنید که P مجموعه متناهی باشد.

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 تنها 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 قانون پخشی

$$S_1\cap(S_2\cup S_3)=(S_1\cap S_2)\cup(S_1\cap S_3)$$

در مورد مجموعه‌ها برقرار است.

۱۱. نشان دهید که

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

۱۲. نشان دهید که اگر $S_1\subseteq S_2$، آنگاه $\overline{S_2}\subseteq\overline{S_1}$.

۱۳. شرط لازم و کافی برای S_1 و S_2 بیابید بطوری که

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

۱۴. با استفاده از تساوی مثال ۴-۱، مجموعه $\{2,4,5,6,9,23,24,25,31,37\}$ را به کلاس‌های هم‌ارزی پارتیشن‌بندی کنید.

۱۵. نشان دهید که اگر $f(n)=O(g(n))$ و $g(n)=O(f(n))$، آنگاه $f(n)=\theta(g(n))$.

۱۶. نشان دهید که $2^{2n}=O(3^n)$ اما $2^{2n}\ne\theta(3^n)$.

۱۷. درستی نتایج مرتبه‌های زیر را نشان دهید.

الف) $n^3+5\log n=O(n^3)$

ب) $3^n = O(n!)$

ج) $n! = O(n^n)$

اثبات کنید که اگر $f(n) = O(g(n))$ و $g(n) = O(h(n))$، آنگاه $f(n) = O(h(n))$

نشان دهید که اگر $f(n) = O(n^2)$ و $g(n) = O(n^3)$، آنگاه

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

و

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

در این حالت، آیا رابطه $g(n) / f(n) = O(n)$ صحیح است؟

فرض کنید که $f(n) = 2n^3 + n$ و $g(n) = O(n^3)$. اشتباه استدلال زیر در کجاست؟

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

بطوریکه

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

بنابراین،

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

نشان دهید که اگر $f(n) = \theta(\log_2 n)$، آنگاه $f(n) = \theta(\log_{10} n)$

گرافی با رئوس $\{v_1, v_2, v_3\}$ و یال‌های $\{(v_1, v_1), (v_1, v_2), (v_2, v_3), (v_3, v_1), (v_1, v_1)\}$ رسم کنید. تمام حلقه‌ها با پایه v_1 را نام ببرید.

گراف $G=(V, E)$ مفروض است. ادعای زیر را اثبات کنید: در صورت وجود هر قدم بین $v_1 \in V$ و $v_2 \in V$، آنگاه باید یک مسیر با حداکثر طول $|V| - 1$ بین این دو رأس وجود داشته باشد.

گراف‌هایی با حداکثر یک یال بین هر دو رأس گراف را در نظر بگیرید. نشان دهید که در اینصورت، نموداری با n رأس، حداکثر n^2 یال خواهد داشت.

نشان دهید که

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

نشان دهید که

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

اثبات کنید که به ازای تمام $n \geq 4$، نامساوی $n! > 2^n$ برقرار است.

۲۸. *دنباله فیبوناچی* به صورت بازگشتی، از رابطه زیر حاصل می‌شود:

$$f(n+2) = f(n+1) + f(n), n = 1, 2, \ldots,$$

که در آن $f(2) = 1$، $f(1) = 1$. نشان دهید که

الف) $f(n) = O(2^n)$.

ب) $f(n) = \Omega(1.5^n)$.

۲۹. نشان دهید که $\sqrt{8}$ عدد گویا نیست.

۳۰. نشان دهید که $\sqrt{2}$ -2 غیرگویا است.

۳۱. نشان دهید که $\sqrt{3}$ غیر گویا است.

۳۲. درستی یا نادرستی عبارات زیر را اثبات کنید:

الف) مجموع یک عدد گویا و غیرگویا، حتماً غیرگویا است.

ب) مجموع دو عدد غیرگویای مثبت، حتماً غیرگویا است.

ج) حاصلضرب یک عدد غیرگویا و گویای غیرصفر، حتماً غیرگویا است.

۳۳. نشان دهید که هر عدد صحیح مثبتی را می‌توان به صورت حاصلضرب اعداد اول بیان کرد.

۳۴. اثبات کنید که مجموعه تمام اعداد اول، نامتناهی است.

۳۵. یک زوج اول متشکل از دو عدد اول با اختلاف دو هستند. زوجهای اول زیادی مانند ۱۱،۱۳، ۱۷،۱۹ و... وجود دارند. سه تایی‌های اول، سه عدد صحیح به صورت $n, n+2, n+4$ $n \geq 2$ می‌باشند که همگی اول هستند. نشان دهید که تنها سه تایی اول، (3,5,7) است.

سه مفهوم اساسی

زبان‌ها، گرامرها و اتوماتا سه مفهوم کلیدی در سرتاسر این کتاب هستند. در ادامه، نتایج متعددی را درباره این مفاهیم و همچنین رابطه آنها با یکدیگر کشف خواهیم کرد. اما در درجه اول باید معنی این اصطلاحات را دقیقاً بدانیم.

زبان‌ها

هرچند همه ما با زبانهای طبیعی از قبیل انگلیسی، فرانسوی ، فارسی و ترکی آشنا هستیم، شاید نتوانیم تعریف دقیقی از کلمه " زبان" ارائه دهیم. در فرهنگهای لغت، زبان معمولاً "یک سیستم مناسب برای بیان برخی ایده‌ها، واقعیت‌ها یا مفاهیم می‌باشد و متشکل از مجموعه‌ای از نشانه‌ها و قوانین برای تغییر آنها" تعریف شده است. هرچند این تعریف، ایده شهودی راجع به زبان در اختیار ما قرار می‌دهد، ولی تعریف کاملی برای آغاز مطالعه زبانهای صوری محسوب نشده و به همین دلیل، نیازمند تعریف دقیق‌تری از این لغت هستیم.

کار را با مجموعه متناهی و غیرتهی Σ از سمبل‌ها به نام الفبا آغاز می‌کنیم. با کنار هم قرار دادن همین سمبل‌ها، رشته می‌سازیم که در واقع دنباله‌های متناهی ازسمبل‌های الفبا هستند. بعنوان مثال، اگر الفبای $\Sigma=\{a,b\}$ باشد، آنگاه $abab$ و $aaabbba$ رشته‌های روی Σ هستند. به استثنای چند مورد، می‌توان از حروف کوچک $a,b,c,\ldots$ برای اعضای Σ و از $u,v,w,\ldots$ برای اسامی رشته‌ها استفاده کرد. مثلاً می‌نویسیم،

$$w = abaaa$$

و منظور ما این است که رشته‌ای به نام w دارای مقدار خاص $abaaa$ است.

الحاق دو رشته w و v، رشته‌ای است که با اتصال سمبل‌های v به گوشه سمت راست w بدست می‌آید، یعنی اگر

$$w = a_1a_2\ldots a_n$$

و

$$v = b_1b_2\ldots b_m,$$

آنگاه الحاق w و v، که با wv نمایش داده می‌شود،

$$wv = a_1a_2\ldots a_nb_1b_2\ldots b_m$$

خواهد بود. **معکوس** رشته با نوشتن سمبل‌ها در جهت عکس بدست می‌آید. بعنوان مثال، معکوس رشته w، یعنی w^R

$$w^R = a_n\ldots a_2a_1.$$

خواهد بود. طول رشته w، که بوسیله $|w|$ نمایش داده می‌شود، تعداد سمبل‌های موجود در رشته است. رشته تهی رشته‌ای است که هیچ سمبلی ندارد و بوسیله λ نمایش داده می‌شود. روابط ساده

$$|\lambda| = 0,$$

$$\lambda w = w\,\lambda = w$$

به ازای تمام wها صدق می‌کند.

هر دنبالهٔ متوالی از سمبل‌ها در w، زیررشته w خوانده می‌شوند. اگر

$$w = vu,$$

آنگاه زیررشته‌های v و u به‌ترتیب **پیشوند** و **پسوند** رشته w خوانده می‌شوند. بعنوان مثال، اگر $w=abbab$، آنگاه $\{\lambda,a,ab,abb,abba,abbab\}$ مجموعه تمام پیشوندهای w و $\{bab,ab,b\}$ برخی از پسوندهای آن هستند.

ویژگی‌های ساده رشته‌ها، از قبیل طول آنها، بسیار شهودی بوده و به همین دلیل، نیازی به توضیح ندارند. بعنوان مثال، اگر u و v رشته باشند، آنگاه طول الحاق آنها برابر با مجموع طول هر یک، یعنی

$$|uv| = |u| + |v| \qquad (۶-۱)$$

خواهد بود. علیرغم روشن بودن این رابطه، بهتر است آنرا مشخص و اثبات کنیم. اهمیت روشهای انجام این‌کار در موارد پیچیده‌تر آشکار می‌شود.

مثال ۱-۱۱

نشان دهید که (۱-۶) برای هر u و v صدق می‌کند. برای اثبات، ابتدا باید تعریفی از طول رشته در اختیار داشته باشیم. اگر w یک رشته روی Σ باشد، اینکار را به صورت بازگشتی به کمک

$$|a| = 1,$$
$$|va| = |v| + 1,$$

به ازای تمام $a\in\Sigma$ انجام می‌دهیم. این تعریف بیان صوری از درک شهودی ما در مورد طول رشته است. بنابراین، طول هر سمبل یک است و در نتیجه با افزودن سمبل جدید، طول رشته هم به اندازه یک واحد افزایش خواهد یافت. با داشتن این تعریف صوری، به راحتی می‌توان (۱-۶) را به کمک استقراء اثبات کرد.

براساس تعریف، (۱-۶) به ازای تمام uها با هر طولی و تمام vها با طول یک برقرار است؛ بنابراین پایه استقراء را تعریف کردیم. مشابه تمام فرض‌های استقراء، (۱-۶) به ازای تمام uها با هر طولی و تمام vها با طول $1,2,\ldots,n$ صدق می‌کند. حال یکی از vها با طول $n+1$ را درنظرگرفته و آنرا به صورت $v = wa$ می‌نویسیم. پس،

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

براساس فرض استقراء (که چون w دارای طول n است، قابل اعمال می‌باشد)،

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

بطوریکه

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

بنابراین، (۱-۶) به ازای تمام u و vها با طول حداکثر $n+1$ برقرار است و در نتیجه استقراء و همچنین استدلال به پایان می‌رسد.

■

اگر w یک رشته باشد، آنگاه w^n رشته‌ای است که از n مرتبه تکرار w بدست می‌آید. بعنوان یک نمونه خاص،

$$w^0 = \lambda,$$

را برای تمام wها تعریف می‌کنیم.

چنانچه Σ یک الفبا باشد، آنگاه از Σ^* برای نمایش مجموعهٔ رشته‌های بدست آمده از الحاق صفر یا بیشتر سمبل دیگر Σ استفاده می‌کنیم. مجموعه Σ^* همواره حاوی λ خواهد بود. برای حذف رشته تهی،

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

را تعریف می‌کنیم. هرچند Σ طبق فرض متناهی است، اما ازآنجایی‌که هیچ محدودیتی در مورد طول رشته در این مجموعه‌ها وجود ندارد، Σ^* و Σ^+ همواره نامتناهی هستند. یک زبان در اغلب موارد بعنوان زیرمجموعه‌ای از Σ^* تعریف می‌شود. هر رشته در زبان L جمله‌ای از L خوانده می‌شود. به دلیل گستردگی این تعریف، می‌توان هر مجموعه‌ای از رشته‌های روی یک الفبای Σ را یک زبان تلقی کرد. در ادامه، به روشهای تعریف و توصیف برخی زبانها اشاره می‌کنیم تا این مفهوم بسیار گسترده تا حدودی سازماندهی شود. اما در حال حاضر به ذکر چند مثال خاص اکتفا می‌کنیم.

مثال ۱-۹

فرض کنیم $\Sigma = \{a,b\}$ باشد آنگاه

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

و مجموعه

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

یک زبان روی Σ است. به دلیل متناهی بودن تعداد جملات این زبان، ما آنرا یک زبان متناهی می‌نامیم. مجموعه

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

هم یک زبان روی Σ است. رشته‌های $aabb$ و $aaaabbbb$ در زبان L هستند، اما دنباله abb در این زبان وجود ندارد. زبان L نامتناهی است. اغلب زبان‌های جالب نامتناهی هستند.

■

از آنجایی‌که زبان‌ها مجموعه هستند، اجتماع، اشتراک و تفاضل دو زبان به راحتی قابل تعریف است. مکمل یک زبان با توجه به Σ^* تعریف می‌شود. یعنی مکمل L

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

می‌باشد. معکوس یک زبان، مجموعه تمام رشته‌های معکوس شده است، یعنی

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

الحاق دو زبان L_1 و L_2 مجموعه تمام رشته‌های بدست آمده از طریق الحاق هریک از اعضاء L_1 به اعضاء L_2 است؛ خصوصاً آنکه،

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

L^n را یک زبان L تعریف می‌کنیم که n مرتبه به خود الحاق شده باشد. همچنین به ازای تمام زبان‌های L، موارد خاص

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

و

$$L^1 = L$$

را در نظر می‌گیریم.

در نهایت، بستار ستاره‌ای یک زبان را به صورت

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

و بستار مثبت را به صورت

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

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 داشته باشید که در این مثال، n و m ارتباطی به هم ندارند و رشته $aabbaaabbb$ در L^2 موجود است.

معکوس L بسادگی بصورت زیر توصیف می‌شود:

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

اما توصیف L^* و $\overline{L}$ به این صورت بسیار مشکل خواهد بود. با بررسی چند نمونه کوچک به راحتی متوجه محدودیت مجموعه سمبل‌ها در مورد ویژگی زبانهای پیچیده خواهید شد.

■

گرامرها

برای مطالعه زبان‌ها از نظر ریاضی، نیازمند وجود مکانیزمی برای تشریح آنها هستیم. زبان محاوره‌ای مبهم و غیردقیق است و به همین دلیل توصیف‌های نادقیق معمولاً غیرکافی هستند. مجموعه نشانه‌ها و سمبل‌ها در مثال ۱-۹ و ۱-۱۰ هرچند مناسب، ولی غیرکافی هستند. در ادامه، مطالب بیشتری راجع به

مکانیزم‌های متعدد تعریف زبان خواهیم آموخت که در شرایط مختلف مفید و کارآمد هستند. در اینجا یکی از معروف‌ترین و البته قدرتمندترین آنها، یعنی گرامر، را بررسی می‌کنیم.

بوسیله دستورات هر زبانی می‌توان در مورد درستی یا نادرستی یک جمله قضاوت کرد. بعنوان مثال، دستور زبان فارسی به ما می‌گوید که "جمله می‌تواند شامل یک مسند و گزاره باشد". برای تفسیر دقیقتر این دستور می‌نویسیم که

$$<\text{جمله}> \longrightarrow <\text{مسند}> <\text{گزاره}>,$$

البته، برای بحث در مورد جملات واقعی یک زبان باید دقیق‌تر بررسی کنیم و تعاریفی برای ساختارهای جدید <گزاره> و <مسند> ارائه دهیم. اگر به این ترتیب عمل کنیم که

$$<\text{مسند}> \longrightarrow <\text{ضمیر اشاره}> <\text{اسم}>,$$

$$<\text{گزاره}> \longrightarrow <\text{فعل}>,$$

و کلمات واقعی "آن" و "این" را با <ضمیر اشاره> و "سگ" و "پسر" را با <اسم> و "می‌دود" و "راه می‌رود" را با <فعل> مرتبط کنیم، آنگاه براساس دستور زبان فارسی جملاتی نظیر آن پسر می‌دود و آن سگ راه می‌رود جملات درستی خواهند بود. برای ارائه یک دستور زبان کامل، باید به صورت نظری، هرجمله‌ای را به همین صورت تعریف کرد.

این مثال نمونه بارزی از تعریف یک مفهوم کلی در قالب مفاهیم ساده‌تر است. بحث را با بالاترین مفهوم یعنی <جمله> آغاز کرده و دائماً آن را به بلوک‌های سازنده زبانی کوچکتر تبدیل می‌کنیم. تعمیم این نظریات، ما را به سمت گرامرهای زبان‌های صوری هدایت می‌کند.

تعریف ۱-۱

گرامر G بوسیله چهارتایی

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

تعریف می‌شود بطوریکه در آن، V مجموعه متناهی از اشیاء به نام متغیرها،

T مجموعه متناهی از اشیاء به نام سمبل‌های پایانی،

$S\in V$ سمبل ویژه‌ای به نام متغیر شروع

و P مجموعه متناهی از قوانین است.

ناگفته پیداست که باید مجموعه‌های V و T را غیرتهی و جدا از هم فرض کرد.

قوانین، هسته اصلی مبحث گرامر را تشکیل می‌دهد؛ این قوانین، نحوه تبدیل رشته‌ها به یکدیگر را تشریح کرده و به این ترتیب، زبان مربوط با آن گرامر را تعریف می‌کنند. در بحث خود فرض می‌کنیم که تمام قوانین مربوط به فرم

$$x\to y,$$

هستند که در آن، x عضو $(V\cup Y)^{*}$ و y عضو $(V\cup T)^{*}$ می‌باشد. قوانین به این صورت اعمال می‌شوند:

با در اختیار داشتن رشته w به فرم

$$w=uxv,$$

می‌گوییم که قانون $x\to y$ در این رشته قابل استفاده است و می‌توان از آن برای جایگزینی x با y استفاده کرد. در نتیجه رشته جدیدی به فرم

$$z=uyv$$

بدست خواهد آمد که این جایگزین و تبدیل به صورت

$$w\Rightarrow z$$

نوشته شده و می‌خوانیم که w ، z را اشتقاق می‌کند یا اینکه z از w اشتقاق شده است. رشته‌های بعدی با اعمال قوانین گرامر با ترتیب اختیاری اشتقاق می‌شوند. یک قانون تولید می‌تواند هر جایی که قابل اعمال باشد، استفاده شود، و می‌تواند هر جایی که مطلوب باشد بکار رود. اگر

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

می‌گوییم که w_1 ، w_n را مشتق می‌کند و می‌نویسیم

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

به این معناست که می‌توان برای مشتق کردن w_n از w_1 از هر تعداد تکرار لازم (حتی صفر بار) استفاده کرد.

با استفاده از گرامرها می توان بوسیله بکاربردن قوانین با ترتیب‌های مختلف، رشته‌های متعددی تولید کرد. مجموعه این رشته‌های پایانی، زبانی است که بوسیله گرامر تولید می‌شود.

تعریف ۱-۲

فرض کنیم که $G=(V,T,S,P)$ یک گرامر باشد. آنگاه مجموعه

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

زبانی است که توسط گرامر G تولید می‌شود.

اگر $w\in L(G)$، آنگاه دنباله

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

اشتقاقی از جمله w است. رشته‌های S ، w_1 ، w_2 ، ...و w_n که حاوی پایانه‌ها و متغیرها هستند، فرم-

های جمله‌ای از اشتقاق نامیده می‌شود.

مثال ۱-۱۱

گرامر زیر را در نظر بگیرید

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

که در آن P از

$$S\to aSb,$$

$$S\to\lambda.$$

بدست می‌آید. پس

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

بنابراین می‌توانیم بنویسیم که $S\Rightarrow aabb$.

رشته $aabb$ در زبان تولید شده بوسیله G یک جمله، ولی $aaSbb$ یک فرم جمله است.

هرچند گرامر G بطور کامل $L(G)$ را تعریف می‌کند، اما شاید نتوان به راحتی شرح کاملاً واضحی از زبان را، از گرامر آن بدست آورد. با این وجود، در اینجا به راحتی می‌توان عبارت

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

را ارائه و آنرا اثبات کرد. با توجه به بازگشتی بودن قانون $S\to aSb$ ، اثبات بوسیله استقراء مطرح می‌شود. ابتدا نشان می‌دهیم که همه فرم‌های جمله‌ای باید به صورت

$$w_i=a^iSb^i \qquad (۱-۷)$$

باشند. فرض کنید که (۱-۷) برای هرفرم جمله‌ای w_i با طول $2i+1$ یا کمتر برقرار باشد. حال می‌توانیم برای بدست آوردن فرم جمله‌ای دیگر (که جمله نباشد)، قانون $S\to aSb$ را اعمال کنیم. بنابراین عبارت زیر بدست می‌آید:

$$a^iSb^i\Rightarrow a^{i+1}Sb^{i+1},$$

بطوریکه تمام فرم‌های جمله‌ای با طول $2i+3$ هم به فرم (۱-۷) هستند. از آنجایی‌که (۱-۷) برای $i=1$ درست است، براساس استقراء، برای تمام i ها هم برقرار می‌باشد. در نهایت، برای بدست آوردن جمله باید قانون $S\to\lambda$ را اعمال می‌کنیم که در اینصورت،

$$S\Rightarrow a^nSb^n\Rightarrow a^nb^n$$

نماینده تمام اشتقاق‌های ممکن می‌باشد. بنابراین، G فقط قادر به تولید رشته‌هایی به فرم a^nb^n خواهد بود.

همچنین باید نشان دهیم که تمام رشته‌های به این فرم، قابل اشتقاق هستند. این‌کار به راحتی و از

طریق استفاده از $S\to aSb$ به هر تعداد لازم و سرانجام با بکارگیری قانون $S\to\lambda$ قابل انجام است.

■

مثال ۱-۱۲

گرامری برای تولید زبان زیر بنویسید:

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

از ایده مثال قبل می‌توان در این مورد هم استفاده کرد. فقط لازم است که به‌کمک قانون $S\to Ab$ یک b اضافه تولید کنیم. مابقی قوانین باید به صورتی انتخاب شوند که A بتواند زبان مثال قبل را تولید کند. براساس همین استدلال می‌توان گرامر $G=(\{S,A\},\{a,b\},S,P)$ را با قوانین

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

بدست آورد.

سعی کنید با اشتقاق چند جمله، از عملکرد درست این گرامر ساخته شده، مطمئن شوید.

■

چون مثال‌های بالا نمونه‌های تقریباً ساده‌ای هستند، مطرح کردن استدلال‌های خشک و جدی ممکن است غیرلازم به نظر برسد. اما در اغلب موارد به راحتی نمی‌توان به طور دقیق گرامری را برای زبان تعریف شده ارائه کرده و یا ویژگی شهودی از زبان تعریف شده بوسیله گرامر را بیان نمود. برای اثبات اینکه زبان مفروض واقعاً بوسیله گرامر خاص G تولید شده است، باید نشان دهیم که

اولاً) می‌توان با استفاده از G ، هر $w\in L$ را از S اشتقاق نمود و

ثانیاً) هر رشته‌ای که به این ترتیب اشتقاق شود، در L موجود است.

مثال ۱-۱۳

$\Sigma=\{a,b\}$ را درنظرگرفته و $n_a(w)$ و $n_b(w)$ به ترتیب بیانگر تعداد aها و bها در رشته w هستند. بنابراین گرامر G با قوانین

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

زبان

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

را تولید می‌کند. به دلیل مشخص نبودن این ادعا، باید استدلال‌های قانع‌کننده‌ای را برای اثبات آن ارائه دهیم.

اولاً، چون تمام قوانینی که a تولید می‌کنند، یعنی $S \to aSb$ و $S \to bSa$، به صورت همزمان یک b هم تولید می‌کنند، مشخص است که فرم جمله‌ای از G دارای تعداد برابری a و b خواهد بود. بنابراین، تمام اعضاء $L(G)$ در L موجود هستند. همچنین می‌توان مشاهده کرد که تمام رشته‌های عضو L بوسیله G قابل اشتقاق هستند.

کار را با نگاه کلی به مسأله آغاز کرده و فرم‌های مختلف $w \in L$ را در نظر می‌گیریم. با این فرض که w از a آغاز و به b ختم می‌شود، بنابراین به فرم

$$w = av_1b,$$

خواهد بود که در آن، w_1 هم عضو L است. چنانچه S واقعاً تمام رشته‌های داخل L را اشتقاق کند، در اینصورت S می‌تواند با

$$S \Rightarrow aSb$$

شروع شود. در صورتی که w با b آغاز و به a ختم شود، این استدلال بازهم درست خواهد بود. اما از آنجایی‌که رشته عضو L می‌تواند با یک سمبل آغاز و به همان ختم شود، تمام موارد را در خود ندارد. رشته‌ای به این صورت، مثلاً $aabbba$ ، را می‌توان حاصل الحاق دو رشته کوتاهتر $aabb$ و ba در نظر گرفت که هر دو در L موجود باشند. آیا این تعمیم همیشه و همواره صحیح است؟ برای اثبات درستی می‌توان به این صورت استدلال کرد :

فرض کنید که با شروع از انتهای سمت چپ رشته، ۱+ را برای یک a و ۱- را برای یک b بشماریم. چنانچه دنباله مفروض w از a آغاز و به آن ختم شود، آنگاه پس از آخرین سمبل سمت چپ، عدد ۱+ و پیش از آخرین سمبل در سمت راست، ۱- را خواهیم داشت. بنابراین، در جایی در میانه دنباله به صفر خواهیم رسید؛ به این معنی که این دست رشته‌ها باید به فرم

$$w = w_1w_2,$$

باشند که در آن، w_1 و w_2 در L موجود هستند. این امر توسط قانون $S \to SS$ محقق می‌شود.

پس از ارائه شهودی استدلال، می‌توانیم بحث را به صورت دقیق‌تری دنبال کنیم. مجدداً با استفاده از استقراء، فرض می‌کنیم که همهٔ رشته‌های $w \in L$ با $|w| \le 2n$ بوسیله G قابل اشتقاق باشند. یک $w \in L$ با طول $2n+2$ را در نظر می‌گیریم. اگر $w = av_1b$، آنگاه w_1 عضو L بوده و $|w_1| = 2n$. بنابراین، براساس فرض

$$S \Rightarrow^* w_1.$$

آنگاه

$$S \Rightarrow aSb \Rightarrow^* av_1b \Rightarrow^* w$$

ممکن بوده و w از G قابل اشتقاق خواهد بود. در صورتیکه $w = bv_1a$، می‌توان استدلال‌های مشابه همین را مطرح کرد.

در غیر اینصورت، یعنی چنانچه w با یک سمبل آغاز و خاتمه پیدا کند، براساس استدلال شمارش می‌توان نتیجه گرفت که به فرم $w = w_1w_2$ بوده، w_1 و w_2 هر دو عضو L هستند و طولی کمتر یا مساوی با $2n$ خواهند داشت. بنابراین مجدداً مشاهده می‌کنیم که

$$S \Rightarrow^* SS \Rightarrow^* w_1S \Rightarrow^* w_1w_2 = w$$

امکان‌پذیر است. از آنجایی‌که فرض استقراء در مورد $n=l$ کاملاً برقرار است، بنابراین پایه استقراء نیز برقرار است. لذا این ادعا به ازای تمام n ها برقرار است. به این ترتیب، استدلال به پایان می‌رسد. ■

معمولاً هر زبان با چندین گرامر معرفی و بوسیله آنها تولید می‌شود. حتی در صورت تفاوت ظاهری این گرامرها، آنها بازهم از بعضی جهات هم ارز خواهند بود. گرامرهای G_1 و G_2 را در صورتی هم ارز می‌گوییم که یک زبان یکسانی را تولید کنند، یعنی

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

همانطور که بعداً خواهیم دید، اثبات هم ارزی گرامرها همیشه چندان هم آسان نیست.

مثال ۱-۱۴

گرامر $G_1 = (\{A,S\},\{a,b\},S,P_1)$ را در نظر بگیرید که در آن P_1 شامل قوانین

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

است. در اینجا یک قرارداد ساده برای نوشتن گرامرها معرفی می‌کنیم که در آن، چندین تولید متفاوت، که طرفین چپ یکسانی داشته باشند، روی یک خط نوشته شده و طرفین راست بوسیله | از هم تفکیک می‌شوند. بنابراین، به‌وسیله این نماد، $S \to aAb \mid \lambda$ به معنای دو قانون $S \to aAb$ و $S \to \lambda$ خواهد بود.

این گرامر هم ارز با گرامر G در مثال ۱-۱۱ است. برای اثبات هم ارزی فقط لازم است نشان دهیم که

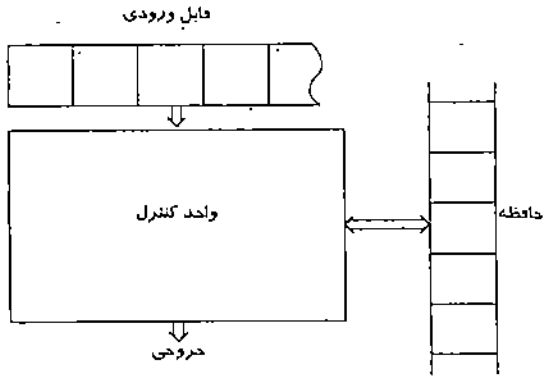
$$L(G_1) = \{a^nb^n : n \ge 0\}.$$

این کار را خودتان بعنوان تمرین انجام دهید.

■

اتوماتا

یک اتومات در واقع مدل انتزاعی از یک کامپیوتر رقمی است. به همین دلیل، هراتومات دارای برخی از ویژگی‌های اساسی کامپیوترها، از جمله مکانیزمی برای خواندن ورودی است. می‌توان ورودی را رشته‌ای برروی یک الفبای مفروض در نظر گرفت که در یک **فایل ورودی** نوشته شده است. اتوماتا قادر به خواندن این فایل‌ها هستند، ولی نمی‌توانند هیچ تغییری در آن انجام دهند. فایل ورودی به چند سلول تقسیم می‌شود و هر سلول قادر به نگهداری یک سمبل از ورودی خواهد بود. هد خواندن می‌تواند فایل ورودی را، با در نظر گرفتن یک سمبل در هر لحظه، از چپ به راست بخواند. همچنین قادر است، به محض حس کردن شرط پایان فایل، انتهای رشته ورودی را تشخیص دهد. اتوماتا می‌توانند خروجی را به شکل خاصی تولید کنند. همچنین، در برخی موارد دارای یک **حافظه** متشکل از تعداد نامحدودی سلول هستند. هریک از این سلول‌ها یک سمبل الفبا (و نه لزوماً همان سمبل در الفبای خروجی) را نگهداری می‌کنند. اتومات قادر به خواندن و ذخیره‌سازی محتوای سلول‌های حافظه است. همچنین، **واحد کنترل** اتومات قادر است در یکی از **حالت‌های داخلی** ماشین قرار داشته و می‌تواند بر طبق تعریفی که برای آن داریم، حالت را تغییر دهد. شکل ۱-۴ شمای کلی از یک اتومات را ارائه می‌دهد.



شکل ۱-۴

براساس فرض، اتومات در چارچوب زمانی گسسته‌ای کار می‌کند. در هر واحد زمانی (یالس ساعت)، واحد کنترل در حالت داخلی خاصی قرار می‌گیرد و هد خواندن، سمبل خاصی را از فایل ورودی می‌خواند. حالت داخلی واحد کنترل در مرحله زمانی بعدی، توسط **حالت بعدی** یا **تابع انتقال** تعیین می‌شود. این تابع انتقال، حالت بعدی را برحسب حالت جاری، سمبل ورودی جاری و اطلاعاتی که اکنون در حافظه موقت موجود است، ارائه می‌دهد. درحین انتقال از یک برش زمانی به برش زمانی بعدی، ممکن است خروجی تولید شده و یا اطلاعات موجود در حافظه موقت، تغییر داده شود. از

اصطلاح **پیکربندی** برای اشاره به حالت خاصی از واحد کنترل، فایل ورودی و محتوای حافظه موقت، هر سه با هم استفاده می‌شود. انتقال اتومات از یک پیکربندی به پیکربندی دیگر **حرکت** خوانده می‌شود.

این مدل عمومی همهٔ اتوماتاهای مورد بحث در این کتاب را پوشش می‌دهد. یک کنترل حالت متناهی برای همهٔ موارد خاص وجود دارد. اما روش تولید خروجی و ماهیت حافظه موقت، باهم متفاوت هستند. همانطور که بعداً خواهیم دید، ماهیت حافظه موقت، بر قدرت اتوماتا تأثیرگذار است.

در مباحث بعدی با دو اصطلاح **اتوماتای معین** و **اتوماتای نامعین** برخورد خواهیم کرد. اتومات معین اتوماتی است که در آن، هر حرکت منحصراً بوسیله پیکربندی فعلی تعیین می‌شود. با اطلاع از حالت داخلی، ورودی و محتوای حافظه موقت می‌توان رفتار بعدی اتومات را دقیقاً تعیین کرد. اما در اتوماتای نامعین این چنین نیست. یک اتومات نامعین می‌تواند در هر نقطه، حرکات احتمالی متعددی داشته باشد. بنابراین فقط مجموعه حرکات احتمالی، قابل پیش‌بینی خواهد بود. رابطه بین انواع مختلف اتوماتای معین و نامعین یکی از مباحث مهم در این کتاب است.

اتوماتی که پاسخ خروجی آن فقط منحصر به یک "بله" یا "خیر" باشد، اصطلاحاً **پذیرنده** نامیده می‌شود. پذیرنده پس از قرارگرفتن در معرض رشته ورودی، یا آنرا می‌پذیرد و یا رد می‌کند. اتوماتای کلی‌تر دیگر، که قادر به تولید رشته‌هایی ازسمبل‌ها بعنوان خروجی هستند، **مترجم** خوانده می‌شوند. در این کتاب بیشتر روی پذیرنده‌ها تمرکز می‌کنیم؛ با این وجود، در بخش بعد نمونه‌های ساده‌ای از مترجم‌ها را نیز ذکر خواهیم کرد.

تمرین‌ها

۱. با استفاده از استقراء روی n نشان دهید که به ازای تمام رشته‌های u و همه n ها، $|u^n| = n|u|$.

۲. معکوس یک رشته را، که بصورت غیررسمی در بالا معرفی شد، می‌توان بطور دقیق‌تر بوسیله قوانین بازگشتی

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

به ازای تمام $a \in \Sigma$، $w \in \Sigma^*$ تعریف کرد. اثبات کنید که برای همه $u, v \in \Sigma^*$

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

۳. اثبات کنید که برای تمام $w \in \Sigma^*$ ، $(w^R)^R = w$.

۴. فرض کنید $L = \{ab, aa, baa\}$ باشد، کدامیک از رشته‌های زیر در L^* و کدامیک در L^4 موجود هستند؟

$abaabaaabaa, aaaabaaaa, baaaaabaaaab, baaaaabaa$

۱۰. فرض کنیم که $\Sigma=\{a,b\}$ و $L=\{aa,bb\}$. با استفاده از توصیف صوری مجموعه‌ها، $\bar{L}$ را توصیف کنید. ⊖

فرض کنیم که L زبانی روی یک الفبای غیرتهی باشد. نشان دهید که L و $\bar{L}$ نمی‌توانند هر دو متناهی باشند.

آیا زبانی وجود دارد که در آن $\overline{L^*}=(\bar{L})^*$ ؟

اثبات کنید که برای همه زبان‌های L_1 و L_2،

$$(L_1L_2)^R=L_2^RL_1^R.$$

۹. نشان دهید که برای هر زبانی، $(L^*)^*=L^*$.

۱۰. درستی یا نادرستی ادعاهای زیر را نشان دهید.

الف) به ازای هر زبان L_1 و L_2، $(L_1\cup L_2)^*=L_1^*\cup L_2^*$

ب) به ازای هر زبان L، $(L^R)^*=(L^*)^R$

۱۱. گرامرهایی را برای $\Sigma=\{a,b\}$ پیدا کنید که مجموعه‌های زیر را تولید کند.

الف) همه رشته‌های دارای دقیقاً یک a.

ب) همه رشته‌های دارای حداقل یک a.

ج) همه رشته‌های دارای حداکثر سه a.

د) همه رشته‌های دارای حداقل سه a. ⊖

در این موارد، با ارائه استدلال‌های قانع‌کننده، نشان دهید که گرامر پیشنهادی شما واقعاً زبان مورد نظر را تولید می‌کند.

۱۲. بطور مختصر، زبان تولید شده بوسیله گرامر با قوانین زیر را توصیف کنید.

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

گرامری با قوانین زیر چه زبانی را تولید می‌کند؟

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

۱۳. فرض کنید $\Sigma=\{a,b\}$. گرامر تولیدکننده زبان‌های زیر را بنویسید:

الف) $L_1=\{a^nb^m:n\geq 0,m>n\}$. ⊖

ب) $L_2=\{a^nb^{2n}:n\geq 0\}$.

ج) $L_3=\{a^{n+2}b^n:n\geq 1\}$.

د) $L_4=\{a^nb^{n-1}:n\geq 3\}$. ⊖

ه) L_1L_2.

و) $L_1\cup L_2$.

ز) L_1^2.

ح) L_1^*.

ط) $L_1-\bar{L}_4$.

۱۵. برای هریک از زبان‌های زیر گرامری روی $\Sigma=\{a\}$ بنویسید.

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

۱۶. گرامری را بنویسید که زبان

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

را تولید کند. جواب خود را دقیق توجیه کنید.

۱۷. بطورمختصر، زبان تولید شده بوسیله

$$S\to aSb\,|\,bSa\,|\,a$$

را شرح دهید.

۱۸. با استفاده از مفاهیم تعریفی مثال ۱۳-۱، گرامرهایی را برای زبان‌های زیر پیدا کنید. فرض کنید که $\Sigma=\{a,b\}$.

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

۱۹. تمرین قبل را با $\Sigma=\{a,b,c\}$ حل کنید.

۲۰. استدلال‌های مثال ۱-۱۴ را تکمیل‌کرده و نشان دهید که $L(G_1)$ واقعاً زبان مورد نظر را تولید شده می باشد.

۲۱. آیا دو گرامر با قوانین

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

و

$$S\to aAb\,|\,ab,$$
$$A\to aAb\,|\,\lambda.$$

هم‌ارز هستند؟ فرض کنید که در هر دو گرامر، S سمبل شروع باشد.

۲۲. نشان دهید که گرامر $G=(\{S\},\{a,b\},S,P)$ با قوانین

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

با گرامر مثال ۱-۱۳ هم‌ارز است.

۲۳. نشان دهید که گرامرهای

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

و

$$S \to aSb \mid bSa \mid a$$

هم ارز نیستند.

۱-۳ برخی از کاربردها

هرچند تا به اینجا روی ماهیت انتزاعی و ریاضی زبانهای صوری و اتوماتا تأکید کرده‌ایم، این‌گونه برداشت می‌شود که این مفاهیم کاربردهای گسترده‌ای در علوم کامپیوتر پیدا کرده و در واقع فصل مشترک بسیاری از حوزه‌های تخصصی محسوب می‌شوند. در این بخش، با ارائه چند مثال ساده، خواننده به عینه مشاهده خواهد کرد که مطالب ارائه شده در این کتاب صرفاً مجموعه‌ای از انتزاع‌ها نیستند، بلکه در واقع ابزاری برای درک بهتر بسیاری از مسائل مهم و واقعی محسوب می‌شوند.

گرامرها و زبان‌های صوری کاربرد گسترده‌ای در زبان‌های برنامه‌سازی پیدا کرده اند. برنامه‌سازها غالباً در برنامه‌سازی‌های خود با درک کم و بیش شهودی از زبان برنامه، مواجه می‌شوند. با این وجود، ممکن است در برخی موارد برای استفاده از یک ویژگی نامتعارف نیاز به استفاده از توصیف‌های دقیقی از قبیل نمودارهای نحوی پیدا کنیم که در اغلب محیط‌های برنامه‌سازی به چشم می‌خورد. بعنوان مثال، در تمام مراحل کار با یک کامپایلر خاص و یا برای بررسی درستی یک نمونه برنامه، باید توصیف دقیقی از زبان در اختیار داشته باشیم. از بین تمام روشهای موجود، برای تعریف دقیق زبان‌های برنامه‌سازی، گرامرها را می‌توان از بقیه پرکاربردتر دانست.

گرامرهای توصیف‌کننده برخی زبان‌ها از قبیل پاسکال یا C بسیار گسترده هستند. برای نمونه، زبان کوچکتری را در نظر می‌گیریم که بخشی از یک زبان بزرگتر محسوب می‌شود.

مثال ۱-۱۵

قوانین مربوط به شناسه‌های متغیر در C به صورت زیر است:

۱. یک شناسه دنباله‌ای از حروف، ارقام و زیرواژه‌هاست.
۲. یک شناسه باید با یک حرف یا یک زیرواژه آغاز شود.
۳. درشناسه‌ها استفاده از حروف بزرگ و کوچک مجاز است.

این قوانین را می‌توان با تعریفی صوری بوسیله گرامر زیر توصیف کرد:

$\langle id\rangle \to \langle letter\rangle\langle rest\rangle \mid \langle undrscr\rangle\langle rest\rangle$

$\langle rest\rangle \to \langle letter\rangle\langle rest\rangle \mid \langle digit\rangle\langle rest\rangle \mid \langle undrscr\rangle\langle rest\rangle \mid \lambda$

$\langle letter\rangle \to a \mid b \mid \ldots \mid z \mid A \mid B \mid \ldots \mid Z$

$\langle digit\rangle \to 0 \mid 1 \mid \ldots \mid 9$

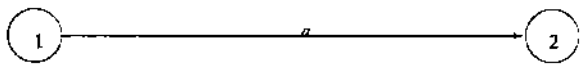
$\langle undrscr\rangle \to \_$

در این گرامر، $\langle id\rangle$، $\langle letter\rangle$، $\langle digit\rangle$، $\langle undrscr\rangle$ و $\langle rest\rangle$ متغیر بوده و، حروف، ارقام و زیرواژه‌ها الفباء هستند. یک اشتقاق شناسه a0 عبارت است از:

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

هر چند تعریف زبان‌های برنامه‌سازی به روش‌های مختلف قابل انجام است، تعریف آنها بوسیله گرامرها یکی از روش‌های متعارف و بسیار مفید محسوب می‌شود. بعنوان نمونه، می‌توان برای توصیف یک زبان مفروض، تعدادی از رشته‌های پذیرفته شده آنرا بعنوان بخشی از زبان در نظر گرفت. برای بحث دقیق‌تر در این مورد باید تعریف دقیق‌تری از اتوماتا در اختیار داشته باشیم. در ادامهٔ درس، این‌کار را انجام می‌دهیم. اما فعلاً اجازه دهید بحث را به صورت شهودی‌تر ادامه دهیم.

یک اتومات را می‌توان بوسیله گرافی نمایش داد که رئوس آن بیانگر حالت‌های داخلی و یال‌ها بیانگر انتقالات باشند. برچسب‌های روی یال‌ها بیانگر رویدادهایی است که در حین انتقال (در ورودی و خروجی) رخ می‌دهد. شکل ۱-۵ انتقال از وضعیت ۱ به وضعیت ۲ را نمایش می‌دهد و در حالی ترسیم شده که سمبل ورودی یک a بوده است. در ادامه روش دیگری را برای توصیف شناسه‌های C بررسی کنیم.

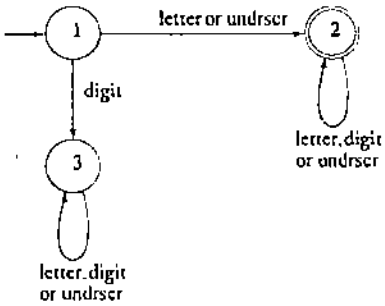


شکل ۱-۵

مثال ۱-۱۶

اتومات شکل ۱-۶ تمام شناسه‌های مجاز C که تنها در آنها رقم و حرف وجود دارد را می‌پذیرد. چند حالت را بررسی می‌کنیم:

تصور می‌کنیم که اتومات ابتدا در وضعیت ۱ قرار گیرد؛ اینکار را با ترسیم یک فلش (که از هیچ رأسی آغاز نشده است) به سمت این حالت انجام می‌دهیم. مطابق معمول، رشته مورد بررسی از چپ به راست و یک سمبل در هر مرحله خوانده می‌شود. چنانچه اولین سمبل یک حرف یا خطزیر باشد، اتومات به وضعیت ۲ رفته و پس از آن بقیهٔ رشته دیگر اهمیتی ندارد. بنابراین وضعیت ۲ بیانگر وضعیت "بله" پذیرنده است. بالعکس، اگر سمبل اول یک رقم باشد، اتومات به وضعیت ۳، یعنی وضعیت "نه"، وارد شده و در همان جا باقی می‌ماند. در این راه‌حل فرض می‌کنیم که هیچ سمبل ورودی غیر از حروف، ارقام و یا خطزیر امکان پذیر نیستند.



شکل ۱-۶

کامپایلرها و دیگر مترجم‌ها، که برنامه را از زبانی به زبان دیگر تبدیل می‌کنند، بطور گسترده‌ای از نظریات مطرح شده در این مثال‌ها استفاده می‌کنند. زبان‌های برنامه‌سازی را می‌توان، مطابق مثال ۱-۱۵، دقیقاً از طریق گرامرها تعریف کرد. همچنین، گرامرها و اتوماتا نقش اساسی در فرآیندهای تصمیم‌گیری بازی می‌کنند که در آنها، از یک قطعه کد خاص برای برآورده‌کردن شرایط زبان برنامه‌سازی استفاده می‌شود. مثال بالا نمونه خوبی از نحوه انجام اینکار است؛ مثال‌های بعدی این مشاهده را گسترش می‌دهند.

حوزه کاربردی مهم دیگر، طراحی مدارهای دیجیتالی است که در آن عمدتاً از پذیرنده‌های مترجم استفاده می‌شود. هرچند قصد نداریم موضوع را در اینجا بطور گسترده مورد بررسی قرار دهیم، نمونه ساده‌ای از آن را ذکر خواهیم کرد. با وجودی‌که هر کامپیوتر دیجیتالی را می‌توان در کل بعنوان یک اتومات در نظر گرفت، اینکار لزوماً و همیشه مفید نیست. اگر ثبات‌های داخلی و حافظه اصلی یک کامپیوتر را بعنوان واحد کنترل اتومات در نظر بگیریم، اتومات در مجموع "2 حالت داخلی خواهد داشت (n مجموع تعداد همهٔ بیت‌ها در ثبات‌ها و حافظه است). حتی در صورت کوچک بودن n هم، مقدار "2 چنان عدد بزرگی خواهد بود که نمی‌توان با آن کار کرد. اما در واحدهای بسیار کوچکتر، نظریه اتوماتا بعنوان یک ابزار طراحی مفید قابل استفاده است.

مثال ۱-۱۱

جمع‌کننده‌های دودویی، جزء سازنده همه کامپیوترهای همه‌منظوره هستند. این جمع‌کننده‌ها با گرفتن دو رشتهٔ بیتی بعنوان نماینده اعداد، حاصل‌جمع آنها را در خروجی تولید می‌کنند. برای درک بهتر موضوع، فرض کنید که فقط با اعداد صحیح مثبت سروکار داریم و با فرض این‌که:

$$x = a_0 a_1 \ldots a_n$$

به معنی عدد صحیح

$$V(x) = \sum_{i=0}^{n} a_i 2^i$$

است.

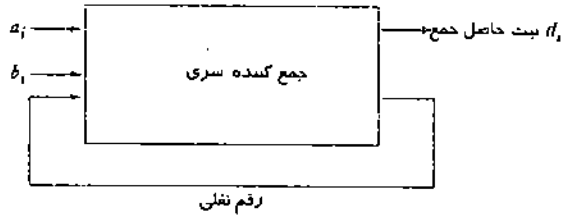
جمع‌کننده‌های سری با شروع از انتهای سمت چپ، دو عدد مفروض $x = a_0 a_1 \ldots a_n$ و $y = b_0 b_1 \ldots b_n$ را بیت به بیت جمع می‌کند. جمع هر بیت، یک رقم برای نتیجه‌جمع و همچنین یک رقم نقلی برای رقم پرارزش بعدی تولید می‌نماید. جدول جمع دودویی (شکل ۱-۷) این فرآیند را بطور خلاصه نمایش می‌دهد.

a_i \ b_i	0	1
0	0 بدون نقلی	1 بدون نقلی
1	1 بدون نقلی	0 نقلی

شکل ۱-۷

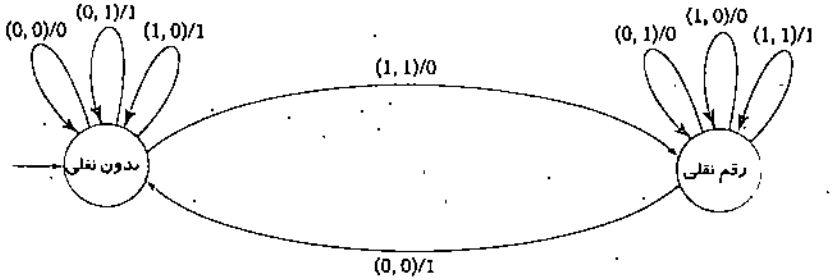
در شکل ۱-۸ یک نمونه از نمودار بلوکی که برروی کامپیوترها انجام شده، ارائه شده است. براساس این شکل، جمع‌کننده، بسته‌ای است که پس از پذیرفتن دو بیت، بیت حاصل‌جمع آنها و احتمالاً یک بیت نقلی را تولید می‌کند. این شکل هرچند وظیفه یک جمع‌کننده را شرح می‌دهد، اما توضیح چندانی راجع به فعالیت‌های داخلی آن ارائه نمی‌کند. درحالی‌که اتوماتا (که امروزه به جای آن از مترجم استفاده می‌شود) اینکار را بسیار واضح‌تر انجام می‌دهند.

ورودی‌های مترجم به صورت زوج‌های بیت (a_i, b_i) و خروجی، بیت حاصل‌جمع d_i خواهد بود. در اینجا هم اتومات را بوسیله گرافی با یال‌های دارای برچسب $(a_i, b_i)/d_i$ نمایش می‌دهیم. رقم نقلی از یک مرحله به مرحله بعد توسط اتومات و از طریق دو حالت داخلی با عناوین "بدون نقلی" و "با نقلی" یادآوری می‌شود. در ابتدا، مترجم در حالت "بدون نقلی" قرار می‌گیرد و آنقدر در این



شکل ۱-۸

حالت باقی می‌ماند که با یک زوج بیت (1,1) مواجه شود. این امر باعث تولید رقم نقلی می‌شود که اتومات را به حالت "با نقلی" حرکت می‌دهد. بنابراین، وجود رقم نقلی پس از خواندن زوج بیت بعدی گزارش می‌شود. شکل ۱-۹ تصویر کاملی از یک جمع‌کننده سری را ارائه می‌دهد. سعی کنید با چند مثال از عملکرد درست این شکل طراحی شده، مطمئن شوید.



شکل ۱-۹

براساس این مثال، اتوماتا در نقش پل، بین توصیف بسیار سطح بالا و کاربردی یک مدار و پیاده‌سازی منطقی آن توسط ترانزیستورها، گیت‌ها و فلیپ فلاپ‌ها عمل می‌کنند. هرچند اتومات آشکارا منطق تصمیم‌گیری را نمایش می‌دهد، رسمیت آن به حدی است که می‌توان تغییرات ریاضی در آن انجام داد. به همین دلیل، روش‌های طراحی دیجیتال تا حد زیادی روی مفاهیم نظریه اتوماتا تکیه و تأکید می‌کنند. برای مطالعه بیشتر به منابع مختلف در این زمینه از جمله کتاب(Kovahi (۱۹۷۸ مراجعه کنید.

■

تمرین‌ها

۱. گرامری برای مجموعه اعداد صحیح در C پیدا کنید.
۲. یک پذیرنده برای اعداد صحیح در C طراحی کنید.
۳. گرامری برای تولید تمام اعداد ثابت حقیقی در C ارائه کنید.

۴. فرض کنید که در زبان برنامه‌سازی مفروض، فقط شناسه‌هایی مجاز هستند که با یک حرف شروع شده، حداقل یک و حداکثر سه رقم را شامل شود و هر تعداد حرف داشته باشد. یک گرامر و یک پذیرنده برای این مجموعه شناسه‌ها ارائه دهید.
۵. گرامر مثال ۱-۱۵ را طوری تغییر دهید که شناسه‌ها از قوانین زیر پیروی کنند:
الف) قوانین C ، با این تفاوت که آخرین سمبل از سمت چپ نمی‌تواند یک زیرواژه باشد.
ب) قوانین C ، با این تفاوت که حداکثر یک زیرواژه می‌تواند وجود داشته باشد.
ج) قوانین C ، با این تفاوت که پس از زیرواژه نمی‌توان رقم قرار داد.
۶. گرامری برای نوع خاصی از اعداد حقیقی به نام اعداد حقیقی با نماد علمی بنویسید که از قوانین زیر پیروی کند:
الف) پیش از عدد ممکن است یک علامت + یا یک – قرار بگیرد یا نگیرد.
ب) مقادیر عددی باید به فرم $ab.b_1b_2...b_n$ باشند که در آن b_i می‌تواند هر رقمی باشد، اما a باید یک رقم غیر صفر باشد.
ج) پس از عدد، ممکن است یک فیلد توانی به فرم $e+yy$ یا $e-yy$ قرار گیرد که در آن y ممکن است هر رقمی باشد.
۷. در سیستم اعداد رومی، اعداد بوسیله رشته‌هایی روی الفبای $\{M, D, C, L, X, V, I\}$ نمایش داده می‌شوند. پذیرنده‌ای طراحی کنید که رشته‌هایی را پذیرد که اعداد رومی را به درستی نشان دهند. برای سادگی کار، می‌توانید از قرارداد " سیستم فرعی" استفاده کنید که در آن عدد ۹ بوسیله IX با یک جزء افزوده نمایش داده می‌شود که به جای آن از $VIIII$ استفاده می‌شود.
۸. قبلاً فرض کردیم که یک اتومات در چارچوب مراحل زمانی گسسته کارمی‌کند. هر چند این دیدگاه تأثیر چندانی بر بحث بعدی ما دارد، پارامتر زمان اهمیت قابل ملاحظه‌ای در طراحی دیجیتال دارد.
برای همگام‌سازی سیگنال‌های ورودی از بخش‌های مختلف کامپیوتر، مدارات تأخیر مورد نیاز هستند. مبدل تأخیرواحد، مبدلی است که، خروجی را (که بصورت جریان پیوسته‌ای از سمبل مشاهده می‌شود) به صورت یک واحد زمانی دیرتر، مجدداً تولید می‌کند. مثلاً، اگر مبدل سمبل a را به عنوان ورودی در زمان t بخواند، این سمبل را بعنوان خروجی در زمان $t+1$ مجدداً تولید خواهد کرد. در زمان $t=0$، مبدل هیچ خروجی ندارد. برای شرح آن می‌گوییم که مبدل، ورودی $a_1a_2...$ را به صورت خروجی $\lambda a_1a_2...$ ترجمه می‌کند.
گرافی رسم کنید که نشان دهد چگونه این مبدل تأخیر واحد، روی $\Sigma=\{a,b\}$ عمل می‌کند.
۹. مبدل تأخیر n واحدی، مبدلی است که ورودی را n واحد زمانی دیرتر تولید می‌کند؛ به این معنا که ورودی $a_1a_2...$ به صورت $\lambda^n a_1a_2...$ ترجمه می‌شود. در نتیجه، مبدل هیچ خروجی را برای n برش زمانی اول تولید نخواهد کرد.
الف) یک مبدل تأخیر دو واحدی روی $\Sigma=\{a,b\}$ بسازید.

ب) نشان دهید که یک مبدل تأخیر n واحدی باید حداقل $|\Sigma|^n$ وضعیت داشته باشد.

۱۰. مکمل 2 یک رشته دودویی که بیانگر یک عدد صحیح مثبت است، ابتدا با مکمل کردن هر بیت و سپس با افزودن یک واحد به کم ارزش‌ترین بیت بدست می‌آید. با این فرض که عدد دودویی مانند مثال ۱-۱۷ ارائه شده و کم ارزش‌ترین بیت‌ها در سمت چپ رشته قرار می‌گیرند، مترجمی برای ترجمه رشته‌های بیتی به مکمل 2 آنها طراحی کنید.

۱۱. مبدلی برای تبدیل یک رشته دودویی به هشت هشتی طراحی کنید. بعنوان مثال، بر روی رشته بیتی 001101110 به عنوان ورودی بایستی خروجی ۱۵۶ تولید شود.

۱۲. $a_1a_2\ldots$ را یک رشته بیتی ورودی در نظر بگیرید. مترجمی طراحی کنید که توازن تمام زیررشته‌های سه بیتی را محاسبه کند. خصوصاً، مترجم بایستی خروجی

$$\pi_1 = \pi_2 = 0,$$
$$\pi = (a_{i-2} + a_{i-1} + a_i) \bmod 2 \quad , \quad i = 3,4,\ldots.$$

را تولید کند. بعنوان مثال، ورودی 110111 باید 000001 را تولید کند.

۱۳. مترجمی طراحی کنید که رشته‌های بیتی $a_1a_2a_3\ldots$ را پذیرفته و مقدار دودویی باقیمانده حاصل تقسیم هر مجموعه از سه بیت متوالی بر پنج را محاسبه کند. به طوز ویژه، مترخم باید $m_1,m_2,m_3,\ldots$ را به صورتی تولید کند که در آن

$$m_1 = m_2 = 0,$$
$$m_i = (4a_i + 2a_{i-1} + a_{i-2}) \bmod 5 \quad , \quad i = 3,4,\ldots$$

۱۴. کامپیوترهای رقمی معمولاً با استفاده از روش‌های کدگذاری همهٔ اطلاعات را بوسیله رشته‌های بیتی نمایش می‌دهند. بعنوان مثال، اطلاعات کاراکتری را می‌توان با استفاده از سیستم معروف ASCII کدگذاری کرد.

در این تمرین، به ترتیب دو الفبای $\{0,1\}$ و $\{a,b,c,d\}$ و همچنین کدگذاری از اولی به دومی را فرض کنید که به صورت تعریف شود. مترجمی برای رمزگشایی رشته‌های از 0 و 1 را $a \to 00, b \to 01, c \to 10, d \to 11$ به پیام اولیه بسازید. بعنوان نمونه، ورودی 010011 بایستی به خروجی *bad* تبدیل شود.

۱۵. x و y را دو عدد دودویی مثبت در نظر می‌گیریم. مترجمی با خروجی $\max(x,y)$ طراحی کنید.

فصل ۲

اتوماتای متناهی

مقدمه‌ای که در فصل یک در مورد مفاهیم اساسی محاسبه و خصوصاً مبحث اتوماتا ارائه کردیم، هم مختصر بود و هم غیرصوری. در حال حاضر، ما فقط درک کلی نسبت به ماهیت اتوماتا و نحوه ارائه آنها در قالب گراف‌ها داریم. جهت پیشرفت در بحث باید دقیق‌تر شده و با ارائه تعاریف صوری، پایه محکم‌تری برای نتایج بعدی بسازیم. این کار را با پذیرنده‌های متناهی که یک نمونه ساده و خاص از مدل‌های جامع ارائه شده در فصل قبلی هستند، آغاز می‌کنیم. از ویژگی‌های این نوع اتومات می‌توان به عدم وجود حافظه‌های موقت اشاره کرد. همچنین، به دلیل عدم امکان بازنویسی در فایل ورودی، اتوماتای متناهی از نظر قدرت "به‌خاطرسپاری" وقایع در حین محاسبه به شدت دچار محدودیت هستند. با برنامه‌ریزی واحد کنترل، می‌توان تعداد محدودی اطلاعات را در حافظه نگهداری کرد. اما به دلیل متناهی بودن تعداد این حالت‌ها، اتوماتای متناهی فقط در مواردی قابل استفاده هستند که اطلاعات قابل ذخیره در هر زمان بسیار ناچیز باشد. اتومات مثال ۱-۱۶ نمونه‌ای از پذیرنده‌های متناهی بود.

پذیرنده‌های متناهی معین

اولین نوع از اتوماتایی که بطور دقیق مطالعه خواهیم کرد، پذیرنده‌های متناهی با عملیات معین هستند. ابتدا تعریف صوری و دقیقی از پذیرنده‌های معین ارائه می‌دهیم.

پذیرنده‌های معین و گراف‌های انتقال

پذیرنده‌های معین، همانند تمام اتوماتا، دارای حالت‌های داخلی، توانینی برای انتقال از یک حالت به

حالت دیگر، تعدادی ورودی و همچنین روش‌هایی برای تصمیم‌گیری هستند. همه این موارد در تعریف زیر گنجانده شده‌اند.

تعریف ۲-۱

یک **پذیرنده متناهی معین** یا **dfa** بوسیله پنج‌تایی

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

تعریف می‌شود که در آن،

Q مجموعه متناهی از حالات داخلی،

Σ مجموعه متناهی از علائمی به نام الفبای ورودی،

$\delta: Q \times \Sigma \to Q$ تابعی تام به نام تابع انتقال،

$q_0 \in Q$ حالت شروع و

$F \subseteq Q$ مجموعه حالات پایانی است.

عملکرد پذیرنده‌های متناهی معین به این صورت است که:

برای شروع، فرض می‌شود که پذیرنده در حالت شروع q_0 و هد خواندن از نوار ورودی، روی آخرین سمبل از سمت چپ رشته ورودی قرار دارد. با هریک از حرکت‌های اتومات، هد خواندن به اندازه یک موقعیت به راست منتقل شده و بنابراین، در هر حرکت یک سمبل ورودی استفاده می‌شود. با رسیدن به پایان رشته، در صورتی که اتومات در یکی از حالت‌های پایانی خود قرار داشته باشد، رشته پذیرفته شده و در غیراینصورت، رد می شود. هد خواندن فقط از چپ به راست قابل جابجایی بوده و در هر مرحله، فقط یک سمبل را می‌خواند. انتقال از یک حالت داخلی به حالت داخلی دیگر توسط تابع انتقال δ انجام می‌شود. بعنوان مثال، اگر

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

یعنی اگر dfa در حالت q_0 قرار داشته و سمبل ورودی جاری a باشد، آنگاه dfa به حالت q_1 خواهد رفت.

در بحث راجع به اتوماتا، باید مفهوم شفاف و شهودی آنها را در اختیار داشته باشیم. برای نمایش و ارائه اتوماتای متناهی از **گراف‌های انتقال** استفاده می‌کنیم. در این گراف‌ها، رئوس بیانگر حالت‌ها و یال‌ها بیانگر انتقالات هستند. همچنین، برچسب رئوس، اسامی حالت‌ها بوده و برچسب‌های روی یال‌ها، مقادیر فعلی سمبل ورودی هستند. بعنوان مثال، اگر q_0 و q_1 حالت‌های داخلی dfa مفروض M باشند، آنگاه گراف مربوط به M، یک رأس با برچسب q_0 و رأس دیگری با برچسب q_1 خواهد داشت. بنابراین، یال (q_0, q_1) با برچسب a بیانگر انتقال $\delta(q_0, a) = q_1$ است. حالت شروع بوسیله یک فلش ورودی فاقد برچسب و بدون آغاز از هیچ‌یک از رئوس نمایش داده می‌شود. حالت‌های پایانی با

دو دایرهٔ تودرتو مشخص می‌شوند.

به بیان دقیق‌تر، اگر $M = (Q, \Sigma, \delta, q_0, F)$ یک پذیرنده معین متناهی باشد، آنگاه گراف انتقال مربوط به آن، یعنی G_M، دقیقاً $|Q|$ رأس خواهد داشت که هریک بوسیله یک $q_i \in Q$ متفاوت، نامگذاری می‌شود. به ازای هریک از قوانین انتقال در $\delta(q_i, a) = q_j$، گراف دارای یالی با عنوان (q_i, q_j) و برچسب a خواهد بود. رأس مرتبط با q_0 به نام رأس شروع و رئوس دارای برچسب $q_f \in F$، رئوس پایانی نامیده می‌شوند. تعریف صوری $M = (Q, \Sigma, \delta, q_0, F)$ به عنوان یک dfa به راحتی به گراف انتقال آن G_M و برعکس تبدیل می‌شود.

مثال ۲-۱

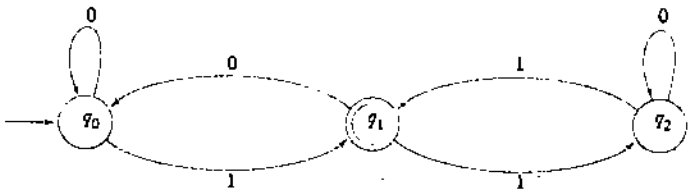
گراف شکل ۲-۱ بیانگر dfa

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

است که در آن، δ از روابط

$$\begin{aligned} &\delta(q_0, 0) = q_0, \quad \delta(q_0, 1) = q_1, \\ &\delta(q_1, 0) = q_0, \quad \delta(q_1, 1) = q_2, \\ &\delta(q_2, 0) = q_2, \quad \delta(q_2, 1) = q_1. \end{aligned}$$

بدست می‌آید. این dfa برای پذیرش رشته 01، با شروع از حالت q_0، ابتدا سمبل 0 را می‌خواند. با نگاهی به یال‌های گراف مشاهده خواهیم کرد که به این ترتیب، اتومات در حالت q_0 قرار می‌گیرد. سپس، سمبل 1 خوانده شده و اتومات به حالت q_1 می‌رود. در این لحظه، هم در پایان رشته و هم درحالت پایانی، یعنی q_1، قرار داریم. بنابراین، رشته 01 پذیرفته می‌شود. پذیرنده متناهی معین، رشته 00 را نمی‌پذیرد. دلیل آنکه پس از خواندن دو 0 متوالی، dfa در حالت q_0 قرار می‌گیرد، رشته 00 قابل پذیرش نمی‌باشد. با استدلالی مشابه، مشخص می‌شود که اتومات رشته‌های 101، 0111 و 11001 را پذیرفته، ولی 100 یا 1100 را رد می‌کند.



شکل ۲-۱

■

اینک می‌توان تابع انتقال گسترش یافته $\delta^*: Q \times \Sigma^* \to Q$ را تعریف کرد. آرگومان دوم δ^* یک رشته است، نه فقط یک سمبل. و مقدار δ^* بیانگر حالت اتومات پس از خواندن این رشته می‌باشد.

بعنوان مثال، اگر

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

و

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

آنگاه،

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

به بیان دقیق‌تر، δ^* را می‌توان به بطور بازگشتی، بصورت زیر:

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

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

برای همه $a\in\Sigma$، $w\in\Sigma^*$ و $q\in Q$ تعریف کرد. برای اطمینان از درستی تعاریف فوق، آنها را در نمونه ساده بالا بکار می‌بریم. ابتدا، با استفاده از (۲-۲)

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

را بدست می‌آوریم.

اما

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

با جایگزینی این رابطه در (۲-۳)، همانطور که انتظار داشتیم،

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

بدست خواهد آمد.

زبان‌ها و dfaها

اینک با در اختیار داشتن تعریف دقیقی از پذیرنده‌ها، منظور خود از زبان مربوط به پذیرنده را در قالب یک تعریف صوری می‌گنجانیم. می‌توان گفت که، زبان مجموعه‌ای از تمام رشته‌های پذیرفته شده توسط اتومات می‌باشد.

تعریف ۲-۲

زبان پذیرفته شده توسط dfa ی $M=(Q,\Sigma,\delta,q_0,F)$، مجموعه تمام رشته‌های روی Σ است که توسط M پذیرفته می‌شوند. به فرم صوری،

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

تعریف می‌شود.

توجه داشته باشید که در اینصورت، δ و در نتیجه δ^* باید توابع تام باشند. در هر مرحله، فقط یک حرکت تعریف می‌شود و به همین دلیل، چنین اتوماتی را می‌توان معین نامید. dfa پس از پردازش هریک از رشته‌های Σ^*، یا آنها را می‌پذیرد و یا رد می‌کند. عدم پذیرش به این معناست که dfa در یکی از حالت‌های غیرپایانی متوقف شده و در نتیجه،

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

مثال ۲-۳

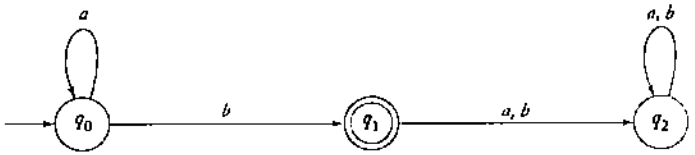
dfa شکل ۲-۲ را در نظر بگیرید.

در این شکل، روی یکی از پال‌ها دو برچسب وجود دارد. اینگونه پال‌های برچسب‌دار، بیانگر دو یا چند انتقال مجزا هستند؛ یعنی به محض تطابق سمبل ورودی با هر یک از برچسب‌های یال، یک انتقال انجام می‌شود.

اتومات شکل ۲-۲ آنقدر در حالت شروع q_0 باقی می‌ماند تا با اولین b برخورد پیدا کند. اگر این b آخرین سمبل ورودی هم باشد، رشته پذیرفته می‌شود. در غیر اینصورت، dfa وارد حالت q_2 شده و هرگز از آن خارج نخواهد شد. حالت q_2 اصطلاحاً حالت دام یا تله نامیده می‌شود. با بررسی گراف به راحتی می‌توان مشاهده کرد که این اتومات، تمام رشته‌ها با هر تعداد دلخواه a را که پس از آن فقط یک b قرارداشته باشد، خواهد پذیرفت. مابقی رشته‌های ورودی حذف می‌شوند. بنابراین زبان پذیرفته شده بوسیله این اتومات

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

خواهد بود.



شکل ۲-۲

■

این مثال‌ها بیانگر کاربرد گراف‌های انتقال برای نمایش اتوماتای متناهی هستند. هرچند می‌توان تمام استدلال‌ها را منحصراً براساس ویژگی‌های تابع انتقال و تعاریف گسترش یافته آن از طریق روابط (۲-۱) و (۲-۲) استوار کرد، نتایج حاصل از این‌کار چندان قابل فهم نیستند. در این کتاب، ما از گراف‌هایی استفاده می‌کنیم که تا حد امکان شهودی‌تر باشند. البته پیش از آن، باید مطمئن شویم که مفاهیم گراف‌ها دقیق بوده و بعلاوه، اعتبار استدلال‌های ناشی از این گراف‌ها به اندازه استدلال‌های ناشی از دیگر گراف‌هایی است که از ویژگی‌های دقیق δ استفاده می‌کنند. نتیجه اولیه زیر این اطمینان را به ما می‌دهد.

قضیه ۲-۱

$M=(Q,\Sigma,\delta,q_0,F)$ یک پذیرنده متناهی معین و G_M گراف انتقال مربوط به آن است. بنابراین به ازای هر $q_i,q_j\in Q$ و $w\in\Sigma^*$، $\delta^*(q_i,w)=q_j$ اگر و تنها اگر در G_M یک قدم با برچسب w از q_i به q_j وجود داشته باشد.

اثبات: این حکم با بررسی نمونه‌های ساده‌ای از قبیل مثال ۲-۱ و همچنین، با استفاده از استقراء روی طول w به راحتی اثبات می‌شود. فرض کنید که این حکم به ازای تمام رشته‌های v با طول $|v|\le n$ برقرار باشد. سپس، هر w با طول $n+1$ را در نظر گرفته و آنرا به صورت

$$w=va$$

می‌نویسیم. حال فرض کنید که $\delta^*(q_i,v)=q_k$. بدلیل آنکه $|v|=n$، بنابراین باید یک قدم در G_M با برچسب v از q_i به q_k وجود داشته باشد. اما اگر $\delta^*(q_i,w)=q_j$، آنگاه M باید یک انتقال $\delta(q_k,a)=q_j$ داشته باشد و به همین دلیل براساس ساختار G_M، یک یال به صورت (q_k,q_j) با برچسب a خواهد داشت. بنابراین در G_M، یک قدم با برچسب $w=va$ بین q_i و q_j وجود دارد. از آنجایی‌که نتیجه برای $n=1$ صحیح است، می‌توان براساس استقراء ادعا کرد که به ازای هر $w\in\Sigma^*$،

$$\delta^*(q_i,w)=q_j \qquad (۲-۴)$$

بطور بدیهی به معنای وجود یک قدم از q_i به q_j با برچسب w در G_M است.

می‌توان با ادامه روند استدلال، نشان داد که وجود چنین مسیری ضمناً به معنای رابطه (۲-۴) بوده و به این ترتیب، اثبات کامل می‌شود. ■

نتیجه: قضیه بالا آنقدر از نظر شهودی مشخص است که نیازی به اثبات صوری برای آن وجود ندارد. اما به دو دلیل به سراغ جزئیات می‌رویم. اولاً، این قضیه نمونه ساده و البته خاصی از اثبات استقراء در ارتباط با اتوماتا است. ثانیاً، از آنجایی‌که یک نتیجه به دفعات قابل استفاده است، بیان و اثبات آن در قالب قضیه به ما امکان می‌دهد تا با اطمینان خاطر آن را در گراف‌ها مورد استفاده قراردهیم. به این ترتیب، مثال‌ها و اثبات‌های ما شفاف‌تر از زمانی خواهد شد که از ویژگی‌های δ^* استفاده می‌کنیم.

هرچند گراف‌ها برای نمایش اتوماتا بسیار مناسب هستند، روش‌های دیگری هم برای این کار وجود دارد. بعنوان مثال، همچنین می‌توان تابع δ را به صورت جدول ارائه کرد. جدول ۲-۳ مربوط به شکل ۲-۲ می‌باشد. در این جدول، نام سطر، بیانگر حالت فعلی و نام ستون، بیانگر سمبل ورودی فعلی است. درایه‌های جدول هم معرف حالت بعدی خواهد بود.

براساس این مثال، dfa را می‌توان به راحتی در قالب یک برنامه کامپیوتری، مثلاً به صورت جستجوی ساده‌ای در جدول یا به شکل دنباله‌ای از عبارات *if*، پیاده‌سازی نمود. بهترین روش پیاده‌سازی یا ارائه، بر حسب کاربرد خاص آن dfa تعیین می‌شود. از آنجایی‌که گراف‌های انتقال برای تمام انواع استدلال‌های این کتاب بسیار مفید و قانع‌کننده هستند، در بیشتر استدلال‌های خود از آنها استفاده خواهیم کرد.

جهت ساخت اتوماتا برای زبان‌هایی که بصورت صوری تعریف شده‌اند، از استدلالی مشابه برنامه‌سازی در زبان‌های سطح بالا استفاده می‌کنیم. اما به دلیل قدرت کم این گروه اتوماتا، برنامه‌سازی برای dfa کسل‌کننده و گاهی از نظر مفهومی پیچیده می‌شود.

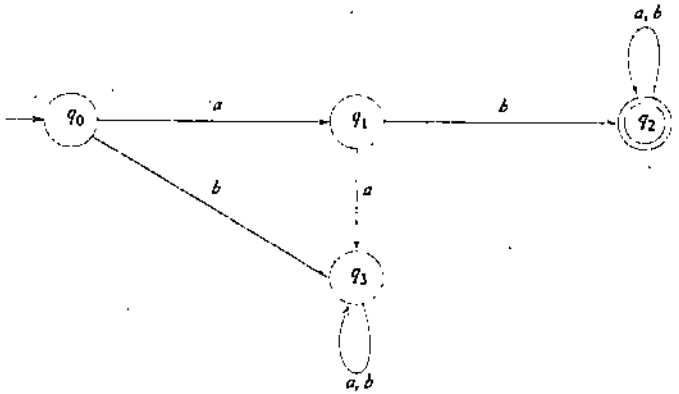
	a	b
q_0	q_0	q_1
q_1	q_2	q_2
q_2	q_2	q_2

شکل ۲-۳

مثال ۲-۳

یک پذیرنده متناهی معین پیدا کنید که مجموعه تمام رشته‌های روی $\Sigma=\{a,b\}$ با پیشوند شروع ab را شناسایی کند.

در اینجا فقط لازم است که دو سمبل اول رشته را تعیین کنیم. گرچه پس از خواندن این دو سمبل، اتومات نیازی به هیچگونه تصمیم‌گیری دیگری ندارد، اما بایستی قبل از تصمیم گیری، تمام رشته را پردازش کند. بنابراین می‌توان مسأله را با یک اتومات چهار حالتی – یک حالت شروع، دو حالت برای شناخت ab منتهی به حالت تله پایانی و یک حالت تله غیرپایانی– حل کرد. اگر سمبل اول یک a و سمبل دوم هم یک b باشد، اتومات به حالت تله پایانی رفته و آنقدر در آنجا باقی می‌ماند تا ورودی تمام شود. اما اگر سمبل اول a نبوده یا سمبل دوم b نباشد، اتومات وارد حالت تله غیرپایانی می‌شود. این روش ساده در شکل ۲-۴ ارائه شده است.



شکل ۲-٤

مثال ۲-۴

یک پذیرنده متناهی معین پیدا کنید که تمام رشته‌های روی {0,1}، بجز رشته‌های دارای زیررشته 001 را بپذیرد.

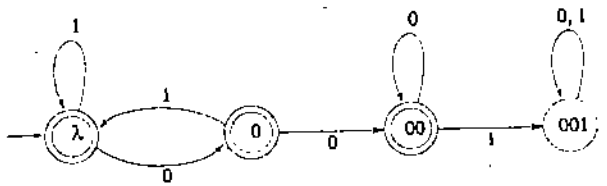
برای تصمیم‌گیری در مورد وقوع یا عدم وقوع زیررشته 001، باید علاوه بر آگاهی از سمبل ورودی فعلی، بدانیم که آیا پیش از آن، یک سمبل 0 قرار دارد یا دو سمبل 0. این کار را می‌توان با قرار دادن اتومات در حالت‌های خاص و برچسب‌گذاری تطابقی حالت‌ها انجام داد. مشابه نامگذاری متغیرها در زبان‌های برنامه‌سازی، اسامی حالات نیز اختیاری بوده و می‌توان آنها را به صورتی انتخاب کرد که راحت‌تر به خاطر سپرده شوند. بعنوان مثال، حالتی را که در آن دو 0 دقیقاً قبل از سمبل‌ها قرار گرفته باشد، می‌توان به فرم 00 برچسب‌دار کرد.

بنابراین رشته‌هایی که با 001 شروع می‌شوند، باید رد شوند. به این معنا که باید یک مسیر با برچسب 001 از حالت شروع به یک حالت غیرپایانی وجود داشته باشد. برای اطمینان، این حالت غیرپایانی به صورت 001 نامگذاری می‌شود. به دلیل بی‌تأثیر بودن سمبل‌های بعدی، این حالت یک حالت تله است. مابقی حالتها، حالتهای پذیرش هستند.

هر چند به این ترتیب ساختار اساسی راه‌حل بدست می‌آید، باید بازهم پیش‌بینی‌هایی را برای زیررشته 001 که در میانه ورودی رخ می‌دهد، ارائه دهیم. یعنی Q و δ را به صورتی تعریف کنیم که هرآنچه برای تصمیم‌گیری درست نیاز است، بوسیله اتومات به خاطر آورده شود. برای این منظور، پس از خواندن یک سمبل، باید بخشی از سمت چپ رشته را هم بدانیم. مثلاً بدانیم که آیا دو سمبل قبلی 00 بوده‌اند یا خیر. اگر حالت‌ها را با نام‌های مرتبط برچسب‌دار کنیم، پیش‌بینی انتقال‌ها بسیار ساده خواهد شد. بعنوان مثال،

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

چون این وضعیت فقط در صورت وجود سه سمبل 0 متوالی رخ می‌دهد. ما فقط به دو 0 آخر علاقه‌مند بوده و به همین دلیل، dfa را در حالت 00 نگه می‌داریم. شکل ۲-۵ راه‌حل کامل این کار را ارائه می‌دهد. از این مثال می‌توان به اهمیت و کارآیی برچسب‌های تطابقی در پی‌گیری مسائل پی برد. با مسیریابی چند رشته، مثلاً 100100 و 1010100، درستی این راه‌حل را بررسی کنید.



شکل ۲-۵

■

زبان‌های منظم

هر اتوماتای متناهی زبان خاصی را می‌پذیرند. بنابراین اگر همه اتوماتاهای متناهی ممکن را

درنظربگیریم، یک مجموعه زبان متناظر با آنها وجود خواهد داشت. این مجموعه زبان‌ها خانواده نامیده می‌شوند. خانواده زبان‌هایی که بوسیله پذیرنده‌های متناهی معین پذیرفته می‌شوند بسیار محدود است. در ادامه، بیشتر راجع به ساختار و ویژگی‌های زبان‌های این خانواده بحث خواهیم کرد. اما در حال حاضر فقط یک نام برای این خانواده ارائه داده و بررسی آنها را به زمان دیگری موکول می‌کنیم.

تعریف ۲-۳

زبان مفروض L منظم خوانده می‌شود اگر و تنها اگر یک پذیرنده متناهی معین M وجود داشته باشد بطوریکه

$$L = L(M).$$

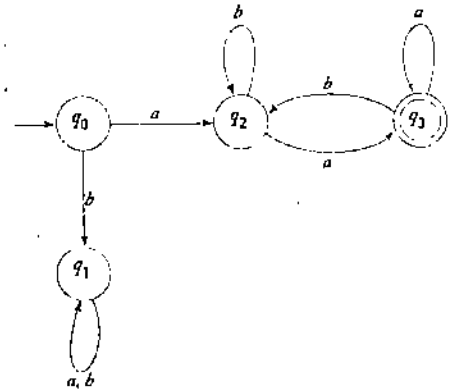
مثال ۲-۵

نشان دهید که زبان

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

منظم است.

برای اثبات منظم بودن این زبان یا هر زبان دیگری، تنها لازم است که یک dfa برای آن پیدا کنیم. ساختار dfa این زبان، مشابه مثال ۲-۳، اما کمی پیچیده‌تر است. dfa مورد نظر باید وجود یا عدم وجود رشته‌ای با شروع و پایان a، با هر زیررشته‌ای، بررسی کند. اما چون هیچ روش مشخصی برای آزمودن پایان رشته وجود ندارد، باید به محض مواجه شدن با a دوم، dfa را در حالت پایانی قرار دهیم. اگر این a پایان رشته نبوده و یک b دیگر پیدا شود، dfa از حالت پایانی خارج خواهد شد. کار بررسی به همین ترتیب تا انتها ادامه پیدا کرده و هرسمبل a ورودی، اتومات را به حالت پایانی آن بازمی‌گرداند. راه‌حل کامل مسأله در شکل ۲-۶ ارائه شده است. سعی کنید با چند مثال از عملکرد



شکل ۲-۶

درست آن مطمئن شوید. پس از یک یا دو بررسی با مثال، به راحتی درک خواهیم کرد که dfa فقط و فقط رشته‌ای را می‌پذیرد که با یک a شروع و با یک a خاتمه یابد. حال که توانستیم یک dfa برای زبان بسازیم، براساس تعریف قبل می‌توان ادعا کرد که زبان مورد نظر منظم است.

■

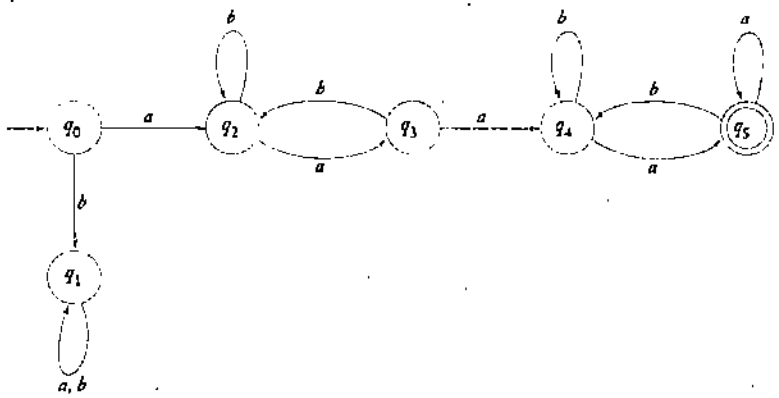
مثال ۲-۶

فرض کنید L زبان مثال ۲-۵ باشد. نشان دهید که L^2 منظم است.

مجدداً با ساخت یک dfa برای زبان، منظم بودن آنرا نشان می‌دهیم. L^2 را به صورت

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

می‌نویسیم. بنابراین، dfaی می‌خواهیم که دو رشته متوالی بر پایه یک مجموعه الفبا (لزوماً رشته‌ها برابر نیستند) را شناسایی کند. می‌توان برای شروع کار از گراف شکل ۲-۶ استفاده کرد، ولی رأس q_3 باید تغییر پیدا کند. اما چون از همین نقطه به دنبال زیررشته دوم به فرم awa می‌گردیم، q_3 دیگر حالت پایانی نخواهد بود. برای شناخت زیررشته دوم، حالت‌های بخش اول را (با اسامی جدید) کپی‌برداری کرده و q_3 را آغاز بخش دوم در نظر می‌گیریم. بدلیل آنکه می‌توان رشته کامل را از محل وقوع aa به بخش‌های تشکیل‌دهنده آن تجزیه کرد، اولین وقوع دو a متوالی را دلیل فعال شدن عملکرد اتومات در بخش دوم در نظر می‌گیریم. این کار را می‌توان بوسیله $\delta(q_3,a)=q_4$ انجام داد. شکل ۲-۷ راه‌حل کامل این مسأله را ارائه می‌دهد. این dfa، L^2 را می‌پذیرد و بنابراین L^2 منظم است.



شکل ۲-۷

■

از مثال آخر می‌توان نتیجه گرفت که اگر زبان مفروض L منظم باشد، L^2، L^3 و ... هم منظم خواهند بود. بعداً صحت این ادعا را اثبات خواهیم کرد.

تمرین‌ها

۱. dfa شکل ۲-۱ کدام یک از رشته‌های 0001، 01001 و 0000110 را می‌پذیرد؟

۲. به ازای $\Sigma = \{a,b\}$، dfaهایی را بسازید که مجموعه‌های زیر را بپذیرد:

الف) تمام رشته‌های دارای دقیقاً یک a،

ب) تمام رشته‌های دارای حداقل یک a،

ج) تمام رشته‌های دارای حداکثر سه a، ◄

د) تمام رشته‌های دارای حداقل یک a و دقیقاً دو b،

ه) تمام رشته‌های دارای دقیقاً دو a و بیش از دو b.

۳. نشان دهید که اگر در شکل ۲-۶، q_1 را حالت غیرپایانی و q_0، q_2 و q_3 را حالتهای پایانی در نظر بگیریم، dfa حاصله $\overline{L}$ را خواهد پذیرفت.

۴. نتیجه تمرین قبل را تعمیم دهید. خصوصاً، نشان دهید که اگر $M = (Q,\Sigma,\delta,q_0,F)$ و $\hat{M} = (Q,\Sigma,\delta,q_0,Q-F)$ دو dfa باشند، آنگاه $\overline{L(M)} = L(\hat{M})$.

۵. Dfa هایی را برای زبان‌های زیر ارائه دهید:

الف) $L = \{ab^nwb^2 : w \in \{a,b\}^*\}$، ◄

ب) $L = \{ab^na^m : n \geq 2, m \geq 3\}$،

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

۶. با فرض $\Sigma = \{a,b\}$، یک dfa برای $L = \{w_1aw_2 : |w_1| \geq 3, |w_2| \leq 5\}$ ارائه دهید.

۷. برای زبان‌های زیر، dfaهایی روی $\Sigma = \{a,b\}$ پیدا کنید:

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) $L = \{w : (n_a(w) + 2n_b(w)) \bmod 3 < 2\}$،

۸. *یک دور در یک رشته، طولانی‌ترین زیررشته‌های مجاز در آن با سمبل‌های یکسان و با حداقل طول دو است. بعنوان مثال، رشته abbbaab حاوی دوری از b ها با طول سه و دوری از a ها با طول دو است. dfa هایی را برای زبان‌های زیر روی $\{a,b\}$ پیدا کنید.

الف) $L = \{w$: w حاوی هیچ دوری با طول کمتر از چهار نباشد$\}$.

ب) $L = \{w$: w هر دوری از aها با طول دو یا سه$\}$.

ج) $L = \{w$: w حداکثر دو دور a با طول سه وجود داشته باشد$\}$.

د) $L = \{w$: w دقیقاً دو دور a با طول ۳ وجود داشته باشد$\}$.

۹. مجموعه رشته‌های روی $\{0,1\}$ را در نظر بگیرید که براساس شرایط زیر تعریف می‌شود. dfa پذیرنده هریک را بسازید.

الف) پس از هر 00 فوراً یک سمبل 1 باشد. بعنوان مثال، رشته‌های 101، 0010 و 0010011001 عضو این زبان می‌باشند، ولی 0001 و 00100 عضو نمی‌باشند.

ب) تمام رشته‌های حاوی 00 و فاقد 000.

ج) آخرین سمبل از سمت چپ با آخرین سمبل از سمت راست متفاوت باشد.

د) هر یک از زیررشته‌های چهار سمبلی که حداکثر دو تا سمبل 0 دارند. بعنوان مثال، 001110 و 011001 در زبان وجود دارند، ولی 10010، به این دلیل که یکی از زیررشته‌های آن، یعنی 0010، حاوی سه صفر است در آن وجود ندارد.

ه) تمام رشته‌های دارای طول پنج یا بیشتر که چهارمین سمبل از سمت راست آن با آخرین سمبل از سمت چپ متفاوت باشد.

و) تمام رشته‌هایی که دو سمبل آخر از سمت چپ و دو سمبل از سمت راست آن مشابه باشند.

ز) تمام رشته‌ها با طول چهار یا بیشتر که سه سمبل آخر از سمت چپ آنها یکسان هستند، اما سمبل سمت راست آنها متفاوت باشند.

۱۰. * – یک dfa بسازید که رشته‌های روی $\{0,1\}$ را فقط و فقط در صورتی بپذیرد که مقدار رشته که به صورت نمایش دودویی از یک عدد صحیح تفسیر می‌شود، با پیمانه پنج، برابر صفر باشد. بعنوان مثال، 0101 و 1111 که به ترتیب بیانگر اعداد صحیح ۵ و ۱۵ هستند، پذیرفته می‌شوند.

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

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

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

۱۴. نشان دهید که زبان $L=\{a^n :$ n یا ضریب سه است یا ضریب ۵$\}$ منظم است.

۱۵. نشان دهید که زبان $L=\{a^n :$ n ضریب سه است و ضریب ۵ نیست$\}$ منظم است.

۱۶. نشان دهید که مجموعه تمام اعداد حقیقی در زبان برنامه نویسی C یک زبان منظم است.

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

۱۸. نشان دهید که اگر L منظم باشد، آنگاه $L \cup \{a\}$ به ازای تمام $a \in \Sigma$ منظم است.

۱۹. با استفاده از (۲-۱) و (۲-۲) نشان دهید که به ازای همه $w,v \in \Sigma^*$،

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

۲۰. فرض کنید که L زبان پذیرفته شده بوسیله اتومات شکل ۲-۲ باشد. یک dfa بسازید که L^2 را بپذیرد.

۲۱. فرض کنید که L زبان پذیرفته شده بوسیله اتومات شکل ۲-۲ باشد. یک dfa برای زبان L^2-L بسازید.

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

۲۳. فرض کنید که G_M گراف انتقال برای dfa مفروض M باشد. موارد زیر را اثبات کنید.

الف) اگر $L(M)$ نامتناهی باشد، آنگاه باید حداقل در یکی از حلقه‌های G_M، یک مسیر از رأس شروع به یکی از رئوس در حلقه و یک مسیر از یکی از رئوس حلقه به یکی از رئوس پایانی وجود داشته باشد.

ب) اگر $L(M)$ متناهی باشد، آنگاه چنین حلقه‌ای وجود نخواهد داشت.

۲۴. عملگر *truncate* آخرین سمبل از سمت راست هر رشته را حذف می‌کند. بعنوان مثال، *truncate(aaaba)* ، *aaab* خواهد بود. این عملگر را می‌توان بوسیله

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

برای زبان تعمیم داد. نشان دهید که چگونه می‌توان، با داشتن یک dfa برای هر زبان منظم L ، یک dfa هم برای *truncate* (L) ساخت. به همین ترتیب اثبات کنید که اگر L یک زبان منظم فاقد λ باشد، آنگاه *truncate* (L) منظم خواهد بود.

۲۵. هر چند زبان پذیرفته شده بوسیله یک dfa منحصربفرد است، معمولاً تعداد زیادی dfa این زبان را می‌پذیرند. dfa شش حالتی پیدا کنید که زبان dfa شکل ۲-۲ را بپذیرد.

۲-۳ پذیرنده‌های متناهی نامعین

پذیرنده‌های متناهی نامعین پیچیده‌تر از انواع معین خود هستند. این نامعین بودن آنها را تبدیل به ابزاری قدرتمند، اما نامتعارف کرده است. ما معمولاً کامپیوترها را بصورت کاملاً معین تصور می‌کنیم. با این وجود و همانطور که در ادامه خواهیم دید، نامعین‌بودن ایده مفیدی محسوب می‌شود.

تعریف پذیرنده‌های نامعین

نامعین‌بودن باعث می‌شود تا بتوان حرکات اتومات را انتخاب کرد. بوسیله این ویژگی می‌توانیم به‌جای اجبارکردن یک حرکت منحصربه‌فرد برای هر وضعیت، مجموعه‌ای از حرکات مجاز را برای آن پیش‌بینی کنیم. این کار را بطور دقیق بوسیله تعریف تابع انتقال انجام می‌دهیم که برد آن مجموعه‌ای از حالات مجاز برای اتومات باشد.

تعریف ۲-۴

یک **پذیرنده متناهی نامعین** یا **nfa** بوسیله پنج‌تایی

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

تعریف می‌شود که در آن Q، Σ، q_0 و F همانند پذیرنده‌های متناهی معین تعریف می‌شوند، ولی

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

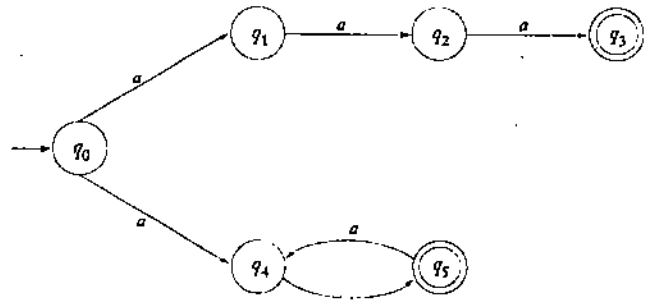
توجه داشته باشید که سه تفاوت عمده بین تعریف nfa و تعریف dfa وجود دارد:
اولاً، در پذیرنده‌های نامعین، برد δ مجموعه توانی 2^Q می‌باشد، بطوریکه اعضاء آن نه عضو Q، بلکه زیرمجموعه‌ای از آن می‌باشد. زیرمجموعه مذکور مجموعه تمام حالات مجازی را تعریف می‌کند که بوسیله انتقال قابل دسترسی هستند. بعنوان مثال، اگر حالت فعلی q_1 باشد، سمبل a خوانده شده و

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

آنگاه، q_0 یا q_2 حالت بعدی nfa خواهد بود.
ثانیاً، λ را بعنوان دومین استدلال δ در نظر می‌گیریم. به این معنا که nfa می‌تواند بدون استفاده از سمبل ورودی، دست به انتقال بزند. هرچند فرض می‌کنیم که هد ورودی فقط به سمت راست حرکت می‌کند، با تعریف نامعین ممکن است در بعضی انتقال‌ها حرکت نکند. نکته آخر اینکه، تهی بودن مجموعه $\delta(q_i,a)$ در nfa به این معناست که هیچ انتقالی برای این وضعیت خاص تعریف نشده است.
پذیرنده‌های نامعین، مانند dfaها، بوسیله گراف‌های انتقال نمایش داده می‌شوند. رئوس بوسیله Q مشخص می‌شوند، در حالیکه یال (q_i,q_j) با برچسب a فقط و فقط در صورتی در گراف وجود خواهد داشت که $\delta(q_i,a)$ حاوی q_j باشد. توجه داشته باشید که به دلیل احتمال تهی بودن a، برخی یال‌ها را می‌توان با λ نامگذاری کرد.
یک رشته در صورتی بوسیله nfa پذیرفته می‌شود که دنباله‌ای از انتقال‌های احتمالی بتواند ماشین را با تمام‌شدن رشته در حالت پایانی قرار دهد. بالعکس، یک رشته فقط در صورتی رد می‌شود (یا پذیرفته نمی‌شود) که نتوان بوسیله هیچ دنباله احتمالی از انتقال‌ها به حالت پایانی دست یافت. بنابراین، (با این فرض که nfa تمام رشته‌ها را می‌پذیرد)، نامعین بودن را می‌توان مدلی از نگرش "شهودی" و استفاده از این نگرش جهت انتخاب بهترین انتقال در هر حالت تصور کرد.

مثال ۲-۸

گراف انتقال شکل ۲-۸ را در نظر بگیرید. به دلیل وجود دو انتقال با برچسب a از q_0، این گراف مربوط به یک پذیرنده نامعین می‌باشد.

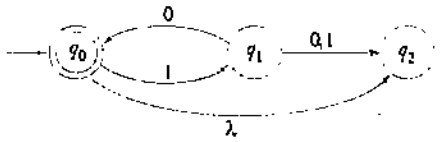


شکل ۲-۸

■

مثال ۲-۸

در شکل ۲-۹ یک اتومات نامعین را مشاهده می‌کنید. در این اتومات، چندین یال با برچسب یکسان از یک رأس آغاز شده و همچنین دارای یک انتقال λ می‌باشد. برخی انتقال‌ها از قبیل $\delta(q_2,0)$ در گراف به چشم نمی‌خورد. این وضعیت را می‌توان انتقال به مجموعه تهی تفسیر کرد، یعنی $\delta(q_2,0)=\varnothing$. اتومات مذکور رشته‌های λ، 1010 و 101010 را می‌پذیرد، اما 110 و 10100 را رد می‌کند. توجه داشته باشید که برای 10 دو قدم وجود دارد که یکی به q_0 و دیگری به q_2 ختم می‌شود. حتی در صورتی که q_2 حالت پایانی نباشد، بدلیل آنکه یکی از قدم‌ها به حالت پایانی ختم می‌شود، رشته پذیرفته خواهد شد.



شکل ۲-۹

■

در اینجا هم، تابع انتقال را می‌توان به صورتی گسترش داد که مولفه دوم آن رشته باشد. به این منظور، نیاز به تابع انتقال گسترش یافته δ^* داریم که اگر

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

آنگاه Q_j مجموعه تمام حالتهای مجاز اتومات با آغاز از حالت q_i و خواندن w از روی نوار ورودی می‌باشد. در اینجا نیز می‌توان تعریفی بازگشتی برای δ^* مشابه (۲-۱) و (۲-۲) ارائه نمود. اما به دلیل ابهام این تعریف، می‌توان به جای آن، از تابع‌های انتقال استفاده نمود.

تعریف ۲-۵

در یک nfa، تابع انتقال گسترش یافته به صورتی تعریف می‌شود که $\delta^*(q_i,w)$ فقط و فقط در صورتی حاوی q_j باشد که در تابع انتقال، یک قدم از q_i به q_j با برچسب w وجود داشته باشد. این ویژگی در مورد همه $q_i,q_j\in Q$ و $w\in\Sigma^*$ صدق می‌کند.

مثال ۲-۹

شکل ۲-۱۰ یک nfa را نمایش می‌دهد. این nfa دارای چندین انتقال λ و تعدادی انتقال تعریف نشده از قبیل $\delta(q_i,a)$ است.

فرض کنید که $\delta^*(q_1,a)$ و $\delta^*(q_1,\lambda)$ مورد سوال باشد. در این nfa، یک قدم با برچسب a شامل دو انتقال λ از q_1 به خود آن وجود دارد. با دو بار استفاده از برخی یالهای λ، قدم‌هایی شامل انتقالات به q_0 و q_2 بوجود می‌آیند. بنابراین،

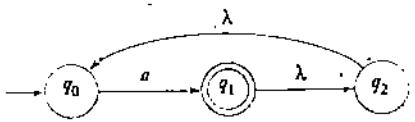
$$\delta^*(q_1,a)=\{q_0,q_1,q_2\}.$$

بدلیل آنکه یک یال λ بین q_2 و q_0 وجود دارد، بنابراین $\delta^*(q_1,\lambda)$ حاوی q_0 خواهد بود. بعلاوه، چون می‌توان بدون هیچ انتقالی از یک حالت به خود آن رسید و در نتیجه هیچگونه سمبل ورودی را استفاده نکرد، $\delta^*(q_2,\lambda)$ حاوی q_2 نیز خواهد بود. بنابراین،

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

با استفاده از تعدادی انتقال λ، شما می‌توانید صحت عبارت زیر را بررسی کنید:

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



شکل ۲-۱۰

■

تعریف δ^* با استفاده از قدم‌هایی برچسب‌دار چندان رسمی نیست، بنابراین بهتر است از روش دیگری استفاده کنیم. در تعریف ۲-۵، بین هریک از رئوس v_i و v_j یک قدم با برچسب w یا وجود دارد و یا ندارد؛ یعنی δ^* بطور کامل تعریف شده و به همین دلیل تعریف مناسبی تلقی می‌شود. بنابراین، همواره می‌توان از آن برای پیداکردن $\delta^*(q_i,w)$ استفاده کرد.

در بخش ۱-۱، الگوریتمی برای پیداکردن تمام مسیرهای ساده بین دو رأس ارائه دادیم. اما چون بر اساس شکل ۲-۹، هر قدم برچسب‌داری، همواره مسیر ساده محسوب نمی‌شود، نمی‌توان بطورمستقیم از این الگوریتم استفاده کرد. بلکه باید با حذف ممنوعیت تکرار رئوس و یال‌ها، الگوریتم مسیر ساده را اصلاح نمود. الگوریتم حاصله به طور کامل تمام قدم‌ها با طول یک، دو، سه و بیشتر را تولیدمی‌کند.

اما هنوز یک پرسش مطرح است. در صورت داشتن w، قدم با برچسب w چه طولی خواهد داشت؟ یافتن پاسخ این سؤال نیاز به بررسی دارد. در مثال ۲-۹، قدم با برچسب a در بین q_1 و q_2، طولی معادل چهار دارد. طولانی بودن قدم، ناشی از انتقال λ بوده و ارتباطی به برچسب ندارد. یک راه‌حل این است که چنانچه بین دو رأس v_i و v_j، قدم با برچسب w وجود داشته باشد، آنگاه باید قدم با برچسب w و با طولی معادل $|w|(1+\Lambda)+\Lambda$ وجود داشته باشد بطوریکه Λ تعداد یالهای λ در گراف است. راه‌حل را به این صورت بررسی می‌کنیم: علیرغم احتمال تکرار یال λ، همواره و قطعاً یک قدم وجود خواهد داشت که در آن، تمامی یالهای λ تکرارشده بوسیله یالی با برچسب سمبل

غیرتهی از هم تفکیک می‌شوند. در غیر اینصورت، قدم شامل یک حلقه با برچسب λ بوده و می‌توان بدون تغییر برچسب قدم، آنرا با یک مسیر ساده جایگزین کرد. اثبات دقیق این ادعا را بعنوان تمرین به خود شما واگذار می‌کنیم.

به این ترتیب، روشی برای محاسبه $\delta^*(q_i,w)$ بوجود می‌آید:

ابتدا تمامی قدم‌ها با طول حداکثر $|w|(1+\Lambda)+\Lambda$ با شروع از q_i را ارزیابی کرده و سپس از بین آنها، قدم‌هایی با برچسب w را انتخاب می‌کنیم. رئوس پایانی قدم‌هایی انتخاب شده، عضو مجموعه $\delta^*(q_i,w)$ خواهند بود.

همانطور که قبلاً هم اشاره شد، می‌توان δ^* را مانند پذیرنده‌های نامعین، به صورت بازگشتی تعریف کرد. اما نتیجه کار چندان روشن نبوده و استدلال‌های مربوط به تابع انتقال گسترش یافته‌ای که به این صورت تعریف می‌شوند به راحتی قابل درک نمی‌باشد. به همین دلیل روش شهودی‌تر و قابل مدیریت‌تر تعریف ۲-۵ ارجح است.

همانند dfaها، زبان پذیرش شده بوسیله یک nfa، بطور دقیق بوسیله تابع انتقال گسترش‌یافته قابل تعریف است.

تعریف ۲-۶

زبان L پذیرفته شده، بوسیله یک پذیرنده متناهی نامعین $M=(Q,\Sigma,\delta,q_0,F)$ توسط مجموعه‌ای از تمام رشته‌های پذیرفته شده براساس مفاهیم بالا، تعریف می‌شود. به بیان دقیق،

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

به عبارت دیگر، این زبان شامل تمام رشته‌های w است که به ازای آن، یک قدم با برچسب w از رأس شروع گراف انتقال به حداقل یکی از رئوس پایانی وجود داشته باشد.

مثال ۲-۱۰

زبان پذیرفته شده توسط اتومات شکل ۲-۹ چیست؟

با توجه به گراف، به راحتی می‌توان مشاهده کرد که تنها راه برای قرارگرفتن و توقف nfa در حالت پایانی آن است که ورودی یا تکراری از رشته 10 باشد و یا رشته تهی. بنابراین، اتومات زبان $L=\{(10)^n:n\geq 0\}$ را می‌پذیرد.

این اتومات پس از قرارگرفتن در ابتدای رشته $w=110$ و خواندن پیشوند 11، خود را در حالت q_3 و انتقال تعریف نشده $\delta(q_3,0)$ می‌بیند. این وضعیت را پیکربندی مرده نامیده و آنرا بوسیله اتومانی نمایش می‌دهیم که بدون انجام هیچ حرکتی متوقف شده است. اما باید همواره به خاطر داشته باشیم که این نوع بررسی‌ها و استدلال‌ها چندان دقیق نبوده و ممکن است منجر به تفسیرهای نادرستی شود. با اطمینان می‌توان گفت که

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

یعنی، با پردازش $w=110$ نمی‌توان به هیچکدام ازحالت‌های پایانی دست یافت و به همین دلیل، رشته پذیرفته نمی‌شود.

■

لزوم بررسی نامعین‌بودن

هنگام استدلال در مورد ماشینهای نامعین، باید با احتیاط بیشتری از مفاهیم شهودی استفاده کنیم. بعلاوه، ممکن است شهود باعث گمراهی ما شده و به همین دلیل، باید برای اثبات نتایج خود از استدلال‌های دقیق استفاده کنیم. نامعین‌بودن یکی از مفاهیم پیچیده و مشکل است. کامپیوترهای دیجیتالی کاملاً معین بوده و حالت آنها در هر زمان منحصراً برحسب ورودی و حالت شروع تعیین می‌شود. بنابراین این سؤال مطرح می‌شود که چه لزومی به مطالعه ماشینهای نامعین وجود دارد؟ اگر قصد داریم از سیستم‌های واقعی مدل‌برداری کنیم، پس چرا باید روی این ویژگی‌های غیرمکانیکی تمرکز کنیم؟ پاسخ‌های مختلفی برای این پرسش وجود دارد.

در بسیاری از الگوریتم‌های معین، از قبیل برنامه بازی‌ها، حتماً در یکی از مراحل مسأله، با مدلی از انتخاب مواجه خواهیم شد. گرچه در اغلب موارد بهترین حرکت نامشخص است، اما می‌توان با جستجو در مسیرهای قبلاً پیموده شده، این حرکت را شناسایی کرد. در صورت وجود چند گزینه، یکی را انتخاب و تا زمانیکه برتری آن اثبات یا رد نشود همان را ادامهٔ می‌دهیم. در غیر اینصورت، به نقطه تصمیم‌گیری قبلی بازگشته و گزینه‌های دیگر را بررسی می‌کنیم. الگوریتم نامعین که بتواند بهترین گزینه را انتخاب نماید، قادر است تا بدون بازگشت به مسیر قبلی، حرکت مناسب را شناسایی کند؛ در حالیکه الگوریتم معین می‌تواند نامعین‌بودن را شبیه‌سازی کند. به همین دلیل، ماشین‌های نامعین را می‌توان بعنوان مدلی برای الگوریتم‌های جستجو و برگشت‌به‌عقب مطرح کرد.

همچنین، نامعین‌بودن گاهی اوقات در حل مسائل، بسیار مفید و مؤثر می‌باشد. با بررسی nfa شکل ۲-۸ دو انتخاب مطرح می‌شود. اولین انتخاب منجر به پذیرش رشته a^3 می‌شود، در حالیکه دومین انتخاب تمام رشته‌های دارای تعداد زوجی از a را می‌پذیرد. زبان پذیرفته شده بوسیله این nfa، $\{a^3\}\cup\{a^{2n}:n\geq 1\}$ می‌باشد. هرچند می‌توان یک dfa برای این زبان پیدا کرد، باز هم نامعین‌بودن به کلی نادیده گرفته نمی‌شود. زبان مذکور حاصل اجتماع دو مجموعه کاملاً متفاوت بوده و مفهوم نامعین‌بودن به ما امکان می‌دهد تا از همان ابتدا راجع به انتخاب مناسب تصمیم بگیریم. استفاده از دیدگاه معین ارتباط چندانی با تعریف ارائه شده نداشته و به همین دلیل استفاده از آن برای ارائه راه‌حل این زبان مفید نمی‌باشد. در ادامه، مثالهای دیگر و قانع‌کننده‌تری از کاربرد نامعین‌بودن ارائه خواهیم داد.

همچنین، نامعین‌بودن مکانیزم مفید و مؤثری برای تشریح دقیق برخی زبان‌های پیچیده است. توجه داشته باشید که در تعریف گرامر، یک عضو نامعین وجود دارد. در قانون

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

می‌توان در هر موقعیت یکی از قوانین اول و دوم را انتخاب کرد. به این ترتیب بسیاری از رشته‌ها فقط بوسیله دو قانون قابل تعیین هستند.

کلام آخر اینکه، یک دلیل فنی برای پرداختن به موضوع نامعین‌بودن وجود دارد. همانطور که بعداً هم خواهیم دید، اثبات برخی نتایج نظری برای nfaها بسیار راحت‌تر از dfaها می‌باشد. بعلاوه در ادامه بحث خواهیم دید هیچ تفاوت اساسی بین این دو نوع اتومات وجود ندارد. در نتیجه، مطرح کردن نامعین‌بودن در اغلب موارد باعث تسهیل روند استدلال‌های دقیق شده و در عین حال، هیچ تأثیری بر کلیت نتیجه‌گیری نمی‌گذارد.

تمرین‌ها

۱. در بخش قبل ادعا شد که اگر در یک تابع انتقال، گرافی با قدمی با برچسب w وجود داشته باشد، آنگاه بایستی یک قدم با برچسب w و طولی بزرگتر از $|w|$ وجود داشته باشد. این ادعا را ثابت کنید.

۲. یک dfa طراحی کنید که زبان تعریف شده بوسیله nfa شکل ۲-۸ را بپذیرد.

۳. یک dfa طراحی کنید که مکمل زبان تعریف شده بوسیله nfa شکل ۲-۸ را بپذیرد.

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

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

۶. به ازای nfa شکل ۲-۹، $\delta^*(q_0,1010)$ و $\delta^*(q_1,00)$ را بدست آورید.

۷. یک nfa با حداکثر پنج حالت برای مجموعه $\{abab^n:n\geq 0\}\cup\{aba^n:n\geq 0\}$ طراحی کنید.

۸. یک nfa سه حالتی بسازید که زبان $\{ab,abc\}^*$ را بپذیرد.

۹. آیا تمرین ۷ با کمتر از سه حالت هم قابل حل است؟

۱۰. الف) یک nfa با سه حالت طراحی کنید که زبان

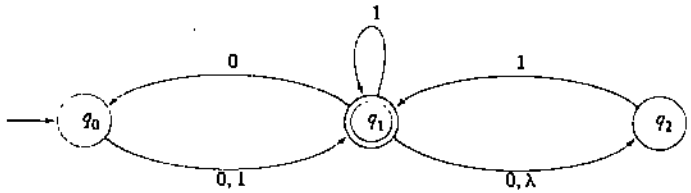
$$L=\{a^n:n\geq 1\}\cup\{b^m a^k:m\geq 0,k\geq 0\}$$

را پذیرش کند.

ب) آیا زبان مذکور در قسمت (الف) بوسیله nfa با کمتر از سه حالت نیز قابل پذیرش است؟

۱۱. nfa چهار حالتی را برای $L=\{a^n:n\geq 0\}\cup\{b^n a:n\geq 1\}$ طراحی کنید.

۱۲. کدامیک از رشته‌های 00، 01001، 10010، 000 و 0000 بوسیله nfa زیر قابل پذیرش است؟



۱۳. مکمل زبان پذیرفته شده بوسیله nfa شکل ۲-۱۰ چیست؟

۱۴. L را زبان پذیرفته شده بوسیله nfa شکل ۲-۸ در نظر می‌گیریم. nfa ای طراحی کنید که $L \cup \{a'\}$ را بپذیرد.

۱۵. در مورد زبان تمرین ۱۳ مختصراً توضیح دهید.

۱۶. یک nfa طراحی کنید که $\{a\}^*$ را پذیرفته و به صورتی باشد که چنانچه فقط یک یال از گراف انتقال آن حذف شود (بدون هیچ تغییر دیگری)، اتومات حاصل، $\{a\}$ را بپذیرد.

۱۷. آیا تمرین ۱۶ بوسیله یک dfa هم قابل حل است؟ در صورت مثبت بودن جواب، راه‌حل آنرا ارائه دهید. در غیراینصورت، دلایل قانع کننده‌ای برای نتیجه خود ارائه دهید.

۱۸. تعریف ۲-۶ را به صورت زیر بازنویسی می‌کنیم:

یک nfa با چندین حالت شروع بوسیله پنج‌تایی

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

تعریف می‌شود که در آن، $Q_0 \subseteq Q$ مجموعه تمام حالتهای شروع مجاز است. زبان پذیرفته شده بوسیله این اتومات به صورت

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

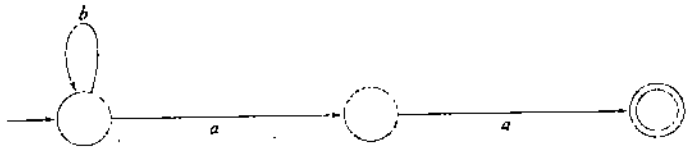
تعریف می‌شود. نشان دهید که به ازای هر nfa با چندین حالت شروع یک nfa با دقیقاً یک حالت شروع وجود دارد. که همان زبان را می‌پذیرد.

۱۹. در تمرین ۱۸ شرط $Q_0 \cap F = \emptyset$ را اعمال می‌کنیم. آیا این کار تغییری در نتیجه ایجاد می‌کند؟

۲۰. با استفاده از تعریف ۲-۵، نشان دهید که برای هر nfa، به ازای هر $q \in Q$ و هر $w, v \in \Sigma^*$،

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

۲۱. nfa که در آن (الف) هیچ انتقال λ وجود ندارد و (ب) به ازای هر $q \in Q$ و هر $a \in \Sigma$، $\delta(q,a)$ حاوی حداکثر یک عضو باشد، dfa ناقص نامیده می‌شود. بدلیل آنکه در این dfaها امکان انتخاب هر انتقالی وجود ندارد، و برای برخی انتقال‌ها نمی‌توان حرکتی کرد. این نامگذاری منطقی به نظر می‌رسد. به ازای $\Sigma = \{a, b\}$، dfa ناقص زیر را به یک dfa استاندارد تبدیل کنید.



۲-۷ هم‌ارزی پذیرنده‌های متناهی معین و نامعین

حال این سؤال اساسی را مطرح می‌کنیم که dfaها و nfaها چه تفاوتی با هم دارند؟ مشخص است که یکی از تفاوت‌ها مربوط به تعریف آنهاست. اما باید گفت که این تفاوت آنقدر مهم نیست که باعث

تمایز آنها شود. پس بهتر است برای یافتن پاسخ سؤال، مفهوم هم ارزی بین اتوماتا را مطرح و بررسی کنیم.

تعریف ۲-۷

دو پذیرنده متناهی M_1 و M_2 را در صورتی هم‌ارز می‌گوییم که

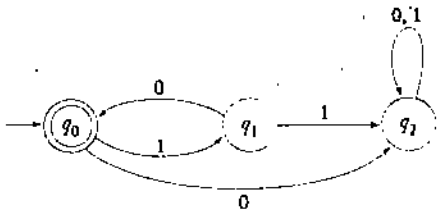
$$L(M_1) = L(M_2),$$

یعنی هر دو زبان یکسانی را بپذیرند.

همانطور که قبلاً هم گفته شد، به ازای هر زبان معمولاً تعداد زیادی پذیرنده وجود دارد؛ و بنابراین هر dfa یا nfa نیز تعداد زیادی پذیرنده هم‌ارز دارد.

مثال ۲-۱۱

dfa شکل ۲-۱۱ با nfa شکل ۲-۹ هم‌ارز است، چون هر دوپذیرنده، زبان $\{(10)^n : n \geq 0\}^*$ را می‌پذیرند.



شکل ۲-۱۱

■

وقتی انواع کلاس‌های مختلف اتوماتا را با هم مقایسه می‌کنیم، بدون شک این پرسش مطرح می‌شود که کدامیک از دیگری قدرتمندتر است؟ منظور از "قدرتمندتر" این است که اتوماتی از یک کلاس اعمالی را انجام می دهد که هیچ اتوماتی از کلاس دیگر قادر به انجام آن نمی باشد. حال بیایید این پرسش را در مورد پذیرنده‌های متناهی پاسخ دهیم. بدلیل آنکه dfa در واقع مدل محدودشده nfa است، بنابراین هر زبانی که بوسیله یک dfa پذیرفته شود، بوسیله nfa هم پذیرفته خواهد شد. اما عکس این مورد به ظاهر درست نمی‌باشد. با در نظر گرفتن مسأله نامعین‌بودن، شاید این شبهه مطرح شود که حداقل یک زبان وجود دارد که گرچه بوسیله یک nfa پذیرفته می‌شود، ولی در کل نمی‌توان یک dfa به ازای آن پیدا کرد. اما این تصور درست نیست و باید اظهار کرد که کلاس‌های dfaها و nfaها دارای قدرت یکسان می‌باشند و بنابراین، به ازای هر زبانی که بوسیله یک nfa پذیرفته می‌شود، یک dfa هم وجود دارد که آن زبان را می‌پذیرد.

نتیجه بدست آمده بدیهی به نظر نمی‌رسد، لذا پیش از پذیرفتن این نتیجه باید آنرا اثبات کرد. این استدلال هم، مانند دیگر استدلال‌های این کتاب، استنباطی خواهد بود. یعنی می‌توان به کمک آن درعمل روشی برای تبدیل هر nfa به dfa هم‌ارز آن پایه‌گذاری کرد. فهم ساختار پیچیده نبوده و به عنوان یک ایده مناسب می‌تواند شروع مناسبی برای یک برهان دقیق باشد. اساس ساختار استدلال به صورت زیر است:

پس از خوانده شدن رشته w توسط nfa، ممکن است دقیقاً ندانیم که nfa در چه وضعیتی قرار دارد؛ اما می‌توان حدس زد که در یکی از حالات مجاز، یعنی $\{q_1,q_2,\ldots,q_k\}$، قرار دارد. در حالی که dfa نظیر پس از خواندن همین رشته فقط در یک حالت قرار می‌گیرد. چگونه می‌توان این دو وضعیت را با هم متناظر کرد؟ پاسخ این است که، می‌توان حالت‌های dfa را با مجموعه‌ای از حالات به گونه‌ای برچسب‌دار کرد که، پس از خواندن w، dfa هم‌ارز فقط در یکی از حالات دارای برچسب $\{q_1,q_2,\ldots,q_k\}$ قرار بگیرد. بدلیل آنکه برای مجموعه‌ای از $|Q|$ حالت، دقیقاً $2^{|Q|}$ زیرمجموعه وجود دارد، بنابراین dfa هم‌ارز تعداد متناهی حالت خواهد داشت.

در این استدلال، تأکید اصلی بر روی آنالیز nfa، جهت ایجاد تناظر بین حالات مجاز و ورودی‌ها می‌باشد. پیش از شرح دقیق، ابتدا سعی می‌کنیم با ذکر یک مثال آنرا مشخص کنیم.

مثال ۲-۱۲

nfa شکل ۲-۱۲ را به dfa هم‌ارز آن تبدیل کنید.

چون nfa از حالت q_0 عملیات خود را آغاز می‌کند، حالت شروع dfa به صورت $\{q_0\}$ برچسب‌دار خواهد شد. nfa پس از خواندن یک a، در حالت q_1 و پس از انتقال λ، در حالت q_2 قرار خواهد گرفت. بنابراین، dfa متناظر باید یک حالت با برچسب $\{q_1,q_2\}$ و یک انتقال

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

داشته باشد. در حالت q_0 و با ورودی b، nfa هیچگونه انتقال خاصی ندارد؛ بنابراین،

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

حالت با برچسب $\varnothing$ بیانگر حرکت غیرمجاز nfa بوده و بنابراین، به معنای عدم پذیرش رشته می‌باشد. در نتیجه، این حالت در dfa یک حالت تله غیرپایانی است.

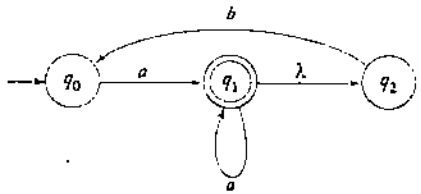
اینک که حالت $\{q_1,q_2\}$ را برای dfa تعریف کردیم، باید انتقالات از این حالت را نیز پیدا کنیم. به خاطر داشته باشید که این حالت dfa، متناظر با دو حالت احتمالی nfa است و بنابراین باید دوباره به سراغ nfa برویم. چنانچه nfa در حالت q_1 اقدام به خواندن یک a نماید، به q_1 منتقل می‌شود. بعلاوه، می‌تواند از q_1 یک انتقال λ به q_2 انجام دهد. اگر با همان ورودی، nfa در حالت q_2 قرار داشته باشد، آنگاه هیچ انتقال مشخصی وجود نخواهد داشت. بنابراین،

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

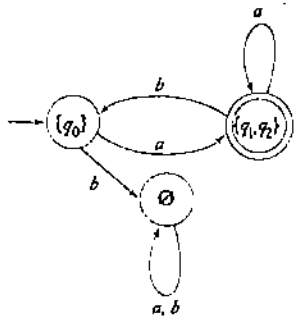
به همین ترتیب،

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

تا به اینجا موفق شدیم برای هر یک از حالات پذیرنده، تمام انتقال‌ها را تعریف کنیم. نتیجه‌ای که در شکل ۲-۱۳ مشاهده می‌کنید، dfa هم‌ارز با nfa می‌باشد. nfa شکل ۲-۱۲ تمام رشته‌هایی را می‌پذیرد که $\delta^*(q_0,w)$ آن حاوی q_1 باشد. برای آنکه dfa نظیر هم بتواند تمامی این wها را بپذیرد، تمام مجموعه حالت‌هایی که دارای q_1 می‌باشند، باید بعنوان حالت پایانی در نظر گرفته شوند.



شکل ۲-۱۲



شکل ۲-۱۳

■

قضیه ۲-۲

فرض کنید که L زبان پذیرفته شده توسط پذیرنده متناهی نامعین $M_N=(Q_N,\Sigma,\delta_N,q_0,F_N)$ باشد. آنگاه یک پذیرنده متناهی معین $M_D=(Q_D,\Sigma,\delta_D,\{q_0\},F_D)$ وجود خواهد داشت که

$$L=L(M_D).$$

اثبات: با داشتن M_N، می‌توان از روال *nfa به dfa* زیر برای ساخت گراف انتقال G_D مربوط به M_D استفاده کرد. برای درک بیشتر، بهتر است بدانیم که G_D باید ویژگی‌های خاصی داشته باشد. یعنی، هریک از رئوس بایستی دقیقاً $|\Sigma|$ یال خروجی داشته و توسط یکی از اعضای Σ نامگذاری شود.

هرچند ممکن است برخی از یالها در حین ساخت نادیده گرفته شوند، این روال را باید تا زمانی‌که همه یال‌ها ملاقات نشده، ادامه دهیم. ■

روال: nfa به dfa

۱. گراف مفروض G_D با رأس $\{q_0\}$ را ایجاد کرده و آن رأس را بعنوان رأس شروع در نظر بگیرید.
۲. تا زمانی که همهٔ یال‌ها در نظر گرفته نشده‌اند، مراحل زیر را تکرار کنید:
   - هریک از رئوس $\{q_i,q_j,\ldots,q_k\}$ از G_D را درنظر بگیرید که برای $a\in\Sigma$ آن هیچ یالی خارج نشود.
   - $\delta_N^*(q_i,a),\delta_N^*(q_j,a),\ldots,\delta_N^*(q_k,a)$ را محاسبه کنید.
   - اجتماع همه این δ_N^*ها را تشکیل دهید در نتیجه مجموعه $\{q_l,q_m,\ldots,q_n\}$ بدست می‌آید.

$$\delta_N^*(q_i,a)\cup\delta_N^*(q_j,a)\cup\ldots\cup\delta_N^*(q_k,a)=\{q_l,q_m,\ldots,q_n\},$$

   - در صورت عدم وجود رأسی با برچسب $\{q_l,q_m,\ldots,q_n\}$ در G_D، این راس را ایجاد کنید. یالی از $\{q_i,q_j,\ldots,q_k\}$ به $\{q_l,q_m,\ldots,q_n\}$ به نام a به G_D اضافه کنید.
۳. تمامی حالت‌های G_D که برچسب آن حاوی حداقل یک $q_f\in F_N$ باشد، بعنوان رأس پایانی شناخته می‌شود.
۴. اگر M_N، λ را بپذیرد، رأس $\{q_0\}$ در G_D نیز، رأس پایانی می‌باشد.

مشخص است که همواره این روال خاتمه پذیر است. با هر بار عبور از گره مرحله دو، یک یال به G_D اضافه می‌شود. اما G_D حداکثر $2^{|Q_N|}|\Sigma|$ یال داشته و به همین دلیل، حلقه تکرار فوق بلاخره متوقف می‌شود. برای اثبات درستی این الگوریتم ساخت، براساس استقراء در مورد طول رشته ورودی بحث می‌کنیم.

فرض کنید که به ازای هر w با طول کمتریا مساوی n، وجود یک قدم در G_N با برچسب w از q_0 به q_i به طور ضمنی بیانگر وجود یک قدم با برچسب w از $\{q_0\}$ به حالت $Q_i=\{\ldots,q_i,\ldots\}$ باشد. حال، تمام $w=va$ را در نظر گرفته و قدم مربوط به آنها در G_N با برچسب w از q_0 به q_i را بررسی می‌کنیم. بنابراین، باید یک قدم با برچسب v از q_0 به q_i و یک یال (یا توالی از یالها) با برچسب a از q_i به q_j وجود داشته باشد. براساس فرض استقراء، در G_D یک قدم با برچسب v از $\{q_0\}$ به Q_i وجود خواهد داشت. در حالی که براساس ساختار، یک یال از Q_i به یکی از حالت‌های دارای برچسب q_j وجود دارد. بنابراین، فرض استقراء به ازای تمام رشته‌های با طول $n+1$ صحیح است. بدلیل آنکه این فرض برای $n=1$ درست است، بنابراین برای تمام nها هم درست خواهد بود. در نتیجه، هرگاه که $\delta_N^*(q_0,w)$ حاوی حالت پایانی q_f باشد، حاوی برچسب $\delta_D^*(q_0,w)$ نیز خواهد بود. برای تکمیل اثبات، با معکوس کردن استدلال نشان می‌دهیم که برچسب $\delta_D^*(q_0,w)$ حاوی q_f و بنابراین، حاوی برچسب $\delta_N^*(q_0,w)$ می‌باشد. ■

استدلال‌های بکار رفته در این اثبات، تا حدودی قابل قبول بوده و فقط مراحل مهم اثبات را نشان می‌دهد. در ادامه کتاب همین روند را دنبال می‌کنیم؛ یعنی بر روی ایده‌های اصلی در اثبات تأکید کرده و با نادیده گرفتن جزئیات کم‌اهمیت‌تر، کشف آنها را به عهده خواننده و ذهن پویای او می‌گذاریم.

ساختار اثبات فوق علیرغم یکنواختی و خسته‌کنندگی، حائز اهمیت است. بیایید با بررسی یک مثال دیگر، مراحل بالا را مجدداً مطرح و از عملکرد درست آنها مطمئن شویم.

مثال ۲-۱۳

nfa شکل ۲-۱۴ را به dfa هم ارز با آن تبدیل کنید.

بدلیل آنکه $\delta_N(q_0,0)=\{q_0,q_1\}$، حالت $\{q_0,q_1\}$ در G_D را معرفی کرده و یک یال با برچسب 0 از $\{q_0\}$ به $\{q_0,q_1\}$ اضافه می‌کنیم. به همین ترتیب و با درنظرگرفتن $\delta_N(q_0,1)=\{q_1\}$، حالت جدید $\{q_1\}$ و یک یال با برچسب 1 از $\{q_1\}$ به $\{q_0\}$ بوجود می‌آید.

تعدادی از یالها هنوز وجود ندارند. و بنابراین برای ادامهٔ مراحل تبدیل از ساختار قضیه ۲-۲ استفاده می‌کنیم. با نگاهی به حالت $\{q_0,q_1\}$ متوجه می‌شویم که هیچ یال خروجی با برچسب 0 وجود ندارد. بنابراین

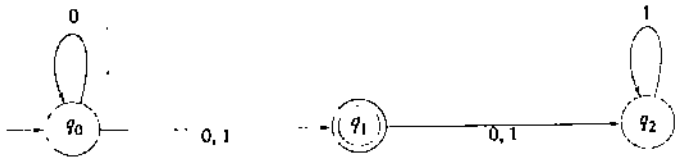
$$\delta_N^*(q_0,0)\cup\delta_N^*(q_1,0)=\{q_0,q_1,q_2\}$$

را محاسبه می‌کنیم. به این ترتیب، حالت جدید $\{q_0,q_1,q_2\}$ و انتقال

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

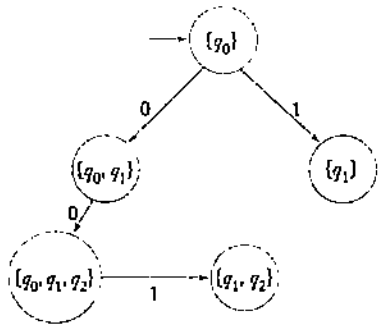
بوجود می‌آید. سپس با استفاده از $a=1$، $i=0$، $j=1$ و $k=2$

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

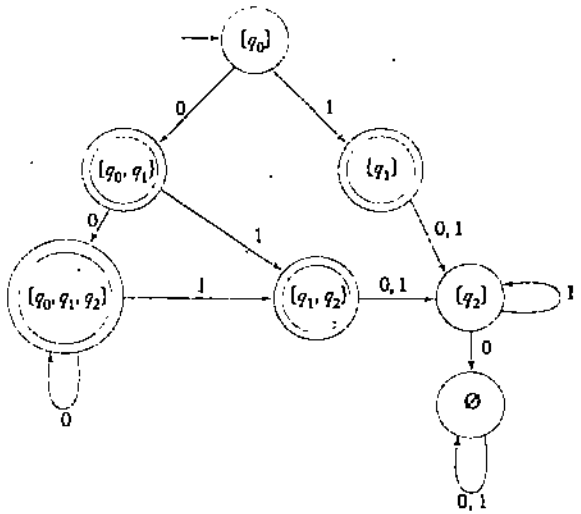


شکل ۲-۱۴

حالت دیگر $\{q_1,q_2\}$ بدست می‌آید. تا اینجا، تا حدودی اتومات شکل ۲-۱۵ کامل شده است. به این دلیل که هنوز تعدادی یال در شکل ۲-۱۵ وجود ندارند، این کار را تا بدست آوردن dfa کامل موجود در شکل ۲-۱۶ ادامه می‌دهیم.



شکل ۲-۱۵



شکل ۲-۱۶

■

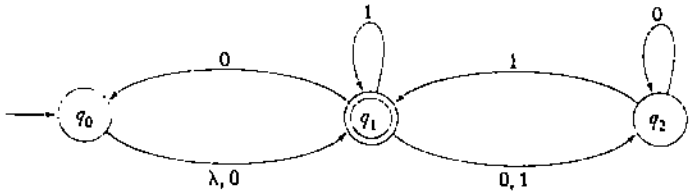
یکی از نتایج مهم قضیه ۲-۲ این است که تمام زبان‌های پذیرفته شده توسط nfa منظم هستند.

تمرین‌ها

۱. با استفاده از ساختار قضیه ۲-۲، nfa شکل ۲-۱۰ را به یک dfa تبدیل کنید. آیا پاسخ ساده‌تر و دقیق‌تری برای این سؤال وجود دارد؟

۲. nfa تمرین ۱۲ بخش ۲-۲ را به dfa هم‌ارز آن تبدیل کنید.

۳. nfa زیر را به dfa هم‌ارز آن تبدیل کنید:



۴. استدلال‌های مربوط به اثبات قضیه ۲-۲ را تکمیل کنید. نشان دهید که اگر برچسب $\delta_n^*(q_0,w)$ حاوی q_f باشد، آنگاه $\delta_n^*(q_0,w)$ نیز حاوی q_f خواهد بود.

۵. درستی یا نادرستی این عبارت را ثابت کنید:
به ازای هر nfa بصورت $M=(Q,\Sigma,\delta,q_0,F)$، مکمل $L(M)$ برابر با مجموعه $\{w\in\Sigma^*:\delta^*(q_0,w)\cap F=\emptyset\}$ خواهد بود. در صورت درستی، آنرا اثبات و در غیر اینصورت، یک مثال نقض برای ردکردن آن ذکر کنید.

۶. درستی یا نادرستی این عبارت را ثابت کنید: به ازای هر nfa بصورت $M=(Q,\Sigma,\delta,q_0,F)$، مکمل $L(M)$ برابر با مجموعه $\{w\in\Sigma^*:\delta^*(q_0,w)\cap(Q-F)\neq\emptyset\}$ می‌باشد. در صورت درستی، آنرا اثبات و در غیر اینصورت، یک مثال نقض در رد آن ذکر کنید.

۷. اثبات کنید که برای هر nfa با هر تعداد دلخواه حالت پایانی، یک dfa با فقط یک حالت پایانی، هم‌ارز با آن nfa وجود دارد. آیا می‌توان برای dfaها هم این ادعا را مطرح کرد؟

۸. یک nfa بدون انتقال λ و فقط با یک حالت پایانی پیدا کنید که مجموعه $\{a\}\cup\{b^n:n\geq 1\}$ را بپذیرد.

۹. L را یک زبان منظم فاقد λ فرض می‌کنیم. نشان دهید یک nfa بدون انتقال λ و فقط با یک حالت پایانی وجود دارد که L را می‌پذیرد.

۱۰. به روشی مشابه nfa هم ارز تمرین ۱۸ بخش ۲-۲، یک dfa با چندین حالت شروع را فرض کنید. آیا همواره یک dfa هم ارز آن با فقط یک حالت شروع وجود دارد؟

۱۱. اثبات کنید که تمام زبانهای متناهی، منظم هستند.

۱۲. نشان دهید که اگر L منظم باشد، L^R هم منظم خواهد بود.

۱۳. به طور مختصر زبان پذیرفته شده بوسیله dfa شکل ۲-۱۶ را شرح دهید. براساس این شرح، dfa دیگری هم‌ارز با dfa داده شده اما با حالت‌های کمتر ارائه دهید.

۱۴. $L-L^*$ یک زبان مفروض است. $even(w)$ را رشته ای تعریف می‌کنیم که از استخراج حروف واقع در موقعیتهای زوج w بدست می‌آید؛ یعنی اگر

$$w=a_1a_2a_3a_4\ldots,$$

آنگاه،

$$even(w)=a_2a_4\ldots.$$

متناظر با آن می‌توانیم زبان

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

را تعریف می‌کنیم. اثبات کنید که اگر L منظم باشد، آنگاه $even(L)$ منظم خواهد بود.

۱۵. با حذف دو سمبل انتهایی از سمت چپ هر رشته‌ای از زبان L زبان جدیدی به نام $chop2(L)$ تعریف می‌کنیم. فرم کلی تعریف آن بصورت،

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

می‌باشد. نشان دهید که اگر L منظم باشد، آنگاه $chop2(L)$ منظم خواهد بود.

کاهش تعداد حالات در اتوماتای متناهی*

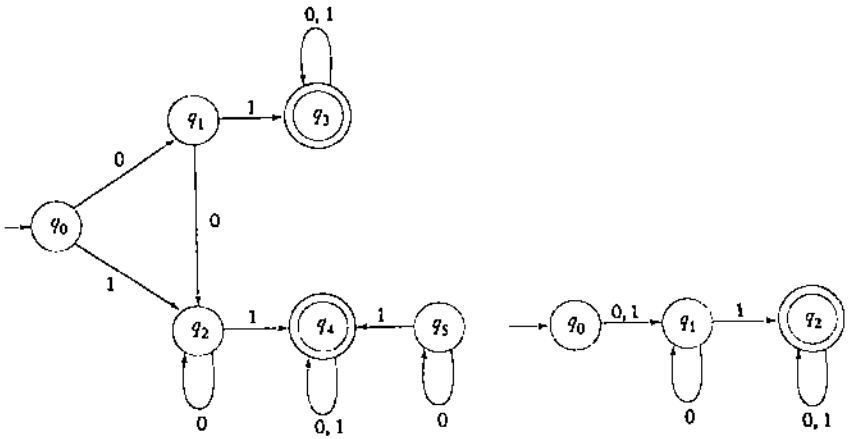
هر dfa یک زبان منحصربفرد را تعریف می‌کند، اما عکس این جمله صحیح نیست. هر زبان توسط تعداد زیادی dfa پذیرفته می‌شود. تعداد حالت‌های این گروه اتوماتاهای هم ارز بسیار با هم متفاوت است. در سؤالاتی که تا به اینجا بررسی کردیم، تمام dfaها برای یک زبان به یک اندازه درست بودند. اما اگر قرار باشد نتایج را در عمل استفاده کنیم، ممکن است به دلایلی یکی بر دیگری ترجیح داده شود.

مثال ۲-۱۴

با آزمایش چند رشته به راحتی می‌توان به هم ارزی dfa شکل ۲-۱۷ (الف) و ۲-۱۷ (ب) پی برد. برخی از ویژگی‌های غیرلازم شکل ۲-۱۷ (الف) را بررسی می‌کنیم. چون هرگز نمی‌توان از حالت شروع q_0 به حالت q_3 دست یافت، این حالت تقریباً هیچ نقشی در اتومات ایفا نمی‌کند. همچنین به دلیل غیرقابل‌دسترس‌بودن این حالت، می‌توان به راحتی و بدون هیچ تأثیری بر زبان پذیرفته شده بوسیله اتومات، اقدام به حذف آن و تمام انتقال‌های مربوطه نمود. اما حتی پس از حذف q_3، باز هم برخی بخش‌ها در اتومات زائد خواهند بود. حالت‌های قابل‌دسترس پس از اولین حرکت $\delta(q_0,0)$ عیناً حالت‌های قابل‌دسترس از اولین حرکت $\delta(q_0,1)$ هستند. در اتومات دوم این دو روش با هم ترکیب شده‌اند.

■

از نقطه نظر تئوری، اتومات ۲-۱۷ (الف) بر اتومات ۲-۱۷ (ب) ترجیح دارد. در حالی که اتومات اخیر به دلیل سهولت بر اتومات اول ارجح‌تر است. ارائه یک اتومات برای امور محاسباتی مستلزم وجود فضایی متناسب با تعداد حالت‌ها است. جهت بالا بردن کارآیی حافظه، می‌توان تعداد حالت‌ها را تا حد امکان کاهش داد. در زیر الگوریتمی برای انجام این کار پیشنهاد شده است.



(الف) (ب)

شکل ۲-۱۷

تعریف ۲-۸

دو حالت p و q یک dfa، در صورتی **ادغام‌پذیر** هستند که به ازای هر $w\in\Sigma^*$،

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

و

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

از طرف دیگر، اگر رشته $w\in\Sigma^*$ وجود داشته باشد که

$$\delta^*(p,w)\in F\ \text{ و }\ \delta^*(q,w)\notin F,$$

یا برعکس، آنگاه حالت‌های p و q **ادغام‌ناپذیر** براساس رشتهٔ w نامیده می‌شوند.

مشخص است که هر دو حالت مفروض یا ادغام‌ناپذیر هستند و یا ادغام‌پذیر. ادغام‌پذیر بودن خواص روابط هم‌ارزی را بکار می‌گیرد، یعنی اگر p و q ادغام‌پذیر بوده و q و r هم ادغام‌پذیر باشند، آنگاه p و r ادغام‌پذیر هستند و در نتیجه هر سه حالت هم ادغام‌پذیر هستند.

یک روش برای کاهش حالت‌های dfa، یافتن و ترکیب حالت‌های ادغام‌پذیر است. اما ابتدا روشی را برای پیدا کردن حالت‌های ادغام‌ناپذیر ارائه می‌کنیم.

روال: علامت‌گذاری

۱. همه حالت‌های غیرقابل‌دسترس را حذف کنید. این کار از طریق طی کردن تمام مسیرهای ساده گراف dfa با آغاز از حالت شروع قابل انجام است. به این ترتیب هر حالتی که جزء این مسیرها نباشد، غیرقابل‌دسترس است.

۲. تمام زوج حالت‌های (p,q) را در نظر بگیرید. اگر $p \in F$ و $q \notin F$ یا برعکس، زوج (p,q) را بعنوان ادغام‌ناپذیر نشانه‌گذاری کنید.

۳. مرحله زیر را آنقدر تکرار کنید تا زوجی برای تصمیم گیری باقی نماند.

به ازای تمام زوج‌های (p,q) و همه $a \in \Sigma$، $\delta(p,a) = p_a$ و $\delta(q,a) = q_a$ را محاسبه کنید. اگر زوج (p_a,q_a) به عنوان ادغام‌ناپذیر بدست آمده باشد، (p,q) را هم بعنوان ادغام‌ناپذیر علامت می‌زنیم.

اینک ادعا می‌کنیم که این روال باعث ایجاد الگوریتمی برای بدست آوردن تمامی زوج‌های (p,q) ادغام‌ناپذیر می‌شود.

قضیه ۲-۳

روال علامت‌گذاری فوق، برای هر پذیرنده متناهی معین $M = (Q,\Sigma,\delta,q_0,F)$ قابل بکارگیری است و تمامی زوج حالت‌های ادغام‌ناپذیر را بدست آورده و متوقف می‌شود.

اثبات: مشخص است که به دلیل متناهی بودن تعداد زوجهای قابل علامت‌گذاری، این روال متوقف می‌شود. می‌توان مشاهده کرد که حالت‌های زوج‌هایی که به این صورت علامت گذاری می‌شوند، ادغام‌ناپذیر هستند. ادعای ما تنها نیازمند بسط و توضیح این است که، این روال همه زوج‌های ادغام‌ناپذیر را پیدا می‌کند.

توجه کنید حالت‌های q_i و q_j ادغام‌ناپذیر هستند (با یک رشته به طول n) اگر و تنها اگر انتقال‌های

$$\delta(q_i,a) = q_k \qquad (۲-۵)$$

و همچنین

$$\delta(q_j,a) = q_l, \qquad (۲-۶)$$

برای $a \in \Sigma$ بدست آید و q_k و q_l با یک رشته به طول $n-1$ ادغام‌ناپذیر باشند.

اینک با استفاده از دو تساوی بالا نشان می‌دهیم که در پایان n امین تکرار مرحله ۳، همه حالت‌های ادغام‌ناپذیر بوسیله رشته‌هایی با طول n یا کمتر مشخص می‌شوند. در مرحله ۲، همه زوج‌های ادغام‌پذیر بوسیله λ را مشخص کرده و به این ترتیب، پایه‌ای به صورت $n=0$ برای استقراء بوجود می‌آوریم. حال فرض می‌کنیم که این ادعا برای همه $i = 0,1,\ldots,n-1$ برقرار باشد. با در نظر گرفتن این فرض استقراء، در آغاز n امین تکرار، همه حالت‌های ادغام‌ناپذیر بوسیله رشته‌هایی با طول حداکثر n-1 بدست می‌آیند. براساس (۲-۵) و (۲-۶)، در پایان این تکرار همه حالت‌های ادغام‌ناپذیر رشته‌های با حداکثر طول n بدست می‌آیند. بنابراین براساس استقراء می‌توان ادعا کرد که به ازای هر n، در پایان n امین تکرار، همه زوج‌های ادغام‌ناپذیر بوسیله رشته‌های با طول n یا کمتر بدست می‌آیند.

برای اثبات اینکه تمام حالت‌های ادغام‌ناپذیر، بوسیله این روال بدست می‌آیند، فرض می‌کنیم که حلقه پس از n تکرار متوقف شود. بنابراین، طی n امین تکرار، هیچ حالت جدیدی بدست نمی‌آیند. اما براساس (۲-۵) و (۲-۶)، می‌توان نتیجه گرفت که هیچ‌یک از حالت‌ها بوسیله رشته‌ای با طول n یا کمتر ادغام‌ناپذیر نخواهد شد. اما اگر هیچ حالتی فقط بوسیله رشته‌های با طول n ادغام نشود، هیچ حالت ادغام‌ناپذیری فقط بوسیله رشته‌هایی با طول $n+1$ وجود نخواهد داشت و الی آخر. در نتیجه، وقتی حلقه متوقف می‌شود، همه زوج‌های ادغام‌ناپذیر بدست می‌آیند. ■

برای پیاده‌سازی روال علامت‌گذاری، حالات را به کلاس‌های هم‌ارزی تقسیم می‌کنیم. در این روش، به محض شناسایی دو حالت ادغام‌ناپذیر، آنها را به کلاس‌های هم‌ارزی جداگانه تقسیم می‌کنیم.

مثال ۲-۱۵

اتومات شکل ۲-۱۸ را در نظر بگیرید.

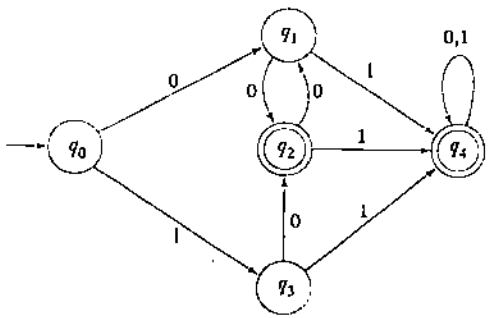
در مرحله دوم روال علامت‌گذاری، با تقسیم مجموعه حالات به حالت‌های پایانی و غیرپایانی، دو کلاس هم‌ارزی $\{q_0,q_1,q_3\}$ و $\{q_2,q_4\}$ بوجود می‌آوریم. در مرحله بعد، با محاسبه

$$\delta(q_0,0) = q_1$$

و

$$\delta(q_1,0) = q_2,$$

مشاهده می‌کنیم که q_0 و q_1 ادغام‌ناپذیر بوده و به همین دلیل، آنها را در مجموعه‌های جداگانه‌ای قرار می‌دهیم. بنابراین $\{q_0,q_1,q_3\}$ به $\{q_0\}$ و $\{q_1,q_3\}$ تجزیه می‌شود. بدلیل آنکه $\delta(q_2,0) = q_3$ و $\delta(q_1,0) = q_4$، کلاس $\{q_2,q_4\}$ به $\{q_2\}$ و $\{q_4\}$ تجزیه می‌شود. در ادامه کار دیگر نیازی به تجزیه بیشتر وجود ندارد.



شکل ۲-۱۸

■

پس از یافتن کلاسهای ادغام‌ناپذیر، ساخت dfa کمینه دیگر چندان مشکل نخواهد بود.

روال: کاهش

با داشتن پذیرندهٔ متناهی‌معین $M=(Q,\Sigma,\delta,q_0,F)$، به ترتیب زیر اقدام به ساخت dfa کاهش‌یافته $\hat{M}=(\hat{Q},\hat{\Sigma},\hat{\delta},\hat{q}_0,\hat{F})$ می‌کنیم:

۱. با استفاده از روال علامت‌گذاری و به روشی که توضیح داده شد، کلاس‌های هم‌ارزی از قبیل $\{q_i,q_j,\ldots,q_k\}$ ایجاد می‌کنیم.

۲. به ازای هر مجموعه $\{q_i,q_j,\ldots,q_k\}$ از حالات ادغام‌پذیر، حالتی با برچسب $ij\ldots k$ برای M ایجاد می‌کنیم.

۳. به ازای هر یک از قانون‌های انتقال موجود در M به فرم

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

مجموعه‌هایی را پیدا می‌کنیم که q_r و q_p به آنها متعلق باشند. اگر $q_r\in\{q_i,q_j,\ldots,q_k\}$ و $q_p\in\{q_l,q_m,\ldots,q_n\}$ باشند، آنگاه قانون

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

را به $\hat{\delta}$ اضافه می‌کنیم.

۴. حالت شروع $\hat{q}_0$ حالتی از $\hat{M}$ است که در برچسب آن 0 وجود داشته باشد.

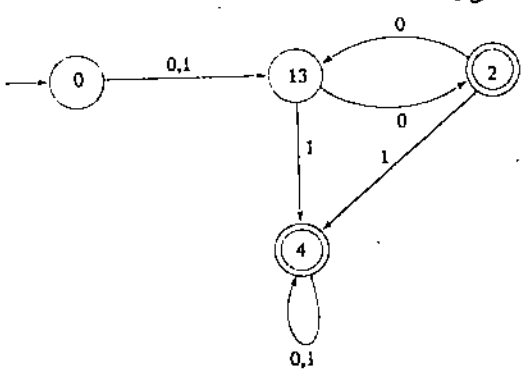
۵. $\hat{F}$ مجموعه تمام حالت‌هایی است که برچسب آنها حاوی i به صورتی باشد که $q_i\in F$

مثال ۲-۱۶

با ادامه مثال ۲-۱۵، حالت‌های مربوط به شکل ۲-۱۹ را ایجاد می‌کنیم. از آنجایی‌که بعنوان مثال

$$\delta(q_1,0)=q_2,$$

یک یال با برچسب 0 از حالت 13 به حالت 2 وجود دارد. بقیه انتقالات، با داشتن dfa کمینه در شکل ۲-۱۹، به راحتی بدست می‌آیند.



شکل ۲-۱۹

■

قضیه ۲-۴

با داشتن dfa مفروض M، بکارگیری روال *کاهش* باعث ایجاد dfa دیگر $\hat{M}$ به گونه‌ای می‌شود که

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

بعلاوه، بدلیل آنکه هیچ dfa دیگری با تعداد حالت‌های کمتر وجود ندارد که آن نیز L(M) را بپذیرد، $\hat{M}$ کمینه است.

اثبات: اثبات در دو قسمت انجام می‌شود. در قسمت اول نشان می‌دهیم که dfa تولید شده بوسیله روال *کاهش*، هم‌ارز با dfa اصلی است. این کار تقریباً به راحتی و به کمک استدلال‌های استقرایی مشابه استدلال‌های صورت گرفته برای ایجاد هم‌ارزهای dfa و nfa انجام می‌شود. فقط لازم است نشان دهیم که $\delta^*(q_i,w)=q_j$ اگر و تنها اگر برچسب $\hat{\delta}^*(q_i,w)$ به فرم $\ldots j\ldots$ باشد. این اثبات را بعنوان تمرین به عهده خودتان واگذار می‌کنیم.

در قسمت دوم، یعنی اثبات کمینه بودن $\hat{M}$، تا حدودی مشکل‌تر است. فرض کنید که $\hat{M}$ دارای حالت‌های $\{p_0,p_1,p_2,\ldots,p_m\}$ و حالت شروع p_0 بوده و بعلاوه، یک dfa ی M_1، هم‌ارز با تابع انتقال δ_1 و حالت شروع p_0 متناظر با $\hat{M}$ اما با حالت‌های کمتر وجود داشته باشد. به دلیل عدم وجود حالت‌های قابل دسترس در $\hat{M}$، باید رشته‌های جداگانه $w_1,w_2,\ldots,w_m$ وجود داشته باشند که

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

اما چون تعداد حالت‌های M_1 کمتر از $\hat{M}$ است، حداقل باید دو رشته مثل w_k و w_i وجود داشته باشند که

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

از طرفی، به دلیل ادغام‌ناپذیربودن p_k و p_i، بایستی رشته x وجود داشته باشد که با آن، $\hat{\delta}^*(p_0,w_kx)=\hat{\delta}^*(p_k,x)$ حالت پایانی و $\hat{\delta}^*(p_0,w_ix)=\hat{\delta}^*(p_i,x)$ یک حالت غیرپایانی باشد (یا برعکس). به بیان دیگر، w_kx بوسیله $\hat{M}$ پذیرفته شده، ولی w_ix پذیرفته نمی‌شود. اما چون

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

بنابراین، M_1 یا هر دو رشته w_kx و w_ix را می‌پذیرد و یا هر دو پذیرفته نمی‌شود. که این با فرض هم‌ارزی $\hat{M}$ و M_1 تناقض دارد. همین تناقض اثبات می‌کند که M_1 نمی‌تواند وجود داشته باشد. ■

تمرین‌ها

۱. تعداد حالت‌های dfa شکل ۲-۱۶ را کمینه کنید.

۲. dfaهای کمینه‌ای برای زبان‌های زیر پیدا کنید. در هر مورد کمینه بودن نتیجه را اثبات کنید.

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

ب) $L=\{a^n b : n \geq 0\} \cup \{b^n a : n \geq 1\}$.

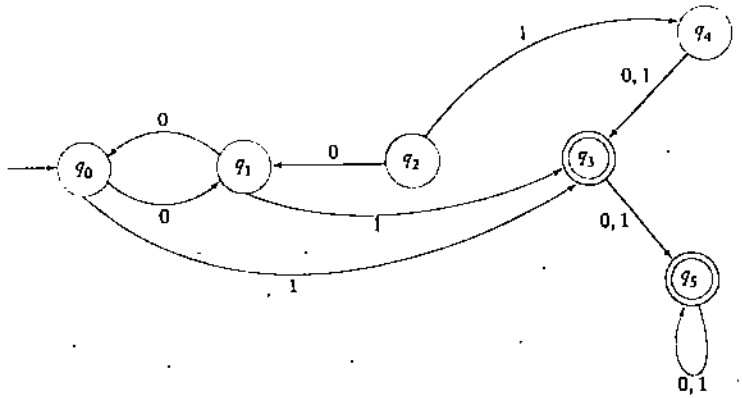
ج) $L=\{a^n : n \geq 0, n \neq 3\}$. ⊖

د) $L=\{a^n : n \neq 2 \text{ و } n \neq 4\}$.

ه) $L=\{a^n : n \bmod 3 = 0\} \cup \{a^n : n \bmod 5 = 1\}$.

۳. نشان دهید که اتومات ایجاد شده بوسیله روال کاهش، معین است.

۴. حالت‌های dfa درگراف انتقال حالت زیر را کمینه کنید.



۵. نشان دهید که اگر L زبان غیرتهی باشد بطوریکه هر w عضو L دارای حداقل طول n باشد، آنگاه هر dfa که L را بپذیرد، باید حداقل $n+1$ حالت داشته باشد.

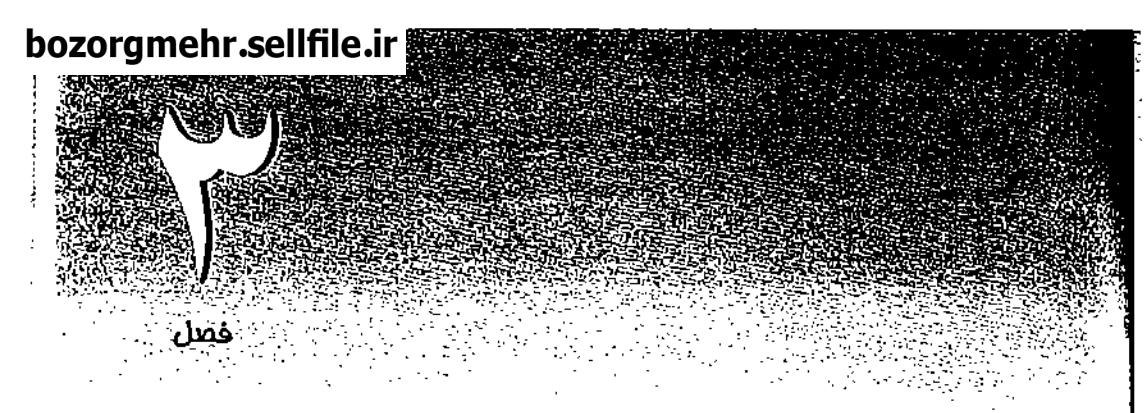
۶. درستی یا نادرستی این عبارت را ثابت کنید:
اگر $M=(Q,\Sigma,\delta,q_0,F)$ یک dfa کمینه برای زبان منظم L باشد، آنگاه $\hat{M}=(Q,\Sigma,\delta,q_0,Q-F)$ یک dfa کمینه برای $\overline{L}$ خواهد بود. ⊖

۷. نشان دهید که ادغام‌پذیربودن، یک رابطه هم‌ارزی است، ولی ادغام‌ناپذیربودن، اینطور نیست.

۸. مراحل آشکار اثبات پیشنهادی برای بخش اول قضیه ۲-۴ را نشان دهید؛ یعنی نشان دهید که $\hat{M}$ هم‌ارز با dfa اولیه است.

۹. **_ برنامه کامپیوتری بنویسید که یک dfa کمینه برای هرdfa مفروض تولید کند.

۱۰. درستی یا نادرستی این ادعا را اثبات کنید:
اگر حالتهای q_a و q_b ادغام‌پذیر باشند و q_a و q_c هم ادغام‌ناپذیر باشند، آنگاه q_b و q_c حتماً ادغام‌ناپذیر خواهند بود. ⊖



فصل

۳

زبان‌های منظم و گرامرهای منظم

براساس تعریفی که در فصل قبل ارائه کردیم، یک زبان در صورتی منظم است که پذیرنده متناهی برای آن وجود داشته باشد. بنابراین، هر زبان منظمی را می‌توان بوسیله یک dfa یا nfa تعریف کرد. بطور نمونه از این توصیف بسیار مفید و کارآمد می‌توان برای نمایش منطق تصمیم‌گیری در مورد وجود یا عدم وجود رشته‌ای مفروض، در یک زبان خاص استفاده کرد. اما در بسیاری از موارد، این توصیف کافی نبوده و باید روشهای دقیق‌تری برای توصیف زبان‌های منظم در اختیار داشته باشیم. در این فصل، به برخی دیگر از روشهای ارائه زبان‌های منظم نگاهی خواهیم داشت. این نوع ارائه‌ها کاربردهای عملی مهمی دارند که به برخی از آنها در قالب مثال و تمرین اشاره خواهیم کرد.

۳-۱ عبارات منظم

یک روش برای توصیف زبانهای منظم، استفاده از مجموعه سمبل‌های عبارات منظم است. این مجموعه سمبل‌ها شامل ترکیبی از سمبل‌ها، از قبیل الفبای Σ، پرانتزها و عملگرهای $+$، . و * می‌باشند. ساده‌ترین مثال، زبان $\{a\}$ است که بوسیله عبارت منظم a به آن اشاره می‌شود. نمونه پیچیده‌تر زبان $\{a,b,c\}$ است که با استفاده از $+$ برای نمایش اجتماع در آن، عبارت منظم $a+b+c$ ایجاد می‌شود.

به همین ترتیب، می‌توان از . برای اشاره به الحاق و از * جهت بستار ستاره‌ای استفاده کرد. بنابراین، عبارت $(a+(b.c))^*$ به معنای بستار ستاره‌ای $\{a\} \cup \{bc\}$، یعنی زبان $\{\lambda,a,bc,aa,abc,bca,bcbc,aaa,aabc,\ldots\}$، است.

تعریف صوری یک عبارت منظم

عبارات منظم با انجام متوالی برخی قوانین بازگشتی روی اجزاء پایه‌ای و به روشی مشابه ساخت عبارات ریاضی ایجاد می‌شوند.

تعریف ۳-۱

Σ را الفبای مفروض در نظر می‌گیریم. آنگاه

۱. λ، $\varnothing$ و $a \in \Sigma$ همگی عبارات منظم هستند. این عبارات را اصطلاحاً عبارات منظم پایه می‌خوانیم.

۲. اگر r_1 و r_2 عبارات منظمی باشند، آنگاه $r_1 + r_2$، $r_1 . r_2$، r_1^* و (r_1) نیز عبارات منظم خواهند بود.

۳. یک رشته فقط و فقط در صورتی عبارت منظم است که با بکارگیری تعداد محدودی از قوانین بند (۲)، از عبارات منظم پایه بدست آیند.

مثال ۳-۱

برای الفبای $\Sigma = \{a,b,c\}$، رشته

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

به دلیل استفاده از قوانین فوق در ساخت آن، عبارت منظم است. بعنوان مثال، اگر $r_1 = c$ و $r_2 = \varnothing$ را در نظر بگیریم، $c+\varnothing$ و $(c+\varnothing)$ عبارات منظم خواهند بود. با ادامه همین روند کل رشته ساخته می‌شود. از طرف دیگر، $(a+b+$ عبارت منظم نیست، چون به هیچ طریقی با استفاده از تعریف قبل، آن را نمی‌توان بدست آورد.

■

زبان‌های مرتبط با عبارات منظم

از عبارات منظم می‌توان برای توصیف برخی زبانهای ساده استفاده نمود. اگر r یک عبارت منظم باشد، زبان مرتبط با آنرا با $L(r)$ نمایش می‌دهیم.

تعریف ۳-۱

زبان $L(r)$ مربوط به عبارت منظم r، با قوانین زیر تعریف می‌شود:

۱. $\varnothing$ عبارت منظم است که به مجموعه تهی دلالت می‌کند،

۲. λ یک عبارت منظم مربوط به $\{\lambda\}$ است،

۳. به ازای هر $a \in \Sigma$، a عبارت منظم مربوط به $\{a\}$ 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. $L(r_1 . r_2) = L(r_1)\, L(r_2)$

۶. $L((r_1)) = L(r_1)$

۷. $L(r_1^*) = (L(r_1))^*$

از چهار قانون آخر تعریف فوق برای تجزیه $L(r)$ به اجزاء کوچکتر بطور بازگشتی استفاده می‌شود؛ در حالیکه سه قانون اول، شرایط پایان این بازگشت هستند. برای اینکه بدانیم یک عبارت مفروض به چه زبانی اشاره می‌کند، این قوانین را بطور متوالی در عبارت مورد نظر اعمال می‌کنیم.

مثال ۳-۲

زبان $L(a^* . (a+b))$ را برحسب مجموعه رشته‌ها نمایش دهید.

$$\begin{aligned} L(a^* . (a+b)) &= L(a^*)\, L(a+b) \\ &= (L(a))^* (L(a) \cup L(b)) \\ &= \{\lambda, a, aa, aaa, \ldots\} \{a,b\} \\ &= \{a, aa, aaa, \ldots, b, ab, aab, \ldots\}. \end{aligned}$$

■

اما درقوانین (۴) تا (۷) تعریف ۳-۲ یک مشکل وجود دارد:

این قوانین فقط در صورتی قادر به تعریف دقیق یک زبان هستند که r_1 و r_2 معلوم باشند؛ در حالیکه تجزیه عبارات پیچیده به بخش‌های کوچکتر و ساخت r_1 و r_2 ممکن است ابهاماتی ایجاد کند. بعنوان مثال، عبارت منظم $a.b+c$ را می‌توان متشکل از دو عبارت ساده‌تر $r_1 = a.b$ و $r_2 = c$ در نظر گرفت. در این صورت مشاهده می‌کنیم که $L(a.b+c) = \{ab, c\}$. اما براساس تعریف ۳-۲، می‌توان $r_1 = a$ و $r_2 = b+c$ را فرض نمود. به این ترتیب یک نتیجه متفاوت بصورت $L(a.b+c) = \{ab, ac\}$ بدست می‌آید. برای حل این مشکل، می‌توان تمامی عبارات را به طور کامل پرانتزبندی کرد، ولی این کار تنها باعث ایجاد نتایج بی‌فایده و غیرلازم می‌شود. بنابراین به جای آن، از یک قرارداد متداول در ریاضیات و زبان‌های برنامه‌سازی استفاده می‌کنیم. یعنی برای بررسی عبارات، مجموعه‌ای از قوانین تقدم را ارائه می‌کنیم که در آن، بستار ستاره‌ای مقدم‌تر از الحاق و الحاق مقدم‌تر از اجتماع می‌باشد. همچنین، می‌توان سمبل الحاق را حذف کرده و بطور نمونه $r_1 . r_2$ را به فرم $r_1 r_2$ نوشت.

ممکن است این قانون جدید در ابتدا کمی پیچیده به نظر برسد، ولی با کمی تمرین می‌توان زبانی را که یک عبارت منظم به آن اشاره می‌کند، به سرعت بررسی نمود.

مثال ۳-۳

برای الفبای $\Sigma = \{a,b\}$، عبارت

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

منظم بوده و به زبان

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

اشاره دارد. صحت این ادعا را با در نظر گرفتن بخش‌های مختلف r بررسی می‌کنیم. بخش اول، $(a+b)^*$، به معنای هر رشته‌ای از a ها و b ها است. بخش دوم، $(a+bb)^*$، بیانگر یک a یا دو b می‌باشد. در نتیجه، $L(r)$ مجموعه تمام رشته‌ها روی $\{a,b\}$ است که به یک a یا یک bb، ختم می‌شوند.

■

مثال ۳-۴

عبارت

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

به مجموعه تمام رشته‌هایی با تعداد زوجی از aها که پیش از تعداد فردی از bها می‌آیند، اشاره می‌کند؛ یعنی،

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

■

تبدیل مجموعه رشته‌ها به عبارات منظم کمی مشکل‌تر از عکس آن است.

مثال ۳-۵

برای $\Sigma=\{0,1\}$، عبارت منظم r را به صورتی ارائه دهید که

$$L(r)=\{w\in\Sigma^*: w \text{ حداقل دارای دو صفر متوالی است}\}.$$

به این ترتیب استدلال می‌کنیم:

هر رشته‌ای در $L(r)$ باید در جایی از خود حاوی 00 باشد، اما آنچه قبل و بعد از این دو صفر قرار می‌گیرد، کاملاً اختیاری است. برای هر رشته دلخواه روی $\{0,1\}$ می‌توان $(0+1)^*$ را تعریف کرد. با کنار هم قرار دادن این مشاهدات، نتیجه می‌گیریم که

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

■

مثال ۳-۶

یک عبارت منظم برای زبان

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

پیدا کنید.

علیرغم شباهت ظاهری این مثال با مثال ۳-۵، بدست آوردن عبارت منظم برای آن تا حدودی مشکل‌تر است. مشاهده می‌کنیم که هر زمان یک سمبل 0 رخ می‌دهد، حتماً بعد از آن یک سمبل 1 قرار می‌گیرد. قبل و بعد از این زیررشته‌ها ممکن است تعداد دلخواهی سمبل 1 ظاهر شود. بنابراین، پاسخ با تکرار رشته‌هایی به فرم $1\cdots101\cdots1$ بوجود می‌آید لذا عبارت منظم $(1^*011^*)^*$ به زبان بالا اشاره می‌کند. اما چون هنوز در مورد رشته‌هایی که به 0 ختم می‌شوند و یا متشکل از 1 هستند. صحبت نکرده‌ایم، پاسخ سؤال ناقص است. با بررسی این حالتهای خاص، به جواب

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

می‌رسیم. با کمی تغییر در استدلال، یعنی در نظر گرفتن L به صورت تکراری از رشته‌های 1 و 01، عبارت کوتاهتر

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

بدست می‌آید. این‌دو عبارت به ظاهر متفاوت، از آنجایی‌که به یک زبان یکسان اشاره می‌کنند، پاسخ‌های درست مسأله هستند. بطورکلی، برای هر زبان مفروض، تعداد نامحدودی عبارت منظم وجود دارد.

با کمی بررسی متوجه می‌شویم که این زبان مکمل زبان مثال ۳-۵ است. با این وجود، عبارت‌های منظم چندان شبیه هم نبوده و نمی‌توان از آنها به ارتباط نزدیک زبان‌ها با یکدیگر پی برد.

■

در مثال آخر، هم‌ارزی عبارات منظم معرفی شده است. دو عبارت منظم را در صورتی هم‌ارز می‌خوانیم که به زبان یکسانی اشاره کنند. هرچند می‌توان برای تسهیل عبارات منظم، از قوانین هم‌ارزی مختلفی استفاده کرد (تمرین ۲۰ بخش تمرین‌ها که در زیر آمده است)، فعلاً صرفنظر می‌کنیم.

تمرین‌ها

۱. تمامی رشته‌های موجود در $L((a+b)^*b(a+ab)^*)$ با طول کمتر از چهار را پیدا کنید.

۲. آیا عبارت $(0+1)^*00(0+1)^*(0+1)^*(0+1)$ اشاره به زبان مثال ۳-۵ دارد؟

۳. نشان دهید که $r=(1+01)^*(0+1^*)$ به زبان مثال ۳-۶ اشاره می‌کند. دو عبارت منظم هم‌ارز دیگر برای آن پیدا کنید.

۴. یک عبارت منظم برای مجموعه $\{a^nb^m: n\geq 3, m \text{ زوج است}\}$ پیدا کنید.

۵. یک عبارت منظم برای مجموعه $\{a^nb^m: (n+m) \text{ زوج است}\}$ پیدا کنید.

۶. عبارات منظمی برای زبان‌های زیر ارائه دهید:

الف) $L_1=\{a^nb^m: n\geq 4, m\leq 3\}$.

ب) $L_2=\{a^nb^m: n<4, m\leq 3\}$.

ج) مکمل L_1.

د) مکمل L_2.

عبارات $(\emptyset^*)^*$ و $a\emptyset$ به چه زبان‌هایی اشاره دارند؟

زبان $L((aa)^*b(aa)^*+a(aa)^*ba(aa)^*)$ را در چند سطر توصیف کنید.

عبارت منظمی برای L^R ارائه دهید که در آن، L همان زبان تمرین ۱ می‌باشد.

عبارت منظمی برای $L=\{a^nb^m : n\geq 1, m\geq 1, nm\geq 3\}$ ارائه دهید.

عبارت منظمی برای $L=\{a\,b\,\hat{}\,w : n\geq 3, w\in\{a,b\}^+\}$ ارائه دهید.

عبارت منظمی برای مکمل زبان مثال ۳-۴ پیدا کنید.

عبارت منظمی برای $L=\{vwv : v,w\in\{a,b\}^*, |v|=2\}$ ارائه دهید.

عبارت منظمی برای $L=\{vwv : v,w\in\{a,b\}^*, |v|\leq 3\}$ ارائه دهید.

عبارت منظمی برای

$L=\{w\in\{0,1\}^* :$ w دارای دقیقا یک زوج صفر متوالی است $\}$

ارائه دهید.

عبارت‌های منظمی برای زبان‌های زیر با $\Sigma=\{a,b,c\}$ ارائه دهید:

الف) تمام رشته‌های حاوی فقط یک a،

ب) تمام رشته‌های حاوی حداکثر سه a،

ج) تمام رشته‌هایی که از هر یک از سمبل‌های Σ در آنها حداقل یک سمبل وجود داشته باشند،

د) تمام رشته‌هایی که شامل زیر رشته‌ای از aها با طول بیشتر از ۲ نباشد،

ه)* تمام رشته‌هایی که در آن طول همه زیر رشته‌های a مضرب ۳ باشند.

عبارت‌های منظمی برای زبان‌های زیر روی $\{0,1\}$ بنویسید.

الف) تمام رشته‌هایی که به 01 ختم می‌شوند،

ب) تمام رشته‌هایی که به 01 ختم نمی‌شوند،

ج) تمام رشته‌های حاوی تعداد زوجی 0،

د) تمام رشته‌هایی که در آنها حداقل دو زیررشته 00 وجود داشته باشد (توجه داشته باشید که براساس تفسیر عادی زیررشته‌ها، 000 حاوی دو زیررشته این چنین است)،

ه) تمام رشته‌هایی که در آنها حداکثر دو زیررشته 00 وجود داشته باشد،

و)* تمام رشته‌هایی که دارای زیررشته 101 نباشند.

عبارات منظمی برای زبان‌های زیر روی $\{a,b\}$ پیدا کنید.

الف) $L=\{w : |w| \bmod 3=0\}$.

ب) $L=\{w : n_a(w) \bmod 3=0\}$.

ج) $L=\{w : n_a(w) \bmod 5>0\}$.

قسمتهای (الف)، (ب) و (ج) تمرین ۱۸ را با $\Sigma=\{a,b,c\}$ حل کنید.

۲۰. مشخص کنید آیا ادعاهای زیر برای تمام عبارات منظم r_1 و r_2 برقرار است یا خیر. علامت $\equiv$ به معنای هم‌ارزی عبارات منظم و به این معناست که هر دو عبارت به زبان یکسانی اشاره 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) $(r_1\,r_2)^*\equiv r_1^*r_2^*$

۲۱. روش کلی برای تبدیل هر عبارات منظم r به $\bar{r}$ ارائه دهید بطوریکه $(L(r))^R=L(\bar{r})$.

۲۲. اثبات کنید که عبارات مثال ۳-۶ واقعاً به زبان ارائه شده، اشاره دارد.

۲۳. برای عبارت منظم و مفروض r که حاوی λ یا $\emptyset$ نباشد، مجموعه‌ای از شروط لازم و کافی را ذکر کنید که باید برای r باید برقرار باشد تا $L(r)$ نامتناهی شود.

۲۴. از زبانهای صوری می‌توان برای شرح انواع شکل‌های دوبعدی استفاده کرد. زبان‌های کدزنجیری روی الفبای $\Sigma=\{u,d,r,l\}$ تعریف می‌شوند که در آن، این سمبل‌ها به معنای خطوط مستقیم با طول واحد به ترتیب در هریک از جهت های بالا، پایین، راست و چپ هستند. یکی از نمونه‌های این مجموعه سمبل‌ها، $urdl$ به معنای مربعی با طول اضلاع یک است. تصاویری از شکل‌های تعریفی بوسیله عبارات منظم $(rd)^*$، $(urddru)^*$ و $(rulldr)^*$ را رسم کنید.

۲۵. در تمرین ۲۴، شرایط کافی را ذکر کنید تا عبارت فوق شکلی با محیط بسته و نقاط آغاز و پایان یکسان را ترسیم کند. آیا این شرایط، لازم هم هستند؟

۲۶. یک nfa پیدا کنید.که زبان $L(aa^*(a+b))$ را بپذیرد.

۲۷. عبارت منظمی پیدا کنید که به تمام رشته‌های بیتی، دلالت کند. که مقدار آن، در صورتی که بعنوان یک عدد صحیح دودویی تفسیر شود، بزرگتر یا مساوی ۴۰ باشد.

۲۸. عبارت منظمی را برای تمام رشته‌های بیتی، با بیت پایانی ۱ پیدا کنید که به عنوان یک عدد صحیح دودویی تفسیر شده و مقادیر آنها بین ۱۰ و ۳۰ نباشند.

ارتباط بین عبارات منظم و زبانهای منظم

زبانهای منظم و عبارات منظم ارتباط نزدیکی با هم دارند. در واقع این دو مفهوم اساساً یکی هستند؛ بطوریکه به ازای هر زبان منظم، یک عبارت منظم و به ازای هر عبارت منظم، یک زبان منظم وجود دارد. این ادعا را در دو مرحله اثبات می‌کنیم.

عبارات منظم، زبانهای منظم را نمایش می‌دهند.

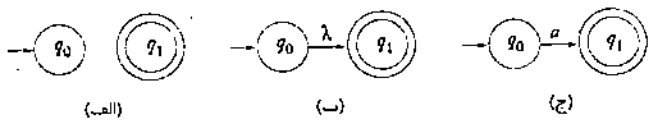
ابتدا نشان می‌دهیم که اگر r یک عبارت منظم باشد، آنگاه $L(r)$ یک زبان منظم است. براساس تعریف، یک زبان در صورتی منظم است که بوسیله یک dfa پذیرفته شود. به دلیل هم‌ارزی nfaها و

dfaها می‌توان گفت: یک زبان در صورتی منظم است که بوسیله یک nfa پذیرفته شود. حال نشان می‌دهیم که با داشتن عبارت منظم مفروض r، می‌توانیم nfaای بسازیم که $L(r)$ را بپذیرد. ساخت این nfa براساس تعریف بازگشتی $L(r)$ انجام می‌شود. ابتدا اتوماتای ساده‌ای را برای قسمتهای (۱)، (۲) و (۳) تعریف ۳-۲ ساخته و سپس نشان می‌دهیم که می‌توان با ترکیب آنها، قسمتهای پیچیده‌تر (۴)، (۵) و (۷) را هم پیاده‌سازی کرد.

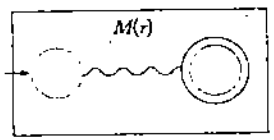
قضیه ۳-۱

فرض کنیم r یک عبارت منظم باشد. آنگاه یک پذیرنده متناهی نامعین وجود دارد که $L(r)$ را می‌پذیرد. در نتیجه، $L(r)$ یک زبان منظم است.

اثبات: کار را با ساخت اتوماتایی آغاز می‌کنیم که زبانهای مربوط به عبارات منظم و پایه‌ای $a \in \Sigma$، λ و $\varnothing$ را بپذیرند. این اتوماتا به ترتیب در شکل‌های ۳-۱ (الف)، (ب) و (ج) نشان داده شده‌اند. حال فرض کنید که اتوماتای $M(r_1)$ و $M(r_2)$ به ترتیب، زبانهای تعریفی عبارتهای منظم r_1 و r_2 را بپذیرند. نیازی به ساخت اجزاء ریزتر این اتوماتا نیست و می‌توان به شمایی از آنها، مانند شکل ۳-۲، اکتفا کرد. در این تصویر، رأس گراف سمت چپ بیانگر حالت شروع و رأس گراف سمت راست بیانگر حالت پایان می‌باشد. در تمرین ۷ بخش ۲-۳، ادعا کردیم که به ازای هر nfa، یک nfa نظیر با فقط یک حالت پایانی وجود دارد؛ بنابراین در اینجا هم فقط یک حالت پایانی وجود خواهد داشت. پس از نمایش $M(r_1)$ و $M(r_2)$، اتوماتای لازم برای عبارات منظم $r_1 + r_2$، $r_1 . r_2$ و r_1^* را می‌سازیم. این ساختارها را در شکل‌های ۳-۳ تا ۳-۵ مشاهده می‌کنید. همانطور که از بررسی شکل‌ها متوجه می‌شویم، حالت‌های شروع و پایان ماشین‌های سازنده از بین رفته و حالت‌های شروع و پایان جدیدی را به خود می‌گیرند. با کنار هم قرار دادن این مراحل می‌توان اتوماتای لازم برای عبارات منظم با هر پیچیدگی را ساخت.

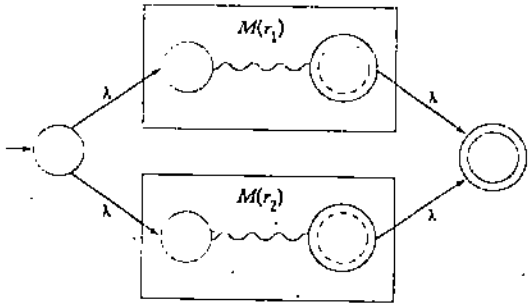


شکل ۳-۱ الف) nfa، $\varnothing$ را می‌پذیرد. ب) nfa، $\{\lambda\}$ را می‌پذیرد. ج) nfa، $\{a\}$ را می‌پذیرد.

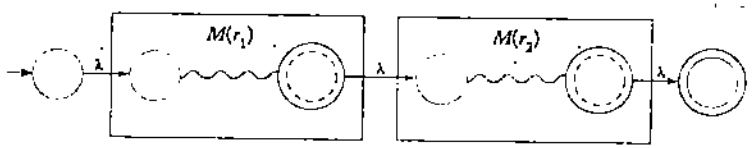


شکل ۳-۲ ارائه شمایی از nfa، که $L(r)$ را می‌پذیرد.

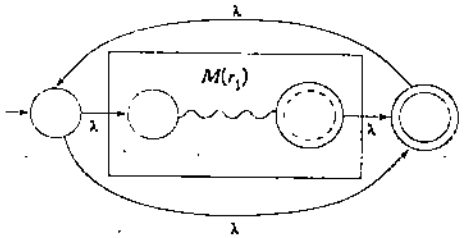
با بررسی تفسیرهای گراف‌های ۳-۳ تا ۳-۵ کارآمدی این ساختار آشکار می‌شود. برای استدلال دقیقتر، می‌توان ابتدا روش قانونمند برای ساخت حالت‌ها و انتقال‌های ماشین، ترکیبی از حالت‌ها و انتقالهای هر یک از بخش‌ها ایجاد نمود؛ سپس، بوسیله استقراء روی تعداد عملگرها اثبات کرد که این ساختار، اتوماتی را ایجاد می‌کند که زبان اشاره شده بوسیله تمامی عبارات منظم را می‌پذیرد. از آنجایی‌که همواره این نتایج بدیهی و درست هستند، از اثبات آنها صرفنظر می‌کنیم. ■



شکل ۳-۳ اتومات مربوط به $L(r_1 + r_2)$.



شکل ۳-۴ اتومات مربوط به $L(r_1 r_2)$.



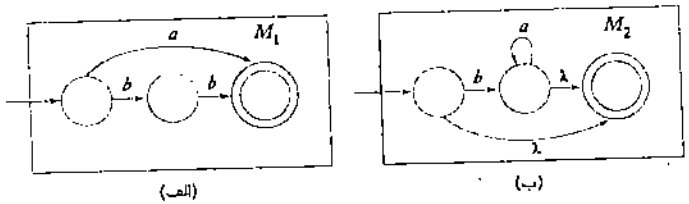
شکل ۳-۵ اتومات مربوط به $L(r_1^*)$.

مثال ۳-۵

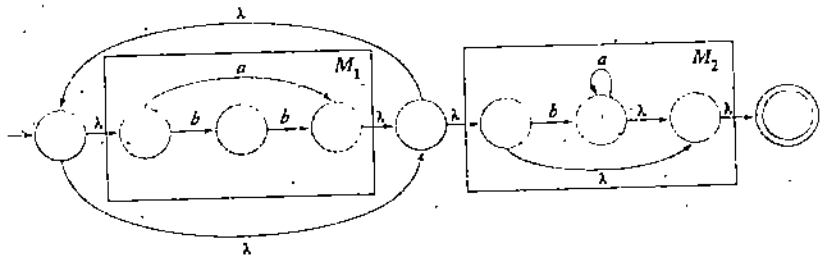
nfa ای را پیدا کنید که $L(r)$ را بپذیرد. بطوریکه:

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

اتوماتای مربوط به $(a+bb)$ و $(ba^*+\lambda)$ که مستقیماً از اصول پایه‌ای قبلی ساخته شده‌اند، در شکل ۳-۶ ارائه شده است. با استفاده از ساختار قضیه ۳-۱ و کنار هم قرار دادن موارد فوق، جواب نهایی بصورت شکل ۳-۷ بدست می‌آید.



شکل ۳-۶ الف) M_1، $L(a+bb)$ را می‌پذیرد. ب) M_2، $L(ba^*+\lambda)$ را می‌پذیرد.



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

■

عبارات منظم برای زبان‌های منظم

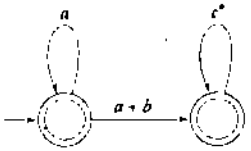
از نظر شهودی، عکس قضیه ۳-۱ منطقاً صحیح بوده و بنابراین، برای هر زبان منظم، یک عبارت منظم متناظر وجود دارد. بدلیل آنکه هر زبان منظمی دارای یک nfa متناظر و در نتیجه، یک گراف انتقال است، فقط لازم است عبارت منظمی پیدا کنیم که بتواند برچسب‌های تمامی قدمها از q_0 به یکی از حالات پایانی را ایجاد کند. این کار چندان مشکل به نظر نمی‌رسد، اما وجود حلقه‌هایی که می‌توان به دلخواه و با هر ترتیبی از آن عبور کرد، باعث پیچیدگی امر می‌شود. روشهای مختلفی برای حل این مشکل وجود دارد که یکی از شهودی‌ترین آنها استفاده از گراف‌های انتقال تعمیم یافته (GTG) است. از آنجایی‌که در این کتاب بندرت از روش مذکور استفاده شده و بعلاوه، هیچ نقشی در بحث بعدی ما ایفا نمی‌کند، بطور غیررسمی به آن می‌پردازیم.

گراف انتقالی تعمیم یافته، گراف انتقالی است که یال‌های آن با عبارات منظم برچسب‌دار شده باشند؛ فارغ از این ویژگی، دقیقاً همان گراف انتقال معمولی خواهد بود. برچسب هر قدم از یک حالت شروع به یکی از حالت‌های پایانی در واقع حاصل الحاق چندین عبارت منظم بوده و از این‌رو، خود

یک عبارت منظم می‌باشد. رشته‌های اشاره شده بوسیله این‌گونه عبارات منظم، زیرمجموعه‌هایی از زبان پذیرفته شده بوسیله گراف انتقال تعمیم یافته هستند. بعلاوه، زبان کلی، حاصل اجتماع زیرمجموعه‌هایی است که به این ترتیب بوجود می‌آیند.

مثال ۳-۸

شکل ۳-۸ یک گراف انتقال تعمیم یافته را نمایش می‌دهد. همانطور که از بررسی گراف‌ها به راحتی می‌توان دریافت، زبان پذیرفته شده بوسیله گراف مذکور $L(a^*+a^*(a+b)c^*)$ است. یال (q_0,q_0) با برچسب a، حلقه‌ای است که هر تعداد a را بوجود آورده و بنابراین، بیانگر $L(a^*)$ می‌باشد. می‌توان این یال را بدون هیچ گونه تغییری در زبان پذیرفته شده در گراف مربوطه، a^* برچسب‌گذاری کرد.



شکل ۳-۸

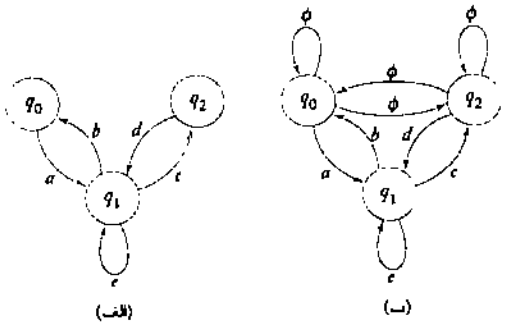
■

در صورت تفسیر درست برچسب یال‌های گراف پذیرنده‌های متناهی نامعین، می‌توان آنها را به عنوان گراف‌های انتقال تعمیم یافته تلقی کرد. یالی که فقط با یک سمبل a برچسب‌دار می‌شود را می‌توان یالی با برچسب عبارت a تفسیر کرد؛ در حالی که یالی با برچسب سمبل‌های متعدد $a,b,\ldots$ بعنوان یالی با برچسب عبارت $a+b+\cdots$ تفسیرمی‌شود. از این مشاهده، نتیجه می گیریم که به ازای هر زبان منظم، یک گراف انتقال تعمیم یافته وجود دارد که آنرا می‌پذیرد. بالعکس، هر زبانی که بوسیله یک گراف انتقال تعمیم یافته پذیرفته شود، منظم است به عنوان نتیجه قضیه ۳-۱. بدلیل آنکه برچسب هر قدم در گراف انتقال تعمیم یافته یک عبارت منظم است. از این استدلال می‌توان به نتایج متعددی دست یافت که لزومی به ذکر آنها در اینجا وجود ندارد؛ ولی خواننده‌های علاقه‌مند را برای مطالعه دقیق‌تر به تمرین ۲۲ بخش ۴-۳ ارجاع می‌دهیم.

هم‌ارزی گرافهای انتقال تعمیم یافته در قالب زبان پذیرفته شده، تعریف می‌شود. در مبحث بعد، دنباله‌ای از GTGهای ساده را معرفی می‌کنیم. به همین دلیل، بهتر و منطقی‌تر است که با GTGهای کامل کار کنیم. GTG کامل گرافی است که تمام یال‌ها در آن حضور داشته باشند. اگر در یک GTG، پس از تبدیل از nfa، برخی یال‌ها حضور نداشته باشند، آنها را با $\emptyset$ برچسب‌دار می‌کنیم. هر GTG کامل با $|V|$ رأس، دقیقاً $|V|^2$ یال دارد.

مثال ۳-۹

GTG شکل ۳-۹ (الف) کامل نیست. شکل ۳-۹ (ب) تکمیل شده آنرا نشان می‌دهد.



شکل ۳-۹

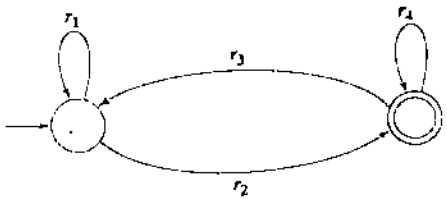
■

حال فرض کنیم که یک GTG کامل با دوحالت با ساختاری ساده، مانند شکل ۳-۱۰، در اختیار داشته باشیم. با بررسی GTG به راحتی می‌توان دریافت که عبارت منظم

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

تمام مسیرهای ممکن از حالت شروع به حالت پایانی را پوشش داده و بنابراین، عبارت منظم متناظر با گراف انتقال حالت است.

در صورتی‌که یک GTG بیش از دو حالت داشته باشد، می‌توان مرحله به مرحله با حذف هریک از حالت‌ها، گراف نظیر آن را پیدا کرد. پیش از ارائه روش کلی کار، آن را به کمک مثال زیر شرح می‌دهیم.



شکل ۳-۱۰

مثال ۳-۱۰

GTG کامل شکل ۳-۱۱ را در نظر بگیرید. پیش از حذف q_2، ابتدا چند یال جدید را معرفی می‌کنیم. به این منظور،

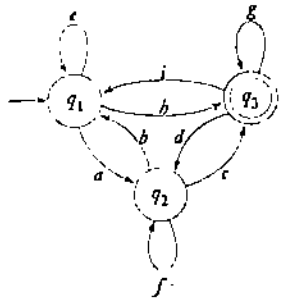
یک یال از q_1 به q_1 ایجاد کرده و آنرا با $e + af^*b$ برچسب‌دار می‌کنیم،

یک یال از q_1 به q_3 ایجاد کرده و آنرا با $h + af^*c$ برچسب‌دار می‌کنیم،

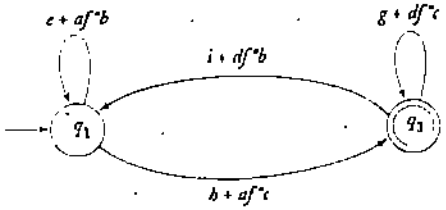
یک یال از q_3 به q_1 ایجاد کرده و آنرا با $i + df^*b$ برچسب‌دار می‌کنیم،

یک یال از q_3 به q_3 ایجاد کرده و آنرا با $g + df^*c$ برچسب‌دار می‌کنیم.

سپس، q_2 و تمام یال‌های مربوط به آنرا حذف می‌کنیم. به این ترتیب، GTG شکل ۳-۱۲ بدست می‌آید. می‌توان با تقلید از نحوه ایجاد عبارات منظمی از قبیل af^*c و e^*ab، هم‌ارزی این دو GTT را بررسی کرد.



شکل ۳-۱۱



شکل ۳-۱۲

■

در GTGهای اختیاری، مرحله به مرحله هر یک از حالت‌ها را حذف کرده و این‌کار را آنقدر ادامه می‌دهیم تا فقط دو حالت باقی بماند. سپس، با استفاده از رابطه (۳-۱) عبارت منظم پایانی را بدست می‌آوریم. این فرآیند طولانی را می‌توان به کمک روال زیر به صورتی ساده و قابل فهم ارائه داد.

روال: nfa به rex

۱. کار را با یک nfa با حالت‌های q_0، q_1،... و q_n و فقط یک حالت پایانی، جدا از حالت شروع آن، آغاز می‌کنیم.

۲. nfa را به گراف انتقال تعمیم یافته کامل تبدیل کرده و r_{ij} را بعنوان برچسب یالی از q_i به q_j در نظر می‌گیریم.

۳. در صورتی‌که GTG فقط دو حالت داشته باشد، q_i حالت شروع و q_j حالت پایانی گراف بوده و عبارت منظم

$$r = r_{ii}^{*} r_{ij} \left(r_{jj} + r_{ji} r_{ii}^{*} r_{ij} \right)^{*} \qquad (۳-۲)$$

نظیر آن می‌باشد.

۴. اگر GTG سه حالت داشته باشد، یعنی q_i حالت شروع، q_j حالت پایانی و q_k هم حالت ثالثی باشد، یالهای جدیدی را با برچسب

$$r_{pq} + r_{pk}\, r_{kk}^{*}\, r_{kq} \qquad (۳-۳)$$

برای $q = i, j$ و $p = i, j$ معرفی می‌کنیم. سپس، حالت q_k و یال‌های مربوط به آنرا حذف می‌کنیم.

۵. اگر GTG چهار یا بیشتر حالت داشته باشد، حالت q_k را برای حذف انتخاب می‌کنیم. قانون شماره ۴ را برای همه زوج حالت‌های (q_i, q_j) با فرض $i \neq k$ و $j \neq k$، اعمال می‌کنیم. در هر مرحله و هر جا که امکان داشته باشد، می‌توان از قانون‌های ساده کننده

$$r + \varnothing = r,$$
$$r\varnothing = \varnothing,$$
$$\varnothing^{*} = \lambda,$$

استفاده کرد. پس از انجام این مراحل، حالت q_k را حذف می‌کنیم.

۶. مراحل ۳ تا ۵ را تا حصول عبارت منظم نهایی، تکرار می‌کنیم.

مثال ۳-۱۱

یک عبارت منظم برای زبان

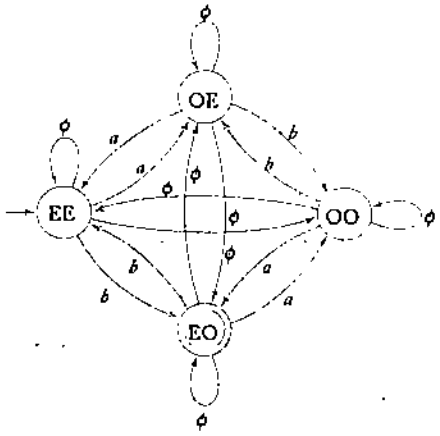
$$L = \{ w \in \{a,b\}^{*} : n_a(w) \text{ فرد است و } n_b(w) \text{ زوج است} \}$$

پیدا کنید. از همان ابتدای ساخت عبارت منظم با استفاده از این شرح با مشکلات مختلفی مواجه می‌شویم. از طرف دیگر، پیدا کردن یک nfa برای این عبارت فقط وقتی براحتی انجام می‌شود که از نام رئوس درست استفاده شود. برای نمایش زوج بودن تعداد a ها و b ها، رأس‌ها را با EE نامگذاری کرده و برای نمایش فرد بودن a ها و زوج بودن b ها، از نام OE استفاده می‌کنیم و الی آخر. به این ترتیب، راه‌حلی بدست می‌آید که بعد از تبدیل به گراف انتقال تعمیم یافته کامل، مشابه شکل ۳-۱۳ خواهدبود.

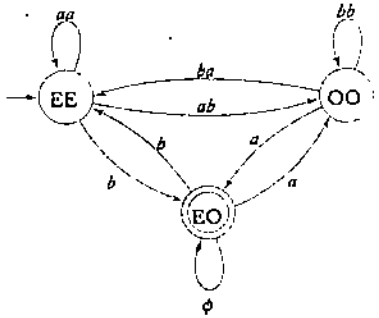
اینک، با استفاده از روال nfa به rex، این تبدیل را در یک عبارت منظم اعمال می‌کنیم. برای حذف حالت OE، از تساوی (۳-۳) استفاده می‌کنیم. یال بین EE و خودش دارای برچسب

$$r_{EE} = \varnothing + a\varnothing^{*}a$$
$$= aa.$$

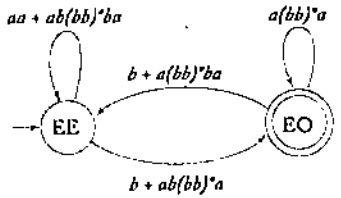
خواهد بود. به همین ترتیب ادامه می‌دهیم تا به GTG شکل ۳-۱۴ برسیم. سپس با حذف حالت OO، شکل ۳-۱۵ بوجود می‌آید. در نهایت، با استفاده از تساوی (۳-۲)، عبارت منظم درست را بدست می‌آوریم.



شکل ۳-۱۳



شکل ۳-۱۴



شکل ۳-۱۵

فرآیند تبدیل یک nfa به عبارت منظم هم‌ارز آن، هرچند مشخص، اما کسل‌کننده و یکنواخت بوده و عبارات منظم حاصل از آن، پیچیده و فاقد کاربرد عملی می‌باشد. اما چون این فرآیند ایده‌ای برای اثبات یک نتیجه مهم را فراهم می‌کند، تصمیم به ارائه آن در اینجا گرفتیم.

قضیه ۳-۲

L را یک زبان منظم فرض می‌کنیم. بنابراین، عبارت منظمی به ازای r وجود خواهد داشت بطوریکه $L = L(r)$.

اثبات: اگر L منظم باشد، حتماً یک nfa برای آن وجود خواهد داشت. با در نظر گرفتن مسأله تعمیم، می‌توان فرض کرد که این nfa فقط یک حالت پایانی، متمایز از حالت شروع آن دارد. این nfa را به یک گراف انتقال تعمیم یافته کامل تبدیل کرده و روال *nfa* به *rex* را برای آن انجام می‌دهیم. به این ترتیب، عبارت منظم مورد نظر r بدست خواهد آمد.

هر چند نتیجه فوق قابل قبول است، برای اثبات دقیق قضیه باید نشان دهیم که در هر مرحله روال، یک GTG هم‌ارز بوجود می‌آید. انجام این کار تکنیکی را بر عهده خواننده واگذار می‌کنیم. ■

استفاده از عبارات منظم برای توصیف الگوهای ساده

در مثال ۱-۱۵ و تمرین ۱۶ بخش ۲-۱، به ارتباط بین پذیرنده‌های متناهی و برخی اجزاء ساده‌تر زبانهای برنامه‌سازی، از قبیل شناسه‌ها یا اعداد صحیح و اعداد حقیقی، پی بردیم. ارتباط بین اتوماتای متناهی و عبارات منظم به این معناست که می‌توان از عبارات منظم نیز، بعنوان روشی برای شرح این ویژگی‌ها استفاده نمود. مثلاً در بسیاری از زبانهای برنامه‌سازی، مجموعه اعداد صحیح ثابت بوسیله عبارت منظم

$$sdd^*,$$

تعریف می‌شود که در آن، s به معنای یکی از سمبل‌های $\{+,-,\lambda\}$ و d به معنای ارقام 0 تا 9 است. اعداد صحیح ثابت نمونه ساده‌ای از مفهومی به نام "الگو" هستند – الگو در واقع مجموعه‌ای از اشیاء با برخی ویژگی‌های مشترک است- تطابق الگو به معنای نسبت دادن یک شیء مشخص به یکی از چند دسته‌بندی می‌باشد. در اغلب موارد، موفقیت تطابق الگو در گرو پیدا کردن روشی مؤثر و مفید برای شرح الگوهاست. هرچند این بحث یکی از حوزه‌های گسترده و پیچیده علوم کامپیوتر است، ما فقط به صورت موردی به آن اشاره می‌کنیم. مثال زیر نمونه ساده‌شده و در عین حال، آموزنده‌ای از کاربرد مفید و مؤثر ایده‌های مورد بحث تا به اینجا در زمینه تطابق الگو است.

مثال ۳-۱۲

یکی از موارد کاربرد تطابق الگو، ویرایش متن است. تمامی ویرایشگرهای متنی، فایلها را از نظر وجود یا عدم وجود رشته‌ای خاص بررسی می‌کنند. در اغلب ویرایشگرها با بسط این ویژگی، امکان

جستجوی الگوها نیز فراهم شده است. بعنوان مثال، ویرایشگر *vi* در سیستم عامل UNIX فرمان /aba*c/ را بعنوان دستورالعملی برای جستجو در فایل جهت یافتن اولین رشته *ab* که پس از آن تعدادی اختیاری *a* و سپس یک *c* قرار داشته باشد، شناسایی می‌کند. از این مثال می‌توان به لزوم هماهنگی بین ویرایشگرهای تطابق‌دهنده الگو با عبارات منظم پی برد.

یکی از چالش‌ها در این قبیل برنامه‌های کاربردی، نوشتن برنامه‌ای کارا برای تشخیص الگوی رشته‌هاست. جستجوی فایل از نظر وجود یا عدم وجود رشته‌ای مفروض یک تمرین برنامه‌سازی بسیار ساده است ولی در اینجا وضعیت پیچیده‌تری داریم. در اینجا ما با تعداد نامحدودی الگوی پیچیده سروکار داریم؛ بعلاوه، الگوها از قبل مشخص نبوده و در زمان اجرا ایجاد می‌شوند. شرح الگو بخشی از ورودی بوده و به همین دلیل، فرآیند تشخیص باید انعطاف‌پذیر باشد. برای حل این مشکل، در اغلب موارد از برخی ایده‌های نظریه اتوماتا استفاده می‌شود.

اگر برای تشخیص الگو از عبارات منظم استفاده شود، برنامه شناخت الگو با اقتباس این شرح و استفاده از ساختار قضیه ۳-۱، آنرا به nfa نظیر تبدیل می‌کند. سپس، می‌توان از قضیه ۲-۲ جهت تبدیل nfa به dfa استفاده کرد. dfa حاصله که به صورت یک جدول انتقال است، در واقع الگوریتم تطابق الگو می‌باشد. فقط لازم است که برنامه‌نویس با ایجاد یک راه‌انداز، چارچوبی کلی برای استفاده از جدول را فراهم آورد. به این ترتیب، می‌توان به صورت خودکار تعداد زیادی الگو را که در زمان اجرا تعریف شده‌اند، اداره کرد.

البته کارآیی برنامه هم باید مدنظر قرارگیرد. ساخت اتوماتای متناهی از عبارات منظم با استفاده از قضایای ۲-۱ و ۳-۱ باعث ایجاد اتوماتا با حالات زیادی می‌شود. در صورت محدودیت فضای حافظه، می‌توان از روش کارای تعریف شده در بخش ۲-۴ برای کاهش حالت استفاده کرد.

■

تمرین‌ها

۱. با استفاده از ساختار قضیه ۳-۱، nfaای پیدا کنید که زبان $L(ab^*aa+bba^*ab)$ را پذیرش کند.

۲. یک nfa پیدا کنید که مکمل زبان تمرین ۱ را پذیرش کند.

۳. یک nfa ارائه دهید که زبان $L((a+b)^*b(a+bb)^*)$ را پذیرش کند. ⊖

۴. dfaهایی پیدا کنید که زبان‌های زیر را پذیرش کنند:

الف). $L(aa^*+aba^*b^*)$. ⊖

ب). $L(ab(a+ab)^*(a+aa))$.

ج). $L((abab)^*+(aaa^*+b)^*)$.

د). $L(((aa^*)^*b)^*)$.

۵. dfaهایی پیدا کنید که زبان‌های زیر را پذیرش کند:

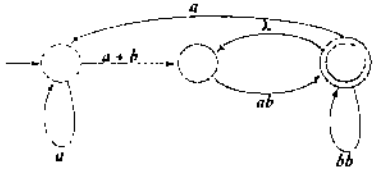
الف). $L=L(ab^*a^*)\cup L((ab)^*ba)$.

ب). $L=L(ab^*a^*)\cap L((ab)^*ba)$.

۶. یک nfa برای تمرین ۱۷ (ر) بخش ۱-۳ پیدا کنید. سپس با استفاده از این nfa، عبارت منظمی را برای آن زبان ارائه دهید.

۷. dfa کمینه‌ای را پیدا کنید که زبان $L(a^*bb) \cup L(ab^*ba)$ را پذیرش کند.

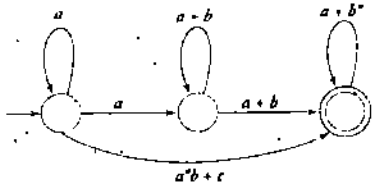
۸. گراف انتقال تعمیم یافته زیر را در نظر بگیرید:



الف) گراف انتقال تعمیم یافته نظیر آن را فقط با دو حالت پیدا کنید.

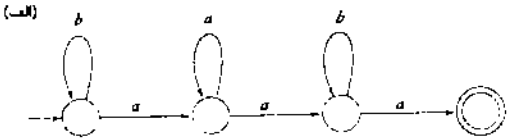
ب) این گراف چه زبانی را می‌پذیرد؟

۹. گراف انتقال تعمیم یافته زیر چه زبانی را می‌پذیرد؟

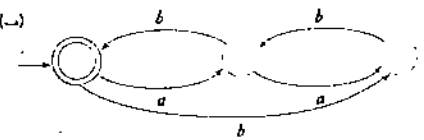


۱۰. عبارات منظمی را برای زبان‌های پذیرفته شده بوسیله اتوماتای زیر پیدا کنی.

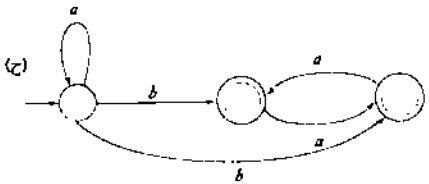
(الف)



(ب)



(ج)



۱۱. مثال ۳-۱۱ را دوباره ولی با شروع عملیات حذف از حالت OO حل کنید.

۱۲. نشان دهید که همه برچسب‌های شکل ۳-۱۴ بدست آمده‌اند.

۱۳. عبارت منظمی را برای زبان‌های زیر روی $\{a,b\}$ بدست آورید:

الف) $L = \{w : n_a(w) \text{ و } n_b(w) \text{ هردو زوج هستند}\}$

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) $L = \{w : 2n_a(w) + 3n_b(w) \text{ زوج است}\}$

۱۴. اثبات کنید که ساختار پیشنهادی در شکل‌های ۳-۱۱ و ۳-۱۲ به عنوان گراف‌های انتقال تعمیم‌یافته هم‌ارز می‌باشند.

۱۵. یک عبارت منظم برای مجموعه تمام اعداد حقیقی زبان برنامه نویسی C بنویسید.

۱۶. در برخی از کاربردها، از قبیل برنامه‌های چک‌کننده دیکته کلمات، ممکن است نیازی به تطابق دقیق الگو وجود نداشته و فقط داشتن یک الگوی تقریبی کفایت کند. پس از قطعی شدن مجموعه سمبل‌های تطابق تقریبی، می‌توان از نظریه اتوماتا برای ساخت تطابق‌دهنده‌های تقریبی الگو استفاده کرد. بعنوان مثال، برخی از الگوها با درج یک سمبل از الگوهای اصلی مشتق می‌شوند. L را یک زبان منظم روی Σ فرض کرده و

$$insert(L) = \{uav : a \in \Sigma, uv \in L\}$$

را تعریف می کنیم. در واقع، $insert(L)$ حاوی تمام کلمات ایجاد شده از L با درج یک سمبل ساختگی در هر جای کلمه است.

*الف) dfaای را برای L ارائه داده و نشان دهید که چگونه می‌توان یک nfa برای $insert(L)$ بدست آورد.

**ب) بحث کنید که چگونه می‌توان با استفاده از عبارت منظم برای L بعنوان ورودی، یک برنامه تشخیص الگو برای $insert(L)$ نوشت؟

۱۷. *_ مشابه تمرین قبل، تمام کلماتی را در نظر بگیرید که با حذف فقط یک سمبل از رشته، از L تولید می‌شوند. بطور رسمی، این عملیات را برای زبانی به نام $drop$ تعریف کنید. با ساخت یک nfa برای L، dfa هم برای $drop(L)$ ایجاد کنید.

۱۸. با استفاده از ساختار قضیه ۳-۱، nfaهایی را برای $L(a\emptyset)$ و $L(\emptyset^*)$ پیدا کنید. آیا نتیجه کار با تعریف این زبان‌ها همخوانی دارد؟

۳-۳ گرامرهای منظم

روش سوم برای شرح زبان‌های منظم، استفاده از برخی گرامرهای ساده است. کاربرد گرامرها غالباً روشی برای تشخیص زبان‌ها محسوب می‌شود. هر وقت که با استفاده از اتومات یا به روشی دیگر

اقدام به تعریف خانواده یک زبان می‌کنیم، بهتر است از گرامر مرتبط با آن خانواده هم اطلاع داشته باشیم. ابتدا نگاهی به گرامرهای تولید کننده زبان‌های منظم خواهیم داشت.

گرامرهای خطی از راست و خطی از چپ

تعریف گرامر مفروض $G=(V,T,S,P)$ در صورتی **خطی از راست** نامیده می‌شود که تمامی قوانین آن به فرم

$$A \to xB,$$
$$A \to x,$$

باشند که در آن، $A,B \in V$ و $x \in T^*$. در حالی که، یک گرامر در صورتی **خطی از چپ** نامیده می‌شود که تمامی قوانین آن به فرم

$$A \to Bx,$$

یا

$$A \to x.$$

باشند. همچنین گرامر منظم گرامری است که خطی از راست یا خطی از چپ باشد.

توجه داشته باشید که در تمامی گرامرهای منظم، حداکثر یک متغیر در سمت راست هر یک از قوانین قرار می‌گیرد. بعلاوه، این متغیر باید همواره آخرین سمبل از سمت راست یا از سمت چپ طرف راست هر یک از قوانین باشد.

مثال ۳-۱۳

گرامر $G_1=(\{S\},\{a,b\},S,P_1)$، که در آن، P_1 بصورت

$$S \to abS \mid a$$

است، خطی از 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 از چپ است. G_1 و G_2 هر دو گرامرهای منظم هستند. دنباله

$$S \Rightarrow abS \Rightarrow ababS \Rightarrow ababa$$

اشتقاقی از G_1 است. از همین مثال ساده می‌توان پذیرفت که $L(G_1)$ زبانی است که عبارت منظم $r=(ab)^*a$ مربوط به آن را نمایش می‌دهد. به طریقی مشابه، مشاهده می‌کنیم که $L(G_2)$ زبان منظم $L(aab(ab)^*)$ است.

■

مثال ۳-۱۴

گرامر $G=(\{S,A,B\},\{a,b\},S,P)$ با قوانین

$$S \to A,$$
$$A \to aB \mid \lambda,$$
$$B \to Ab,$$

منظم نیست. هر چند هر یک از قوانین، خطی از راست و یا خطی از چپ می‌باشند، ولی خود گرامر نه خطی از راست است و نه خطی از چپ. بنابراین، گرامر منظم نیست. این گرامر نمونه‌ای از یک گرامر خطی است.

گرامر خطی گرامری است که در آن، حداکثر یک متغیر می‌تواند در سمت راست هر قانون وجود داشته و بعلاوه، هیچ محدودیتی در مورد محل این متغیر وجود ندارد. بنابراین، گرامرهای منظم، همواره خطی هستند، اما لزوماً همه گرامرهای خطی منظم نیستند.

■

در قدم بعدی، نشان می‌دهیم که گرامرهای منظم با زبان‌های منظم مرتبط بوده و همچنین، به ازای هر زبان منظم، یک گرامر منظم وجود دارد. بنابراین، گرامرهای منظم روش دیگری برای بیان زبان‌های منظم محسوب می‌شوند.

گرامرهای خطی از راست زبان‌های منظم را تولید می‌کنند

اولاً، نشان می‌دهیم که زبان تولید شده بوسیله یک گرامر خطی از راست، همواره منظم است. برای این منظور، nfa‌ای می‌سازیم که اشتقاق‌های یک گرامر خطی از راست را بکاربرد. توجه داشته باشید که فرم‌های جمله‌ای یک گرامر خطی از راست، فرم خاصی دارند؛ یعنی در آنها فقط یک متغیر وجود داشته و این متغیر، آخرین سمبل از سمت راست فرم‌های جمله‌ای می‌باشد. حال فرض کنید که در یک اشتقاق، مرحله‌ای به صورت

$$ab\cdots cD \Rightarrow ab\cdots cdE,$$

داریم که با استفاده از قانون $D \to dE$ بوجود آمده است. nfa هم‌ارز پس از دیدن سمبل d، با انتقال از حالت D به حالت E، این مرحله را بکار می‌برد. در این طرح کلی، حالت اتومات، متناظر با متغیر موجود در فرم جمله‌ای است؛ در حالی‌که بخش پردازش نشده رشته، به صورت فرم جمله‌ای شامل الفبا می‌باشد. از این ایده ساده بعنوان اساس قضیه زیر استفاده می‌کنیم.

قضیه ۳-۳

فرض کنیم $G=(V,T,S,P)$ یک گرامر خطی از راست باشد. آنگاه $L(G)$ یک زبان منظم خواهد بود.

اثبات: فرض می‌کنیم که $V=\{V_0,V_1,\ldots V_n\}$ و $S=V_0$، و بعلاوه، قوانینی به فرم $V_0\to v_1V_i$، $V_i\to v_2V_j$ و ... ، و در انتها $V_n\to v_1$ داشته باشیم. اگر w رشته‌ای در $L(G)$ باشد، آنگاه به دلیل فرم قوانین در G، اشتقاق باید به صورت زیر باشد:

$$\begin{aligned}V_0 &\Rightarrow v_1V_i\\ &\Rightarrow v_1v_2V_j\\ &\vdots\\ &\Rightarrow v_1v_2\ldots v_kV_n\\ &\Rightarrow v_1v_2\ldots v_kv_l=w.\end{aligned}\qquad(۴-۳)$$

اتومات مورد نظر با بکارگیری مرحله به مرحله از هر یک از Vها اشتقاق را دوباره تولید می‌کند. حالت شروع اتومات با V_0 نامگذاری شده و به ازای هر متغیر V_i، یک حالت غیرپایانی با برچسب V_i وجود خواهد داشت. به ازای هر قانون

$$V_i\to a_1a_2\ldots a_mV_j,$$

اتومات با انجام انتقالاتی، V_i را به V_j متصل می‌کند؛ یعنی δ به صورتی تعریف خواهد شد که

$$\delta^*(V_i,a_1a_2\ldots a_m)=V_j.$$

و به ازای هر قانون

$$V_i\to a_1a_2\ldots a_m,$$

انتقال متناظر برای اتومات بصورت زیر خواهد بود:

$$\delta^*(V_i,a_1a_2\ldots a_m)=V_f,$$

که در آن، V_f حالت پایانی است. حالت‌های میانی در این مسأله چندان اهمیتی نداشته و می‌توان از هر نام دلخواهی برای آنها استفاده کرد. شمای کلی در شکل ۳-۱۶ ارائه شده است. با کنار هم قراردادن این اجزاء، اتومات کامل ساخته می‌شود.

اینک $w\in L(G)$ را به گونه‌ای فرض کنید که در (۳-۴) صدق کند. براساس ساختار، در nfa یک مسیر از V_0 به V_i با برچسب v_1، یک مسیر از V_i به V_j با برچسب v_2 و الی آخر وجود دارد. بنابراین،

$$V_f\in\delta^*(V_0,w),$$

و w بوسیله M پذیرفته می‌شود.

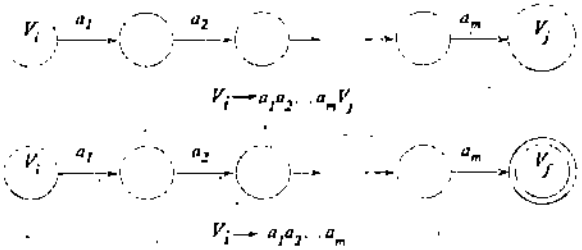
بالعکس، فرض کنید که w توسط M پذیرفته شود. براساس نحوه ساخت M، اتومات باید برای پذیرش w از مسیرهای دارای نام v_1، v_2 و ...، استفاده کرده و به کمک عبور از دنباله ای از حالات V_0، V_i و ...، به V_f برسد. بنابراین، w باید به فرم

$$w=v_1v_2\ldots v_kv_l$$

بوده و اشتقاق

$$V_0\Rightarrow v_1V_i\Rightarrow v_1v_2V_j\Rightarrow v_1v_2\ldots v_kV_n\Rightarrow v_1v_2\ldots v_kv_l$$

امکان‌پذیر است. بنابراین، w عضو $L(G)$ بوده و قضیه ثابت می‌شود.



شکل ۳-۱۶

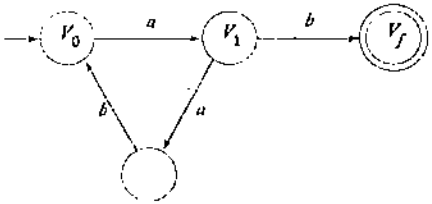
■

مثال ۳-۱۵

اتومات متناهی بسازید که زبان تولید شده بوسیله گرامر

$$V_0\to aV_1,$$
$$V_1\to abV_0\mid b.$$

را بپذیرد که در آن، V_0 متغیر شروع است. گراف انتقالی را با رئوس V_0، V_1 و V_f آغاز می‌کنیم. براساس قانون اول تولید، یالی با برچسب a بین V_0 و V_1 ایجاد می‌نماییم. براساس قانون دوم، باید رأس دیگر را به گونه‌ای ایجاد می‌کنیم که مسیری با برچسب ab بین V_1 و V_0 وجود داشته باشد. در پایان، لازم است یالی با برچسب b بین V_1 و V_f ایجاد کنیم تا اتومات شکل ۳-۱۷ بدست آید. زبان تولید و پذیرفته شده بوسیله گرامر، زبان منظم $L((aab)^*ab)$ خواهد بود.



شکل ۳-۱۷

■

گرامرهای خطی از راست برای زبان‌های منظم

برای اثبات اینکه هر زبان منظمی را می‌توان بوسیله یک گرامر خطی از راست معرفی کرد، با استفاده از dfa زبان مورد نظر، ساختار قضیه ۳-۳ را معکوس می‌کنیم. در اینصورت، حالت‌های dfa، تبدیل به متغیرهای گرامر و همچنین سمبل روی انتقالات، تبدیل به پایانی‌ها در قوانین می‌شوند.

قضیه ۳-۴

اگر L یک زبان منظم روی الفبای Σ باشد، آنگاه گرامر خطی از راست $G=(V,\Sigma,S,P)$ به صورتی وجود خواهد داشت که $L=L(G)$.

اثبات: فرض کنیم $M=(Q,\Sigma,\delta,q_0,F)$، یک dfa باشد که L را می‌پذیرد و همچنین، $Q=\{q_0,q_1,\ldots,q_n\}$ و $\Sigma=\{a_1,a_2,\ldots,a_m\}$. گرامر خطی از راست $G=(V,\Sigma,S,P)$ را به فرمی می‌نویسیم که در آن،

$$V=\{q_0,q_1,\ldots,q_n\}$$

و $S=q_0$. به ازای هر انتقال

$$\delta(q_i,a_j)=q_k$$

از M، قانون

$$q_i\rightarrow a_jq_k \qquad (۳-۵)$$

را در P قرار می‌دهیم. بعلاوه، اگر q_k در F باشد، قانون

$$q_k\rightarrow\lambda \qquad (۳-۶)$$

را به P اضافه می‌کنیم. ابتدا نشان می‌دهیم، G که به این صورت تعریف می‌شود، قادر به تولید تمام رشته‌های L خواهد بود. $w\in L$ را به فرم

$$w=a_ia_j\ldots a_ka_t$$

در نظر می‌گیریم. برای اینکه M بتواند این رشته را پذیرش کند، باید در مسیر

$$\begin{aligned}&\delta(q_0,a_i)=q_p,\\&\delta(q_p,a_j)=q_r,\\&\vdots\\&\delta(q_s,a_k)=q_t,\\&\delta(q_t,a_t)=q_f\in F\end{aligned}$$

حرکت کند. براساس ساختار بالا، این گرامر یک قانون برای هر یک از این δها خواهد داشت. بنابراین، می‌توانیم

$$q_0\Rightarrow a_iq_p\Rightarrow a_ia_jq_r\Rightarrow a_ia_j\ldots a_kq_t$$

$$\Rightarrow a_ia_j\cdots a_ka_tq_f\Rightarrow a_ia_j\cdots a_ka_t, \qquad (۳-۷)$$

را از گرامر G مشتق کرده و لذا نتیجه $w\in L(G)$ بدست می‌آید.

بالعکس، اگر $w\in L(G)$ باشد، آنگاه اشتقاق آن باید به فرم (۳-۷) باشد. بنابراین به طور ضمنی

$$\delta^*(q_0,a_ia_j\ldots a_ka_t)=q_f,$$

و اثبات به پایان می‌رسد. □

در ساخت گرامر، باید به خاطر داشته باشیم که محدودیت پذیرنده متناهی در معین بودن M، نقش اساسی در اثبات قضیه ۳-۴ ندارد. بنابراین، با کمی تغییر، می‌توان از همین ساختار برای مواردی که M، nfa است، استفاده کرد.

مثال ۳-۱۶

یک گرامر خطی از راست برای $L(aab^*a)$ بسازید. تابع انتقال برای nfa و قوانین گرامر متناظر آنرا در شکل ۳-۱۸ مشاهده می‌کنید. نتیجه با استفاده از ساختار قضیه ۳-۴ بدست آمده است. رشته $aaba$ را می‌توان از گرامر ساخته شده بوسیله

$$q_0\Rightarrow aq_1\Rightarrow aaq_2\Rightarrow aabq_2\Rightarrow aabaq_f\Rightarrow aaba.$$

مشتق نمود.

$\delta(q_0,a)=\{q_1\}$	$q_0\rightarrow aq_1$
$\delta(q_1,a)=\{q_2\}$	$q_1\rightarrow aq_2$
$\delta(q_2,b)=\{q_2\}$	$q_2\rightarrow bq_2$
$\delta(q_2,a)=\{q_f\}$	$q_2\rightarrow aq_f$
$q_f\in F$	$q_f\rightarrow\lambda$

شکل ۳-۱۸

□

هم‌ارزی زبان‌های منظم و گرامرهای منظم

در قضیه قبل، ارتباط بین زبان‌های منظم و گرامرهای خطی از راست را نشان می‌دهد. همچنین می‌توان بین زبان‌های منظم و گرامرهای خطی از چپ نیز، ارتباط مشابهی را برقرار کرده و به این ترتیب، هم‌ارزی کامل گرامرهای منظم و زبان‌های منظم را اثبات نمود.

قضیه ۳-۵

زبان L منظم است اگر و تنها اگر گرامر خطی از چپ G وجود داشته باشد بطوریکه $L = L(G)$.

اثبات: تنها چارچوب کلی اثبات را شرح می‌دهیم. با داشتن یک گرامر خطی از چپ با قوانینی به فرم

$$A \to Bv,$$

یا

$$A \to v,$$

و با جایگذاری هر یک از قوانین

$$A \to v^R B,$$

یا

$$A \to v^R$$

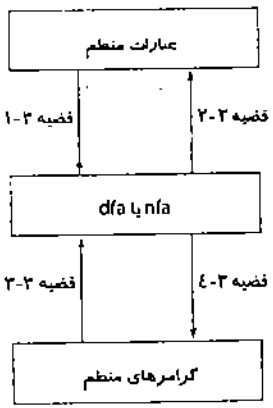
در G، گرامر خطی از راست $\hat{G}$ را ایجاد می‌کنیم. با چند مثال ساده، به راحتی می‌توان فهمید که $L(G) = (L(\hat{G}))^R$. سپس، با استفاده از تمرین ۱۰ بخش ۲-۳ درمی‌یابیم که معکوس هر زبان منظم، منظم است. به دلیل خطی از راست بودن $\hat{G}$، $L(\hat{G})$ منظم است. پس $L((\hat{G}))^R$ و $L(G)$ نیز منظم هستند. ■

با کنار هم قراردادن قضایای ۳-۴ و ۳-۵، به هم‌ارزی زبان‌های منظم و گرامرهای منظم می‌رسیم.

قضیه ۳-۶

زبان L منظم است اگر و تنها اگر گرامر منظم G وجود داشته باشد بطوریکه $L = L(G)$ باشد. ■

اینک با روشهای مختلف توصیف زبان‌های منظم، یعنی استفاده از dfaها، nfaها، عبارات منظم و گرامرهای منظم آشنا شده‌اید. هر چند در برخی مثال‌ها یکی از این روش‌ها مناسب‌تر از بقیه است، ولی در عمل همگی قدرت برابری داشته و هر یک تعریف کامل و شفافی از یک زبان منظم ارائه می‌دهند. شمایی از ارتباط بین این مفاهیم را که در قالب چهار قضیه در این فصل آمده است، در شکل ۳-۱۹ مشاهده می‌کنید.



شکل ۳-۱۹

شکل ۳-۱۹

تمرین‌ها

۱. dfaای بسازید که زبان تولید شده بوسیله گرامر زیر را پذیرش کند:

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

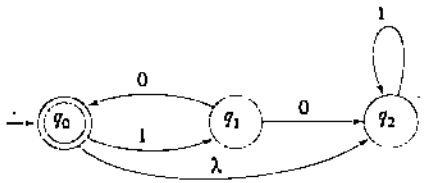
۲. گرامر منظمی بنویسید که زبان $L(aa^*(ab+a)^*)$ را تولید کند.

۳. یک گرامر خطی از چپ برای زبان تمرین ۱ بنویسید.

٤. گرامرهای خطی از راست وخطی از چپ برای زبان زیر بنویسید:

$$L = \{a^n b^m : n \geq 2, m \geq 3\}.$$

۵. با استفاده از ساختار قضیه ۳-۴، یک گرامر خطی از چپ برای زبان پذیرفته شده بوسیله nfa زیر بنویسید:



۶. یک گرامر خطی از راست برای زبان $L((aab^*ab)^*)$ بنویسید.

۷. گرامر منظمی بنویسید که زبانی روی $\Sigma = \{a,b\}$ متشکل از تمام رشته‌ها با حداکثر سه a را تولید کند.

۸. در قضیه ۳-۵، اثبات کنید که $L(\hat{G}) = (L(G))^R$.

۹. ساختاری پیشنهاد کنید که بتوان بوسیله آن، یک گرامر خطی از راست را مستقیماً از nfa بدست آورد.

۱۰. گرامر خطی از چپ را برای زبان تمرین ۶ ارائه دهید.

۱۱. گرامر منظمی را برای زبان $L = \{a^n b^m : n+m$ زوج است$\}$ بنویسید.

۱۲. گرامر منظمی بنویسید که زبان

$$L = \{w \in \{a,b\}^* : n_a(w) + 3n_b(w) \text{ زوج است}\}$$

را تولید کند.

۱۳. گرامرهای منظمی را برای زبان‌های زیر روی $\{a,b\}$ بنویسید:

الف) $L = \{w : n_a(w)$ و $n_b(w)$ هر دو زوج هستند$\}$.

ب) $L = \{w : (n_a(w) - n_b(w)) \bmod 3 = 1\}$.

ج) $L=\{w:(n_a(w)-n_b(w)) \bmod 3 \neq 1\}$.

د) $L=\{w:(n_a(w)-n_b(w)) \bmod 3 \neq 0\}$.

ه) $L=\{w:$ فرد است $\left|n_a(w)-n_b(w)\right|\}$.

نشان دهید که برای هر زبان منظم فاقد λ، یک گرامر خطی از راست وجود دارد که قوانین آن فقط به فرم

$$A \to aB,$$

یا

$$A \to a,$$

هستند که در آن، $A,B \in V$ و $a \in T$.

نشان دهید که هر گرامر منظم G که در آن $L(G) \neq \varnothing$ باشد، باید دارای حداقل یک قانون به فرم

$$A \to x$$

باشد بطوریکه، $A \in V$ و $x \in T^*$.

گرامر منظمی بنویسید که مجموعه تمام اعداد حقیقی در C را تولید کند.

فرض کنید که $G_1=(V_1,\Sigma,S_1,P_1)$ یک گرامر خطی از راست، $G_2=(V_2,\Sigma,S_2,P_2)$ یک گرامر خطی از چپ و V_1 و V_2 جدا از هم باشند. گرامر خطی $G=(\{S\}\cup V_1\cup V_2,\Sigma,S,P)$ را در نظر بگیرید که در آن، S در $V_1\cup V_2$ وجود نداشته و $P=\{S \to S_1 \mid S_2 \cup P_1 \cup P_2\}$. نشان دهید که $L(G)$ منظم است.

فصل

ویژگی‌های زبان‌های منظم

در فصل‌های قبل، زبان‌های منظم را تعریف کرده، برخی روش‌های ارائه آنها را مورد مطالعه قرار داده و نمونه‌هایی از کاربرد این زبان‌ها را بررسی کردیم. حال این سؤال مطرح می‌شود که زبان‌های منظم در حالت کلی چه ویژگی‌هایی دارند؟ و آیا تمامی زبان‌های صوری منظم هستند؟ انتظار داریم هر مجموعه‌ای توسط یک اتومات متناهی، حتی بسیار پیچیده، پذیرفته شده و در نتیجه منظم تلقی شود. اما همانطور که در ادامه خواهیم دید، در زبان‌های صوری چنین نیست و بنابراین، پاسخ سؤال بالا منفی است. برای درک علت، باید ماهیت زبان‌های منظم را با دقت بیشتری مطالعه نموده و ویژگی‌های کل خانواده را بررسی کنیم.

اولین پرسشی که مطرح می‌کنیم این است که، عملگرهای مختلف چه تأثیری روی زبانهای منظم می‌گذارند؟ منظور ما از عملگرها، عملگرهای ساده‌ای از قبیل الحاق روی مجموعه‌ها و عملگرهایی از قبیل تمرین ۲۴ بخش ۲-۱ است که در اثر اعمال آنها رشته‌های زبان تغییر می‌کنند. آیا زبان حاصل باز هم منظم خواهد بود؟ این مسأله را تحت عنوان مسئله بسته‌بودن(بستار) مطرح می‌کنیم. ویژگی‌های بستار، هر چند غالباً از جنبه نظری اهمیت پیدا می‌کنند، همچنین باعث درک بهتر تمایز بین خانواده زبان‌های مختلف می‌شوند.

گروه دوم سؤالات در مورد خانواده زبان‌ها، اشاره به توانایی ما در تصمیم‌گیری راجع به ویژگی‌هایی از قبیل متناهی یا نامتناهی بودن زبان دارد. لازم به ذکر است که این دست سؤالات به سادگی در مورد زبان‌های منظم پاسخ داده شده‌اند، اما تحقیق این سوالات در دیگر خانواده‌های زبان‌ها چندان آسان نیست.

در پایان، این سؤال مهم را مطرح می‌کنیم که چه معیاری برای تشخیص منظم بودن یا نبودن یک

زبان خاص وجود دارد؟ اگر یک زبان واقعاً منظم باشد، همواره می‌توان بوسیله یک dfa، عبارت منظم یا گرامر منظم آن را ارائه کرد. در غیر اینصورت، باید به دنبال چاره دیگری برای ارائه آن باشیم. یک روش برای اثبات منظم نبودن زبان، مطالعه ویژگی‌های عمومی زبان‌های منظم یا به عبارت دیگر، خصوصیاتی است که در تمامی زبان‌های منظم به چشم می‌خورد. با اطلاع از برخی از این ویژگی‌ها و اثبات عدم وجود آنها در زبان موردنظر، می‌توان منظم نبودن زبان را هم اثبات کرد.

در این فصل، نگاهی به انواع ویژگی‌های زبانهای منظم خواهیم داشت. این ویژگی‌ها اطلاعات قابل ملاحظه‌ای در مورد نقاط ضعف و قوت زبان‌های منظم در اختیار ما قرار می‌دهند. سپس، با بررسی همین سؤالات در خانواده زبان‌های دیگر، به شباهت‌ها و تفاوت‌های این ویژگی‌ها پی برده و به این ترتیب، به سراغ مقایسه خانواده زبان‌های منظم می‌رویم.

ویژگی‌های بستاری زبان‌های منظم

حال این پرسش را مطرح می‌کنیم که در صورت منظم بودن دو زبان L_1 و L_2، آیا اجتماع آنها هم منظم است؟ برای زبان‌های مشخص، پاسخ این پرسش معلوم و "بلی" است؛ اما در این بخش می‌خواهیم مسأله را با دید جامع‌تری بررسی کنیم. آیا این پاسخ در مورد تمام L_1 و L_2های منظم صدق می‌کند؟ در جواب باید با اطمینان بله گفت و براساس آن می‌گوئیم که خانواده زبان‌های منظم تحت اجتماع بسته است. طرح سؤالاتی این چنین در مورد انواع عملگرها روی زبان‌ها، ما را به سمت مطالعه ویژگی‌های کلی بستار سوق می‌دهد.

ویژگی‌های بستار خانواده زبان‌های مختلف، تحت عملگرهای مختلف، از بعد نظری اهمیت دارد. در نگاه اول، ممکن است اهمیت کاربردی این ویژگی‌ها چندان مشخص نباشد. البته باید اعتراف کرد که برخی از این ویژگی‌ها، نتایج ناچیزی دارند، اما بسیاری از همین نتایج محدود هم مفید هستند. ویژگی‌های بستاری ضمن توسعه دید ما در مورد ماهیت عمومی خانواده زبان‌ها، پاسخگویی به دیگر انواع سؤالات کاربردی‌تر را تسهیل می‌کنند. در ادامه همین فصل با شرح برخی نمونه‌ها (قضیه ۴-۷ و مثال ۴-۱۳) موارد فوق را روشن می‌کنیم.

بستار تحت عملگرهای ساده روی مجموعه‌ها

کار را با بررسی بستار زبان‌های منظم تحت عملگرهای ساده روی مجموعه‌ها از جمله اجتماع و اشتراک آغاز می‌کنیم.

قضیه ٤-١

اگر L_1 و L_2 زبان‌های منظمی باشند، آنگاه زبان‌های $L_1 \cup L_2$، $L_1 \cap L_2$، L_1L_2، $\overline{L_1}$ و L_1^* منظم خواهند بود. می‌گوییم که خانواده زبان‌های منظم تحت اجتماع، اشتراک، الحاق، مکمل‌گیری و بستارستاره‌ای بسته هستند.

اثبات: اگر L_1 و L_2 منظم باشند، آنگاه عبارات منظم r_1 و r_2 وجود دارند که $L_1 = L(r_1)$ و $L_2 = L(r_2)$. براساس تعریف $r_1 + r_2$، r_1r_2، r_1^* نیز عبارات منظمی هستند که به ترتیب، به زبان‌های $L_1 \cup L_2$، L_1L_2 و L_1^* اشاره می‌کنند. بنابراین، بسته بودن تحت اجتماع، الحاق و بستارستاره‌ای به همین صورت است.

برای اثبات بسته بودن تحت مکمل‌گیری، dfa ای به صورت $M = (Q, \Sigma, \delta, q_0, F)$ درنظرمی‌گیریم که L_1 را می‌پذیرد. بنابراین dfa ای به صورت

$$\hat{M} = (Q, \Sigma, \delta, q_0, Q - F)$$

$\overline{L_1}$ را می‌پذیرد. اثبات این امر به راحتی قابل انجام است و قبلاً نتیجه آن را در تمرین ۴ بخش ۲-۱ ارائه کرده‌ایم. توجه داشته باشید که در تعریف dfa، δ^* را تابع تام فرض کرده و بنابراین، $\delta^*(q_0, w)$ برای تمامی $w \in \Sigma^*$ قابل تعریف است. در نتیجه، یا $\delta^*(q_0, w)$ حالت پایانی است که در اینصورت $w \in L$ و یا $\delta^*(q_0, w) \in Q - F$ و بنابراین، $w \in \overline{L}$.

اثبات بسته بودن تحت اشتراک کمی مشکل‌تر است. فرض کنیم $L_1 = L(M_1)$ و $L_2 = L(M_2)$ که در آن، $M_1 = (Q, \Sigma, \delta_1, q_0, F_1)$ و $M_2 = (P, \Sigma, \delta_2, p_0, F_2)$ دو dfa باشند. با ترکیب M_1 و M_2، اتومات $\hat{M} = (\hat{Q}, \Sigma, \hat{\delta}, (q_0, p_0), \hat{F})$ را می‌سازیم که مجموعه حالت آن، یعنی $\hat{Q} = Q \times P$، متشکل از زوج‌های (q_i, p_j) می‌باشد؛ بعلاوه، تابع انتقال آن، یعنی $\hat{\delta}$، به گونه‌ای است که وقتی M_1 در حالت q_i و M_2 در حالت p_j باشد، $\hat{M}$ در حالت (q_i, p_j) قرار می‌گیرد. این نتایج با فرض

$$\hat{\delta}((q_i, p_j), a) = (q_k, p_l),$$

در صورتی که

$$\delta_1(q_i, a) = q_k$$

و

$$\delta_2(p_j, a) = p_l$$

بدست آمده است. $\hat{F}$ را مجموعه تمام (q_i, p_j) تعریف می‌کنیم که $q_i \in F_1$ و $p_j \in F_2$ می باشند. بنابراین، به راحتی می‌توان نشان داد که $w \in L_1 \cap L_2$ اگر و تنها اگر w بوسیله $\hat{M}$ پذیرفته شود. در نتیجه $L_1 \cap L_2$ نیز منظم است. ■

اثبات بسته بودن تحت اشتراک، علاوه بر تأمین نتیجه مورد نظر، بوضوح نحوه ساخت پذیرنده‌های متناهی برای اشتراک زبان‌های منظم را هم نمایش می‌دهد و به همین دلیل، نمونه خوبی از اثبات‌های ساختاری محسوب می‌شود. اثبات‌های ساختاری در سرتاسر این کتاب به چشم می‌خورند و از این نظر حائز اهمیت هستند که باعث درک بهتر نتایج شده و غالباً بعنوان نقطه آغازی برای ارائه الگوریتم‌های عملی تلقی می‌شوند. اما باید توجه کرد که در بسیاری از موارد و از جمله این مورد، استدلال‌های کوتاه اما غیرآموزنده (و یا نه چندان آموزنده) هم وجود دارند. برای اثبات بسته بودن

تحت اشتراک، اثبات را با قانون دمورگان و معادله ۱-۳ آغاز کرده و از طرفین معادله مکمل می‌گیریم. بنابراین، به ازای تمامی زبانهای L_1 و L_2

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

حال اگر L_1 و L_2 منظم باشند، آنگاه براساس بسته بودن تحت مکمل، $\overline{L_1}$ و $\overline{L_2}$ نیز منظم خواهند بود. پس براساس بسته‌بودن تحت اجتماع، $\overline{L_1} \cup \overline{L_2}$ نیز منظم هستند. دوباره با استفاده از بسته‌بودن تحت مکمل مشاهده می‌کنیم که،

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

منظم است.

در مثال زیر نمونه دیگری ارائه شده است.

نشان دهید که خانواده زبان‌های منظم تحت تفاضل بسته هستند. به بیان دیگر، می‌خواهیم نشان دهیم که اگر L_1 و L_2 منظم باشند، آنگاه $L_1 - L_2$ نیز لزوماً منظم می‌باشد.

موجودیت مجموعه مورد نظر براساس تعریف تفاضل مجموعه‌ها مشخص می‌شود، یعنی

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

منظم بودن L_2، به معنای منظم بودن $\overline{L_2}$ نیز می‌باشد. براساس بسته بودن زبان‌های منظم تحت اشتراک، می‌دانیم که $L_1 \cap \overline{L_2}$ هم منظم است و به این ترتیب اثبات صورت می‌گیرد.

انواع دیگر خواص بسته‌بودن را می‌توان مستقیماً از اثبات‌ها و استدلال‌های پایه نتیجه‌گیری کرد.

خانواده زبان‌های منظم تحت معکوس بسته است.

اثبات: اثبات این قضیه بعنوان تمرین در بخش ۲-۳ ارائه شده است. بنابراین، در اینجا به جزئیات می‌پردازیم. L را یک زبان منظم فرض کرده و nfa متناظر را فقط با یک حالت پایانی، برای آن می‌سازیم. براساس تمرین ۷ بخش ۲-۳ عملیات ساختن nfa همواره امکان‌پذیر است. در گراف انتقال این nfa، رأس شروع را به رأس پایانی و رأس پایانی را به رأس شروع تغییر داده و جهت تمام یالها را معکوس می‌کنیم. به راحتی می‌توان اثبات کرد که اگر و تنها اگر nfa اصلی w را بپذیرد، nfa جدید هم w^R را می‌پذیرد. بنابراین، nfa جدید L^R را می‌پذیرد وبسته بودن تحت معکوس اثبات می‌شود.

بسته بودن تحت دیگر عملگرها

علاوه بر عملگرهای استاندارد روی زبان‌ها، می‌توان عملگرهای دیگری را نیز تعریف کرده و خواص بسته بودن روی آنها را تحقیق کرد. از بین انواع مختلف اینگونه عملگرها، فقط دو مورد را انتخاب کرده و مابقی را در قالب تمرین‌ها در انتهای همین بخش مطرح می‌کنیم.

تعریف ۳-۱

Σ و Γ را دو الفبا فرض کنید. آنگاه تابع

$$h : \Sigma \to \Gamma^*$$

اصطلاحاً همریختی نامیده می‌شود. همریختی را می‌توان یک نوع جایگزینی تعریف کرد که در آن، به جای هر سمبل، از یک رشته استفاده می‌شود. اعضای دامنه تابع h برطبق ضابطه‌ای مشخص به رشته‌ها تبدیل می‌شود؛ اگر

$$w = a_1 a_2 \cdots a_n,$$

آنگاه

$$h(w) = h(a_1)h(a_2)\cdots h(a_n).$$

اگر L زبانی روی Σ باشد، آنگاه **تصویر همریخت** با

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

تعریف می‌شود.

مثال ۳-۴

فرض کنیم $\Sigma = \{a,b\}$ و $\Gamma = \{a,b,c\}$ باشند. اگر h را به صورت

$$h(a) = ab, \quad h(b) = bbc,$$

تعریف کنیم، آنگاه $h(aba) = abbbcab$. تصویر همریخت $L = \{aa, aba\}$ زبان $h(L) = \{abab, abbbcab\}$ خواهد بود.

اگر عبارت منظم r مربوط به زبان L را داشته باشیم، آنگاه یک عبارت منظم برای $h(L)$ می‌تواند به سادگی با انجام عمل همریختی برروی هر سمبل Σ از r بدست آید.

مثال ۳-۴

فرض کنید $\Sigma = \{a,b\}$ و $\Gamma = \{b,c,d\}$. h را بصورت

$$h(a) = dbcc,$$
$$h(b) = bdc,$$

تعریف می‌کنیم. اگر L زبان منظمی باشد که عبارت منظم

$$r=(a+b^*)(aa)^*,$$

آنرا معرفی کند، آنگاه عبارت منظم

$$r_1=(dbcc+(bdc)^*)(dbccdbcc)^*$$

زبان منظم $h(L)$ را معرفی می‌کند.

■

نتیجه کلی بسته بودن زبان‌های منظم تحت همریختی‌های دیگر، بطور مشابه اثبات می‌شود.

قضیه ٤-٣

فرض کنیم h یک همریختی باشد. اگر L یک زبان منظم باشد، آنگاه تصویر همریختی آن یعنی $h(L)$، منظم خواهد بود. بنابراین، خانواده زبان‌های منظم تحت همریختی مورد نظر بسته است.

اثبات: فرض کنیم L زبان منظمی باشد که بوسیله عبارت منظم r، معرفی می‌شود. $h(r)$ با جایگزینی $h(a)$ به ازای هریک از سمبل‌های $a\in\Sigma$ موجود در r بدست می‌آید. براساس تعریف عبارات منظم می‌توان نشان داد که عبارت منظم حاصل، منظم خواهد بود. به همین ترتیب می‌توان مشاهده کرد که عبارت حاصل، $h(L)$ را معرفی می‌کند. فقط لازم است نشان دهیم که به ازای هر $w\in L(r)$، $h(w)$ نظیر در $L(h(r))$ وجود داشته و بالعکس، به ازای هر v عضو $L(h(r))$ یک w عضو L وجود دارد، بطوریکه $v=h(w)$. جزئیات را به عنوان تمرین به شما واگذار می‌کنیم و تنها ادعا می‌کنیم که $h(L)$ منظم است. ■

تعریف ٤-٢

فرض کنیم L_1 و L_2 زبان‌هایی روی یک الفبای یکسان باشند. آنگاه **تقسیم راست** L_1 بر L_2 به صورت

$$L_1/L_2=\{x:\ y\in L_2 \text{ برای برخی } xy\in L_1\} \qquad (۱-۴)$$

تعریف می‌شود.

برای تعیین تقسیم راست L_1 بر L_2، تمام رشته‌های موجود در L_1 با پسوندهای متعلق به L_2 را در نظر می‌گیریم. هر رشته با این فرض، پس ازحذف پسوند مذکور، متعلق به L_1/L_2 خواهد بود.

مثال ۴-۴

اگر

$$L_1=\{a^nb^m:n\geq 1,m\geq 0\}\cup\{ba\}$$

و

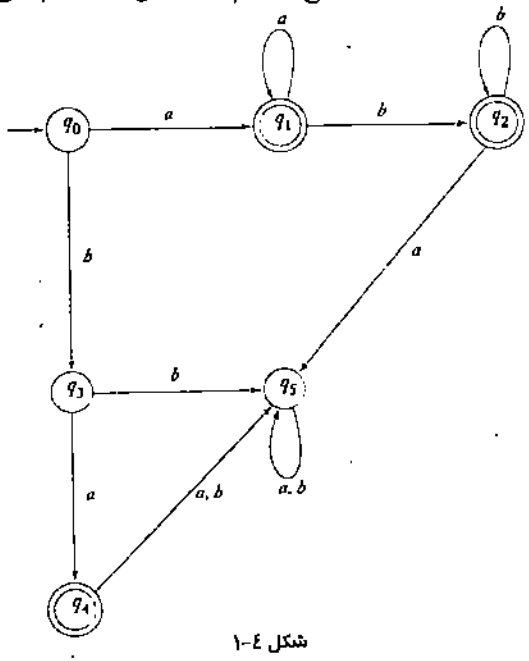
$$L_2=\{b^m:m\geq 1\},$$

آنگاه،

$$L_1/L_2=\{a^nb^m:n\geq 1,m\geq 0\}.$$

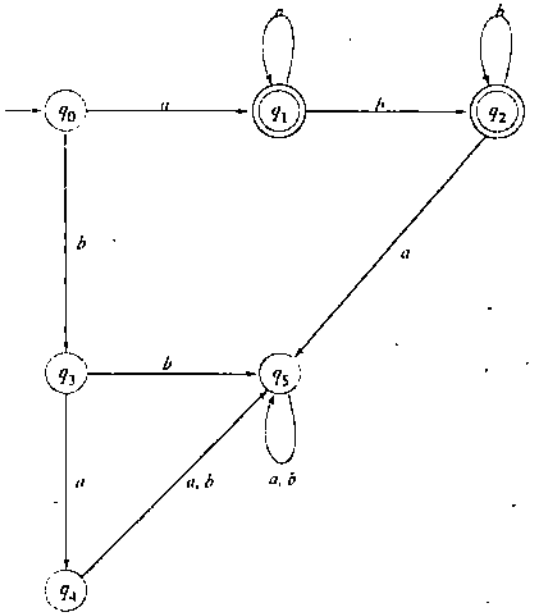
رشته‌های موجود در L_2 از یک یا چند b تشکیل شده‌اند. بنابراین، با حذف یک یا چند b از رشته‌های عضو L_1 که به حداقل یک b ختم می‌شوند، به جواب می‌رسیم.

توجه کنید که در اینجا L_1، L_2 و L_1/L_2 منظم هستند و بنابراین می‌توان نتیجه گرفت که تقسیم راست هر دو زبان منظم، منظم است. این ادعا را در قضیه بعد توسط استدلالی قدم به قدمی، اثبات می‌کنیم که با گرفتن dfaهای مربوط به L_1 و L_2، dfa مورد نظر برای L_1/L_2 را می‌سازد. پیش از تشریح کامل ساختار، ابتدا کاربرد آنرا طی یک مثال بررسی می‌کنیم. برای این کار، ابتدا dfa مربوط به L_1، یعنی اتومات $M_1=(Q,\Sigma,\delta,q_0,F)$ در شکل ۴-۱ را، در نظر می‌گیریم. بدلیل آنکه اتومات مربوط به L_1/L_2 باید تمامی پیشوندهای رشته‌های موجود در L_1 را بپذیرد، M_1 را به گونه‌ای اصلاح می‌کنیم که درصورت وجود y که در (۴-۱) صدق کند، x را بپذیرد. اما در تعیین وجود یا عدم وجود y بطوریکه $xy\in L_1$ و $y\in L_2$ به مشکل برمی‌خوریم. برای حل مشکل، با درنظرگرفتن هر $q\in Q$، مشخص می‌کنیم که آیا قدمی به یکی از حالات پایانی با برچسب v وجود دارد که $v\in L_2$ باشد. در صورت مثبت بودن جواب، هر x که $\delta(q_0,x)=q$ باشد، در L_1/L_2 نیز وجود خواهد داشت. اتومات را به این صورت اصلاح می‌کنیم تا q تبدیل به حالت پایانی شود.



شکل ٤-١

برای انجام روش فوق در این مثال، هر یک از حالت‌های q_0,q_1,q_2,q_3,q_4,q_5 را بررسی می‌کنیم تا ببینیم آیا قدمی با برچسب bb^* به هر یک از حالتهای q_1,q_2 یا q_4 وجود دارد یا خیر. مشاهده می‌کنیم که فقط q_1 و q_2 این ویژگی را دارا بوده و q_0,q_3,q_4 فاقد آن هستند. اتومات حاصل برای L_1/L_2 در شکل ۴-۲ نمایش داده شده است. آن را بررسی می‌کنیم تا به درستی این ساختار پی ببریم. این ایده در قالب قضیه بعد تعمیم یافته است.



شکل ٤-٢

■

قضیه ٤-٤

اگر L_1 و L_2 زبان‌های منظم باشند، آنگاه L_1/L_2 نیز منظم خواهد بود. می‌گوییم که خانواده زبان‌های منظم تحت تقسیم راست بر یک زبان منظم، بسته است.

اثبات: فرض کنیم $L_1=L(M)$، که در آن $M=(Q,\Sigma,\delta,q_0,F)$ یک dfa است. dfa دیگر $\hat{M}=(Q,\Sigma,\delta,q_0,\hat{F})$ را به شرح زیر می‌سازیم. به ازای هر $q_i\in Q$، تعیین کنید آیا $y\in L_2$ وجود دارد که

$$\delta^*(q_i,y)=q_f\in F.$$

ابتدا، dfa ی $M_i=(Q,\Sigma,\delta,q_i,F)$ را بررسی می‌کنیم. اتومات M_i همان اتومات M است که حالت شروع q_0 آن با q_i تعویض شده است. حال مشخص می‌کنیم آیا y وجود دارد که هم

در $L(M_i)$ و هم در L_2 موجود باشد. برای اینکار، می‌توان با استفاده از استدلال اشتراک در زبان منظم که در قضیه ۴-۱ مطرح شد، گراف انتقال را برای $L_2\cap L(M_i)$ پیدا کرد. در صورت وجود حتی یک مسیر بین رأس شروع و هریک از رئوس پایانی گراف مذکور، $L_2\cap L(M_i)$ غیرتهی خواهد بود. در اینصورت، q_i را به $\hat{F}$ اضافه می‌کنیم. با ادامه همین روش برای هر $q_i\in Q$، $\hat{F}$ را تعیین کرده و به این ترتیب، $\hat{M}$ را می‌سازیم.

برای اثبات $L(\hat{M})=L_1/L_2$، x را عضوی از L_1/L_2 در نظر می‌گیریم. آنگاه باید $y\in L_2$ وجود داشته باشد بطوریکه $xy\in L_1$. این ویژگی بطور ضمنی به این معناست که

$$\delta^*(q_0,xy)\in F,$$

و می‌دانیم که $q\in Q$ وجود دارد که

$$\delta^*(q_0,x)=q$$

و

$$\delta^*(q,y)\in F.$$

بنابراین، براساس مراحل ساخت، $q\in\hat{F}$ و از آنجا که $\delta^*(q_0,x)$ عضو $\hat{F}$ است، $\hat{M}$ نیز x را پذیرش می‌کند.

بالعکس، به ازای هر x که در $\hat{M}$ پذیرش می‌شود، داریم که

$$\delta^*(q_0,x)=q\in\hat{F}.$$

دوباره براساس مراحل ساخت، بدیهی است که $y\in L_2$ وجود دارد بطوریکه $\delta^*(q,y)\in F$. بنابراین xy عضو L_1 و x عضو L_1/L_2 می‌باشد. بنابراین می‌توان نتیجه گرفت که

$$L(\hat{M})=L_1/L_2,$$

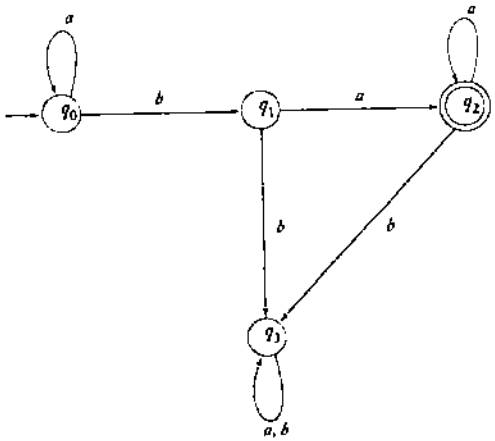
و در نتیجه، L_1/L_2 منظم است.

برای

$$L_1=L(a^*baa^*),$$
$$L_2=L(ab^*)$$

L_1/L_2 را پیدا کنید.

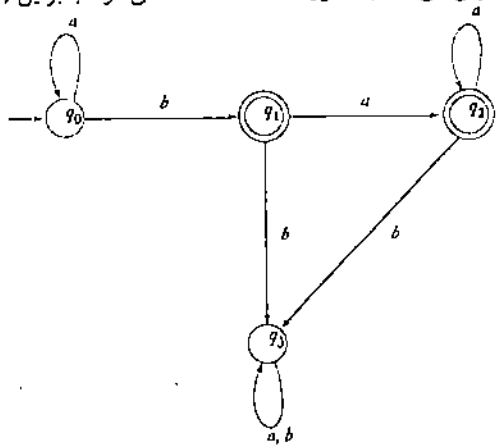
ابتدا یک dfa پیدا می‌کنیم که L_1 را بپذیرد. آن را در شکل ۴-۳ مشاهده می‌کنید. به دلیل راحتی ساخت dfa ، از ذکر مقدمات ساخت صرفنظر می‌کنیم. به راحتی از گراف شکل ۴-۳ به راحتی می‌توان مشاهده کرد که



شکل ٤-۳

$$L(M_0)\cap L_2=\varnothing,$$
$$L(M_1)\cap L_2=\{a\}\neq\varnothing,$$
$$L(M_2)\cap L_2=\{a\}\neq\varnothing,$$
$$L(M_3)\cap L_2=\varnothing.$$

به این ترتیب، اتومات پذیرنده L_1/L_2، مطابق شکل ۴-۴، بدست می‌آید. این اتومات که عبارت منظم $a^*b+a^*baa^*$ آن را معرفی می‌کند، به صورت a^*ba^* خلاصه می‌شود. بنابراین $L_1/L_2=L(a^*ba^*)$.



شکل ٤-٤

■

تمرین‌ها

۱. جزئیات اثبات ساختاری(قدم به قدم و الگوریتمی) بسته بودن تحت عمل اشتراک را برای قضیه ۴-۱ کامل کنید.

۲. با استفاده از استدلال ساختاری قضیه ۴-۱، nfa هایی را رسم کنید که زبان‌های زیر را پذیرش کند:

الف) $L((a+b)a^*)\cap L(baa^*)$.

ب) $L(ab^*a^*)\cap L(a^*b^*a)$.

۳. هرچند در مثال ۴-۱ بسته بودن تحت تفاضل زبان‌های منظم را نشان دادیم، این اثبات ساختاری نبود. استدلال ساختاری برای این نتیجه براساس روش مورد استفاده در استدلال اشتراک در قضیه ۴-۱ ارائه دهید.

٤. در اثبات قضیه ۴-۳، نشان دهید که $h(r)$ یک عبارت منظم است. سپس، نشان دهید که $h(r)$ به $h(L)$ دلالت می‌کند.

۵. نشان دهید که خانواده زبان‌های منظم تحت اجتماع و اشتراک متناهی بسته هستند؛ 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 در مجموعه S_1 و S_2 به صورت

$$S_1\Theta S_2=\{x : x\in S_1 \text{ یا } x\in S_2 \text{، اما } x \text{ هم در } S_1 \text{ و هم در } S_2 \text{ نباشد}\}$$

تعریف می‌شود. نشان دهید که خانواده زبان‌های منظم تحت تفاضل متقارن بسته است.

۷. nor دو زبان به صورت

$$nor(L_1,L_2)=\{w : w\notin L_2 \text{ و } w\notin L_1\}$$

تعریف می‌شود. نشان دهید که خانواده زبان‌های منظم تحت عملیات nor بسته است.

۸. جفت مکمل یا (cor) دو زبان به صورت

$$cor(L_1,L_2)=\{w : w\in \overline{L_1} \text{ یا } w\in \overline{L_2}\}$$

تعریف می‌شود. نشان دهید که خانواده زبان‌های منظم تحت عملیات cor بسته است.

۹. کدامیک از موارد زیر به ازای تمام زبان‌های منظم و به ازای تمام هم‌ریختی‌ها برقرار است؟

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

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

ج) $h(L_1 L_2) = h(L_1)h(L_2)$.

۱۰. $L_1 = L(a^*baa^*)$ و $L_2 = L(aba^*)$ را فرض می‌کنیم، L_1/L_2 را بدست آورید.

۱۱. نشان دهید که $L_1 = L_1 L_2 / L_2$ لزوماً به ازای تمامی زبانهای L_1 و L_2 برقرار نمی‌باشد.

۱۲. *- فرض کنید که $L_1 \cup L_2$ منظم و L_1 متناهی باشد. آیا می‌توان نتیجه گرفت که L_2 نیز منظم است.

۱۳. اگر L یک زبان منظم باشد، اثبات کنید که $L_1 = \{uv : u \in L, |v| = 2\}$ نیز منظم است.

۱۴. اگر L یک زبان منظم باشد، اثبات کنید که $L_1 = \{uv : u \in L, v \in L^R\}$ نیز منظم است.

۱۵. تقسیم چپ زبان L_1 نسبت به L_2 به صورت

$$L_1/L_2 = \{y : x \in L_2 \text{ برای برخی } xy \in L_1\}$$

تعریف می‌شود. نشان دهید که خانواده زبان‌های منظم تحت تقسیم چپ بسته است.

۱۶. نشان دهید که اگر گزاره " اگر L_1 و $L_1 \cup L_2$ منظم باشند، آنگاه L_2 نیز حتماً منظم است" به ازای تمامی L_1 و L_2ها برقرار باشد، آنگاه تمامی زبان‌ها منظم خواهند بود.

۱۷. *tail* یک زبان به صورت مجموعه تمام پسوندهای رشته‌های آن زبان تعریف می‌شود؛ یعنی

$$tail(L) = \{y : x \in \Sigma^* \text{ برای برخی } xy \in L\}.$$

نشان دهید که اگر L منظم باشد، $tail(L)$ نیز منظم خواهد بود.

۱۸. *head* یک زبان مجموعه تمام پیشوندهای رشته‌های آن زبان است؛ یعنی،

$$head(L) = \{x : y \in \Sigma^* \text{ برای برخی } xy \in L\}.$$

نشان دهید که خانواده زبان‌های منظم تحت این عملیات بسته است.

۱۹. عملیات *third* روی رشته‌ها به صورت

$$third(a_1a_2a_3a_4a_5a_6\cdots) = a_3a_6\cdots$$

تعریف کرده که آن را به صورتی مناسب برای زبان‌ها نیز تعمیم می‌دهیم. بسته بودن خانواده زبان‌های منظم تحت این عملیات را اثبات کنید.

۲۰. برای رشته $a_1a_2\cdots a_n$ عملیات *shift* را با عملکرد

$$shift(a_1a_2\cdots a_n) = a_2\cdots a_na_1.$$

تعریف می‌کنیم و این عمل را روی یک زبان به صورت

$$shift(L) = \{v : w \in L \text{ برای برخی } v = shift(w)\}$$

تعریف می‌کنیم. نشان دهید که منظم بودن زبان تحت عملیات *shift* حفظ می‌شود.

۲۱. دو رابطه

$$exchange(a_1a_2\cdots a_{n-1}a_n) = a_na_2\cdots a_{n-1}a_1,$$

و

$$exchange(L) = \{v : w \in L \text{ به ازای برخی } v = exchange(w)\}$$

را تعریف می‌کنیم. نشان دهید که خانواده زبان‌های منظم تحت *exchange* بسته است.

۲۲. *_عملیات *Shuffle* دو زبان L_1 و L_2 به صورت

$$shuffle(L_1, L_2) = \{w_1v_1w_2v_2\cdots w_mv_m :$$
$$w_1, v_i \in \Sigma^* \text{ برای هر } w_1w_2\cdots w_m \in L_1, v_1v_2\cdots v_m \in L_2\}$$

تعریف می‌شود. استدلال کنید که خانواده زبان‌های منظم تحت عملیات *shuffle* بسته است.

۲۳. *_عملیات *minus5* روی زبان L را، مجموعه تمام رشته‌های L تعریف می‌کنیم که پنجمین سمبل از سمت چپ آنها حذف شده باشد (رشته‌های با طول کمتر از پنج تغییر نمی‌کنند). نشان دهید که خانواده زبان‌های منظم تحت عملیات *minus5* بسته است.

۲۴. *_عملیات *leftside* روی L را به صورت

$$leftside(L) = \{w : ww^R \in L\}$$

تعریف می‌کنیم. آیا خانواده زبان‌های منظم تحت این عملیات بسته است؟

۲۵. *min* زبان L به صورت

$$\min(L) = \{w \in L : w = uv \text{ وجود نداشته باشد بطوریکه } u \in L, v \in \Sigma^+\}$$ هیچ

تعریف می‌کنیم. نشان دهید که خانواده زبان‌های منظم تحت عملیات *min* بسته است.

۲۶. G_1 و G_2 را دو گرامر منظم در نظر می‌گیریم. گرامرهای منظمی برای زبان‌های زیر ارائه دهید:

الف) $L(G_1) \cup L(G_2)$.

ب) $L(G_1)L(G_2)$.

ج) $L(G_1)^*$.

سؤالات مقدماتی در مورد زبان‌های منظم

حال این پرسش بسیار اساسی را مطرح می‌کنیم که با داشتن زبان L و رشته w، آیا می‌توان عضویت یا عدم عضویت w در L را تعیین کرد؟ در این سؤال، مسأله **عضویت** مطرح شده و روش پاسخ دادن به آن اصطلاحاً الگوریتم عضویت* خوانده می‌شود. زبان‌هایی که نتوان الگوریتم‌های عضویت مؤثری برای آنها ارائه کرد، چندان کارآمد نیستند. مسأله عضویت و ماهیت الگوریتم‌های عضویت در مباحث بعدی ما نقش مهمی دارند. این بحث علیرغم دشواری ذاتی خود، در زمان کاربرد در زبانهای منظم ساده‌تر می‌شود.

ابتدا بیایید منظور دقیق خود را از عبارت "زبانی برای ... بنویسید" که در بسیاری از استدلال‌ها مطرح می‌شود، روشن کنیم. هرچند تا به اینجا از روش‌های مختلفی از قبیل توصیف‌های شفاهی غیرصوری، تعریف صوری با مجموعه‌ها، اتوماتای متناهی، عبارات منظم و گرامرهای منظم برای توصیف زبان‌های منظم استفاده کردیم، فقط سه موردآخر به اندازه کافی شرح و بسط پیدا کرده و یا در قضایا استفاده شده‌اند. بنابراین، می‌توان گفت که زبان‌های منظم فقط و فقط در صورتی قابل ارائه استاندارد هستند که بوسیله یک اتوماتای متناهی، عبارت منظم و یا گرامر منظم معرفی شوند.

قضیه ٤-٥

در صورت داشتن تعریف استانداردی از زبان منظم L روی Σ برای $w \in \Sigma^*$، یک الگوریتم برای تعیین وجود یا عدم وجود w در L قابل بررسی است.

اثبات: ابتدا زبان مورد نظر را بوسیله یک dfa نمایش می‌دهیم، سپس بررسی می‌کنیم آیا w بوسیله این اتوماتا پذیرش می‌شود یا خیر. ■

از جمله سؤالات مهم دیگر، می‌توان به متناهی یا نامتناهی بودن یک زبان، تساوی دو زبان و همچنین زیرمجموعه بودن یکی از زبانها برای دیگری، اشاره کرد. پاسخگویی به این سؤالات، حداقل در مورد زبانهای منظم، به راحتی امکان‌پذیر است.

قضیه ٤-٦

با این فرض که زبان به صورت تعریف استاندارد خود داده شده باشد، آیا الگوریتمی برای تعیین تهی بودن و متناهی یا نامتناهی بودن آن وجود دارد؟

اثبات: اگر گراف انتقال dfa یک زبان را داشته باشیم، پاسخ سؤال معلوم است. به این معنا که در صورت وجود حتی فقط یک مسیر ساده از رأس شروع به یکی از رئوس پایانی، زبان تهی نخواهد بود.

برای تشخیص متناهی یا نامتناهی بودن، همه رئوسی را پیدا می‌کنیم که عضو یکی از حلقه‌ها باشند. چنانچه هر یک از این رئوس در مسیر یکی از رئوس شروع به رئوس پایانی واقع باشند، زبان نامتناهی و در غیراینصورت، متناهی خواهد بود. ■

مسأله تساوی دو زبان هم یکی از مسائل مهم و کاربردی محسوب می‌شود. در اغلب موارد تعاریف مختلفی از یک زبان برنامه‌سازی وجود دارد و ما باید بدانیم که آیا این تعاریف، علیرغم ظاهر متفاوت خود، یک زبان یکسان را تعریف می‌کنند یا خیر. پاسخگویی به این سؤال معمولاً چندان آسان نبوده و در زبانهای منظم حتی پیچیده‌تر هم می‌شود. در این مورد، نمی‌توان براساس مقایسه رشته به رشته بحث کرد، چون اینکار فقط در مورد زبانهای متناهی قابل انجام است. همچنین نمی‌توان از عبارات منظم، گرامرهای منظم یا dfaها استفاده کرد. بهترین راه‌حل بهره‌گیری از خواص بسته بودن است.

قضیه ٤-٧

با داشتن توصیف استاندارد دو زبان منظم L_1 و L_2، به کمک الگوریتم می‌توان تعیین کرد که آیا $L_1 = L_2$ یا خیر.

اثبات: با استفاده از L_1 و L_2، زبان

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

را تعریف می‌کنیم. براساس خواص بسته بودن، L_3 منظم بوده و می‌توان برای M dfaای را پیدا کرد که L_3 را پذیرش کند. سپس با داشتن M و با استفاده از الگوریتم قضیه ۴-۶ می‌توان تهی بودن یا نبودن L_3 را تعیین کرد. اما براساس تمرین۸ بخش ۱-۱، مشاهده می‌کنیم که $L_3 = \emptyset$ اگر و تنها اگر $L_1 = L_2$. ■

این نتایج علیرغم واضح و معروف بودن خود، اصول محکمی هستند. در زبان‌های منظم به راحتی می‌توان به سؤالات ناشی از قضایای ۴-۵ تا ۴-۷ پاسخ داد، در حالی که در دیگر خانواده‌های زبان‌ها این کار، همیشه هم چندان آسان نیست. بعداً در موارد متعددی با این دست سؤالات مواجه می‌شویم. به تدریج و در ادامه درس، یافتن پاسخ برای برخی از پرسش‌ها مشکل‌تر و در مواردی، حتی غیرممکن می‌شود.

تمرین‌ها

در همه تمرینهای این بخش فرض کنید که زبان‌های منظم به صورت تعریف استاندارد خود آمده‌اند.

۱. نشان دهید که به ازای wهایی مفروض و هر زبان منظم L_1 و L_2، یک الگوریتم برای تأیید یا رد $w \in L_1 - L_2$ وجود دارد.

۲. نشان دهید که به ازای زبان‌های منظم L_1 و L_2، یک الگوریتم برای مشخص کردن اینکه آیا $L_1 \subseteq L_2$ وجود دارد.

۳. نشان دهید که به ازای هر زبان منظم L، یک الگوریتم برای مشخص کردن اینکه آیا $\lambda \in L$ وجود دارد.

٤. نشان دهید که به ازای هر زبان منظم L_1 و L_2، یک الگوریتم برای تأیید یا رد $L_1 = L_1/L_2$ وجود دارد.

۵. یک زبان در صورتی *دوطرفه* نامیده می‌شود که $L = L^R$. الگوریتمی برای تشخیص زبان‌های منظم دوطرفه پیدا کنید.

۶. الگوریتمی برای مشخص کردن عضویت یا عدم عضویت هر رشته w در زبان L وجود دارد، بطوریکه $w^R \in L$.

۷. الگوریتمی ارائه دهید که با داشتن سه زبان منظم L_1، L_2 و L، درستی یا نادرستی برابری $L = L_1L_2$ را مشخص کند.

۸. الگوریتمی ارائه دهید که با داشتن زبان منظم L، درستی یا نادرستی $L = L^*$ را تعیین کند.

۹. L را یک زبان منظم روی Σ و w را رشته‌ای عضو Σ^* فرض می‌کنیم. الگوریتمی ارائه دهید که مشخص کند، آیا L حاوی رشتهٔ wای است که wزیررشته‌ای از آن باشد؛ یعنی داشته باشیم $w = uvv$ که در آن $u, v \in \Sigma^*$.

۱۰. نشان دهید که الگوریتمی وجود دارد که به ازای هر زبان منظم L، مشخص می‌کند آیا $L = shuffle(L, L)$ است.

۱۱. عملیات $tail(L)$ به صورت

$$tail(L) = \{v : uv \in L, u, v \in \Sigma^*\}.$$

تعریف می‌شود. نشان دهید الگوریتمی برای مشخص کردن درستی یا نادرستی برابری $L = tail(L)$ به ازای هر زبان منظم L وجود دارد ؟

۱۲. L را یک زبان منظم روی $\Sigma = \{a, b\}$ در نظر می‌گیریم. نشان دهید که الگوریتمی برای مشخص کردن اینکه L حاوی رشته‌ای با طول زوج است، وجود دارد.

۱۳. نشان دهید که الگوریتمی وجود دارد که وجود یا عدم وجود رابطه $|L| \geq 5$ در هر زبان منظم L را مشخص کند.

۱۴. الگوریتمی را شرح دهید که مشخص کند که آیا زبان منظم L حاوی تعداد متناهی رشته با طول زوج است یا خیر.

۱۵. الگوریتمی را شرح دهید که با داشتن گرامر منظم G، صحت یا عدم صحت $L(G) = \Sigma^*$ را مشخص کند.

۴-۳ شناسایی زبان‌های نامنظم

همانطور که در بیشتر مثال‌ها مشاهده کردیم، زبان‌های منظم گاهی اوقات نامتناهی هستند. با این وجود، ارتباط زبان‌های منظم با اتوماتای دارای حافظه متناهی، محدودیت‌هایی را بر ساختار زبان‌های منظم وارد می‌کند. برای حفظ خاصیت منظم بودن در زبان‌ها باید برخی محدودیت‌ها را رعایت کرد. براساس شهود، یک زبان فقط در صورتی منظم است که در جریان پردازش رشته‌ها، اطلاعاتی که باید در هر مرحله به خاطر آورده شوند کاملاً محدود باشد. این گفته درست است، اما باید پیش از هر نوع استفاده‌ای به دقت اثبات شود. روش‌های زیادی برای اثبات وجود دارد که در ادامه به برخی از آنها اشاره خواهیم کرد.

استفاده از اصل لانه کبوتری

ریاضی‌دانان از اصطلاح "اصل لانه کبوتری" برای اشاره به این مشاهده ساده استفاده می‌کنند که: اگر n شیء را در m جعبه (لانه کبوتر) قرار دهیم و اگر $n > m$، آنگاه حداقل در یک جعبه بیش از یک شیء قرارمی‌گیرد. از این واقعیت روشن می‌توان نتایج زیادی گرفت.

مثال ۴-۶

آیا زبان $L = \{a^n b^n : n \geq 0\}$ منظم است؟ به کمک برهان خلف نشان می‌دهیم که پاسخ منفی است. فرض کنیم که L منظم باشد. بنابراین، یک dfa به فرم $M = (Q, \{a, b\}, \delta, q_0, F)$ برای آن وجود خواهد داشت. حال $\delta^*(q_0, a^i)$ را به ازای $i = 1, 2, 3, \ldots$ در نظر می‌گیریم. به دلیل نامحدود بودن تعداد iها و در عین حال متناهی بودن تعداد حالتها در M، براساس اصل لانه کبوتری باید تعدادی حالت q وجود داشته باشد که

$$\delta^*(q_0, a^n) = q$$

و

$$\delta^*(q_0, a^m) = q,$$

و در آن $n \neq m$. اما چون M، $a^n b^n$ را می‌پذیرد، باید داشته باشیم:

$$\delta^*(q, b^n) = q_f \in F.$$

بر این اساس، می‌توان نتیجه گرفت که

$$\begin{aligned}\delta^*(q_0, a^m b^n) &= \delta^*(\delta^*(q_0, a^m), b^n) \\ &= \delta^*(q, b^n) \\ &= q_f.\end{aligned}$$

این با فرض اولیه ما، که M فقط در صورتی $a^m b^n$ را می‌پذیرد که $n = m$، در تناقض بوده و به همین دلیل می‌توان نتیجه گرفت که L نمی‌تواند منظم باشد.

■

در این استدلال، اصل لانه کبوتری فقط روشی برای بیان دقیقتر معنای محدودیت حافظه اتومات‌های متناهی بود. برای پذیرش تمامی $a^n b^n$ها، اتومات باید بین تمامی پیشوندهای a^n و a^m تمایز قائل شود. ولی از آنجا که تعداد حالات داخلی، متناهی است، m و n ای وجود دارند که نتوان بین آنها تمایزی قائل شد.

برای استفاده از این نوع استدلال در موارد دیگر، بهتر است آنرا در قالب یک قضیه کلی تدوین کنیم. گرچه روشهای مختلفی برای تعریف این قضیه وجود دارد، روش زیر را می‌توان متداول‌ترین آنها دانست.

لم تزریق

در نتیجه زیر، که به نام لم تزریق برای زبان‌های منظم خوانده می‌شود، از اصل لانه کبوتری به صورتی متفاوت استفاده شده است. اثبات قضیه بر این مشاهده استوار است که در یک گراف انتقال حالت با n رأس، هر قدم به طول n یا بیشتر، حتماً برخی رئوس تکراری و یا به بیان بهتر، حلقه وجود خواهد داشت.

قضیه ٤-۸

فرض کنیم L یک زبان منظم نامتناهی باشد. آنگاه عدد صحیح مثبت m وجود دارد بطوریکه هر $w \in L$ را با شرط $|w| \geq m$ ، می‌توان به صورت

$$w = xyz$$

تجزیه کرد، با فرض

$$|xy| \leq m,$$

و

$$|y| \geq 1,$$

بطوریکه

$$w_i = xy^i z, \qquad (۴-۲)$$

به ازای تمام $i = 0,1,2,\ldots$ عضو L خواهد بود.

به طور خلاصه، تمامی رشته‌های طولانی L را می‌توان به سه بخش تقسیم نمود، بطوریکه تعداد دلخواهی از تکرارهای بخش میانی رشته دیگری را در L ایجاد کند. می‌گوییم که رشته میانی "پمپاژ شده" و به همین دلیل از اصطلاح پمپاژکردن یا تزریق کردن برای بیان نتیجه استفاده می‌کنیم.

اثبات: اگر L منظم باشد، یک dfa برای تشخیص آن وجود دارد. فرض کنیم حالات این dfa با $q_0, q_1, q_2, \cdots, q_n$ نامگذاری شده باشند. اینک رشته w را در L به صورت $|w| \geq m = n+1$ فرض می‌کنیم. به دلیل فرض نامتناهی بودن L ، چنین امکانی همواره وجود خواهد داشت. فرض کنید مجموعه حالات اتومات با پردازش w مشخص می‌شود، یعنی

$$q_0, q_i, q_j, \ldots, q_f.$$

بدلیل آنکه این دنباله دقیقاً $|w|+1$ عضو دارد، حداقل یک حالت تکرار شده و این تکرار باید تا قبل از n امین حرکت انجام شود. بنابراین، دنباله به فرم

$$q_0, q_i, q_j, \ldots, q_r, \ldots, q_r, \ldots, q_f,$$

خواهد بود. به این معنا که باید زیررشته‌های x,y,z در w وجود داشته باشند تا

$$\delta^*(q_0, x) = q_r,$$
$$\delta^*(q_r, y) = q_r,$$
$$\delta^*(q_r, z) = q_f,$$

بطوریکه $|xy| \leq n+1 = m$ و $|y| \geq 1$. بر این اساس، می‌توان نتیجه گرفت که

$$\delta^*(q_0, xz) = q_f,$$

و همچنین

$$\delta^*(q_0, xy^2z) = q_f,$$
$$\delta^*(q_0, xy^3z) = q_f,$$

و الی آخر، که اثبات قضیه کامل می‌شود. ■

تا به اینجا لم تزریق را فقط برای زبان‌های نامتناهی استفاده کردیم. اگرچه زبان‌های متناهی همواره منظم هستند، نمی‌توانند پمپاژ شوند، زیرا پمپاژکردن بطور خودکار باعث ایجاد مجموعه‌های نامتناهی می‌شود، این قضیه برای زبان‌های متناهی نیز برقرار است ولی استفاده از آن بی معنا می‌باشد. به همین دلیل در لم تزریق، m باید بزرگتر از طولانی‌ترین رشته در نظر گرفته شود تا هیچ رشته‌ای قابل پمپاژ نباشد.

از لم تزریق مشابه اصل لانه کبوتری در مثال ۴-۶، برای نشان دادن اینکه زبان‌های خاصی منظم نیستند استفاده می‌شود. اثبات همواره از طریق برهان خلف انجام می‌شود. همانطور که قبلاً هم گفتیم، از لم تزریق نمی‌توان برای اثبات منظم بودن زبان‌ها استفاده کرد. اگر چه می‌توانیم نشان دهیم (و این کار خیلی مشکل است) که هر رشتهٔ پمپاژ شده باید در زبان اولیه وجود داشته باشد و برای این منظور در قضیهٔ ۴-۸ برهانی وجود ندارد که بتوانیم از آن نتیجه بگیریم که زبان مورد بحث منظم است.

مثال ۴-۱۰

با استفاده از لم تزریق نشان دهید که $L = \{a^n b^n : n \geq 0\}$ منظم نیست. زبان L را منظم فرض می‌کنیم تا بتوان برای آن از لم تزریق استفاده کرد. هرچند مقدار عددی m نامعلوم است، اما هر چه که باشد همواره می‌توان $n = m$ را انتخاب کرد. بنابراین، زیررشته y تماماً متشکل از a خواهد بود. فرض کنید که $|y| = k$. آنگاه رشته بدست آمده با استفاده از $i = 0$ در معادله (۴-۲) عبارت است از

$$w_0 = a^{m-k}b^m$$

که حتما عضو L نیست. این نتیجه‌گیری با لم تزریق تناقض داشته و بنابراین، فرض منظم بودن L نادرست است.

■

دربکاربردن لم تزریق باید بخاطر داشته باشیم که لم تزریق چه می‌گوید. ما از وجود m و همچنین تجزیه xyz اطلاع داریم، اما از ماهیت آنها بی‌خبر هستیم. نمی‌توان ادعا کرد که فقط به این دلیل که لم تزریق به ازای برخی مقادیر خاص m یا xyz برقرار نباشد، به تناقض رسیده‌ایم. از طرف دیگر، لم تزریق به ازای تمام $w \in L$ و تمامی i ها صدق می‌کند. بنابراین، اگر لم تزریق حتی برای یک w یا i هم نقض شود، آنگاه زبان نمی‌تواند منظم باشد.

استدلال صحیح برای یک زبان را می‌توان نوعی بازی در برابر حریف تصور کرد. هدف ما یعنی برنده شدن در بازی، به کمک ایجاد یک تناقض در لم تزریق بدست می‌آید. حریف هم تلاش می‌کند تا با ایده‌ای برنامه‌ریزی ما را برهم زند. در این بازی چهار حرکت انجام می‌شود.

۱. حریف m را انتخاب می‌کند.

۲. با در اختیار داشتن m، رشته w از L را با طول بزرگتر یا مساوی m انتخاب می‌کنیم. ما آزاد هستیم هر w را با شرایط $w \in L$ و $|w| \geq m$ انتخاب کنیم.

۳. حریف تجزیه xyz را با شرایط $|xy| \leq m$ و $|y| \geq 1$ انتخاب می‌کند. همواره باید به خاطر داشته باشیم که حریف با انتخاب خود سعی می‌کند به نحوی مانع از برنده شدن ما در بازی شود.

۴. باید i را به گونه‌ای انتخاب کنیم که رشته پمپاژ شده w_i، که بوسیله معادله (۴-۲) تعریف شده، عضو L نباشد. موفقیت در انتخاب i، کلید برنده شدن ما در بازی است.

راهکاری که برنده شدن ما را در بازی، با هر نوع انتخاب حریف، تضمین می‌کند، اثبات منظم نبودن زبان است. در این روش، حرکت شماره ۲ بسیار مهم است. وقتی نتوان حریف را به انتخاب صورت تجزیه شده خاصی از w مجبور کرد، می‌توان w را به صورتی انتخاب کرد که حریف در حرکت ۳ مجبور به انتخاب x,y,z شود. به این ترتیب، می‌توانیم در حرکت بعدی خود، لم تزریق را نقض کنیم.

مثال ۴-۸

نشان دهید که زبان

$$L = \{ww^R : w \in \Sigma^*\}$$

منظم نیست.

برای هر m که حریف در مرحله اول انتخاب می کند، همواره می‌توانیم w را مطابق شکل ۴-۵ انتخاب کنیم. در اثر این انتخاب و شرط $|xy| \leq m$، حریف در مرحله ۳ مجبور به انتخاب y می‌شود که تماماً از a تشکیل شده باشد. در مرحله ۴، از $i = 0$ استفاده می‌کنیم. در رشتهٔ بدست آمده از این طریق، تعداد a های واقع در سمت چپ کمتر از طرف راست بوده و بنابراین از فرم ww^R پیروی نخواهد کرد. در نتیجه، L منظم نیست.

توجه داشته باشیدکه اگر w را بسیار کوتاه انتخاب کنیم، آنگاه حریف می‌تواند y را با تعداد b های زوج انتخاب کند. در اینصورت، لم تزریق در مرحله آخر نقض می‌شود. همچنین اگر رشته‌ای را انتخاب کنیم که تماماً از a تشکیل شده باشد، بطور نمونه

$$w = a^{2m},$$

که عضو L است، باز هم شکست می‌خوریم. حریف برای شکست دادن ما فقط باید

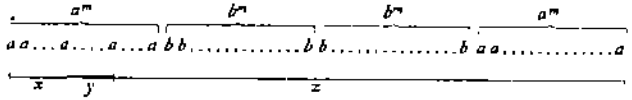
$$y = aa$$

را انتخاب کند. اکنون به ازای هر i، w_i عضو L است و بنابراین، ما شکست می‌خوریم.

نمی‌توان لم تزریق را به امید خطای حریف در انتخاب حرکت استفاده کرد. اگر با انتخاب $w = a^{2m}$ توسط ما، حریف مجبور به انتخاب

$$y = a,$$

شود، آنگاه w_0 رشته‌ای با طول فرد بوده و بنابراین در L وجود ندارد. هر استدلالی با فرض انجام این حرکت توسط حریف، ذاتاً اشتباه خواهد بود.



شکل ۴-۵

■

مثال ۴-۹

فرض کنیم $\Sigma = \{a,b\}$ باشد. زبان

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

منظم نیست.

فرض کنید که m معلوم باشد. بدلیل آنکه در انتخاب w آزادی کامل داریم، ما $w = a^m b^{m+1}$ را انتخاب می‌کنیم. اکنون، بدلیل آینکه $|xy|$ نمی‌تواند بزرگتر از m باشد، حریف هیچ چاره‌ای غیر از انتخاب یک y که تماماً شامل a می‌باشد، ندارد، یعنی

$$y = a^k,\ 1 \leq k \leq m.$$

حال با استفاده از $i = 2$ پمپاژ می‌کنیم. رشته حاصل شدهٔ

$$w_2 = a^{m+k} b^{m+1}$$

عضو L نیست. بنابراین، لم تزریق نقض شده و L منظم نیست.

■

مثال ۴-۱۰

زبان

$$L = \{(ab)^n a^k : n > k, k \geq 0\}$$

منظم نیست.

با معلوم بودن m، رشته

$$w = (ab)^{m+1} a^m,$$

را انتخاب می‌کنیم که عضو L است. به دلیل شرط $|xy| \leq m$، x و y هر دو باید در بخشی از رشته قرار

داشته باشند که از ab تشکیل شده است. بدلیل آنکه انتخاب x تأثیری بر استدلال ندارد، انواع انتخاب‌های y را بررسی می‌کنیم. اگر حریف ما $y=a$ را انتخاب کند و ما $i=0$، رشته حاصل عضو $L((ab)^*a^*)$ نیست. اگر حریف $y=ab$ را انتخاب کند، ما باز هم می‌توانیم $i=0$ را انتخاب کنیم. در اینصورت، رشته $(ab)^m a^m$ بدست می‌آید که عضو L نیست. به همین ترتیب، می‌توانیم در مورد تمام انتخاب‌های احتمالی حریف بحث کرده و در نهایت ادعای خود را اثبات کنیم.

■

مثال ۴-۱۱

نشان دهید که

$$L=\{a^n : n \text{ مربع کامل نیست}\}$$

منظم نیست.

با این فرض که حریف m را انتخاب می‌کند، ما نیز

$$w=a^{m!}$$

را انتخاب می‌کنیم. برای تجزیه $w=xyz$، مشخص است که

$$y=a^k$$

بطوریکه $1\le k\le m$. در این صورت به ازای $i=0$ خواهیم داشت:

$$w_0=a^{m!-k}.$$

اما چون $m^2-k>(m-1)^2$، بنابراین w_0 نمی‌تواند عضو L باشد. در نتیجه، زبان منظم نیست.

■

در بعضی موارد، می‌توان با استفاده از خواص بسته بودن، یک مسأله جدید را با مسأله‌ای که قبلاً در مورد آن بحث شده مرتبط کرد. اینکار تا حدودی ساده‌تر از کاربرد مستقیم لم تزریق است.

مثال ۴-۱۲

نشان دهید که زبان

$$L=\{a^n b^k c^{n+k} : n\ge 0, k\ge 0\}$$

منظم نیست.

گرچه در این مثال لم تزریق را می‌توان به راحتی و مستقیماً بکاربرد، اما بهتر است به جای آن از بسته بودن تحت همریختی استفاده کرد. فرض کنید

$$h(a)=a,\ h(b)=a,\ h(c)=c,$$

آنگاه

اما به دلیل منظم نبودن $h(L)$، L هم نمی‌تواند منظم باشد.

■

مثال ۴-۱۳

نشان دهید که زبان

$$L=\{a^n b^l : n\ne l\}$$

منظم نیست.

در این مثال باید در بکارگیری مستقیم لم تزریق کمی ابتکار به خرج دهیم. بهتر است رشته‌های با طول $n=l+1$ یا $n=l+2$ را انتخاب نکنیم، چون حریف ما می‌تواند تجزیه‌ای را انتخاب کند که نتوان به کمک آن رشته مذکور را از زبان پمپاژ کرد (یعنی، تعداد a ها و b ها برابر باشد). بیایید فرض کنیم که $n=m!$ و $l=(m+1)!$. حال اگر حریف یک y با طول $k<m$ را انتخاب کند(که لزوماً فقط از a تشکیل شده باشد)، i بار تزریق می‌کنیم تا طرف مقابل مجبور به انتخاب رشته $m!+(i-1)k$ از aها شود. اگر i را طوری انتخاب کنیم که

$$m!+(i-1)k=(m+1)!$$

به تناقضی در لم تزریق می‌رسیم. این امکان همواره وجود دارد زیرا

$$i=1+\frac{m\ m!}{k}$$

و $k\le m$، می‌باشد. بنابراین طرف راست یک عدد صحیح است و ما موفق به نقض شرایط لم تزریق شده‌ایم.

اما روش بسیار بهتری هم برای حل این مسأله وجود دارد. فرض کنید L منظم باشد. آنگاه، براساس قضیه ۴-۱، $\overline{L}$ و زبان

$$L_1=\overline{L}\cap(a^*b^*)$$

نیز منظم هستند. با توجه به این که قبلاً نامنظم بودن $L_1=\{a^n b^n : n\ge 0\}$ را نشان دادیم، در نتیجه، L نمی‌تواند منظم باشد.

■

درک مفهوم لم تزریق مشکل بوده و استفاده از آن ممکن است باعث اشتباه ما شود. در ادامه برخی از نقاط ضعف این لم را ارائه می‌دهیم تا در حین استفاده مراقب آنها باشید.

یک اشتباه، مربوط به استفاده از این لم برای اثبات منظم بودن زبان‌هاست. حتی اگر نشان دهیم که هیچ‌کدام از رشته‌های زبان L قبلاً پمپاژ نشده‌اند، هنوز نمی‌توان براساس آن به منظم بودن L پی برد. از لم تزریق فقط می‌توان برای اثبات منظم نبودن زبان‌ها استفاده کرد.

اشتباه دیگر، شروع کار استدلال با رشته‌ای است که عضو L نیست. بعنوان مثال، فرض کنید بخواهیم نشان دهیم که

(۴-۳) $L=\{a^n : n \text{ یک عدد اول است}\}$

منظم نیست. شروع استدلال با جملاتی از قبیل "در صورت مشخص بودن m، فرض کنیم که ... $w=a^m$ " اشتباه است، چون m لزوماً عدد اول نیست. در عوض، باید استدلال را عبارتی مشابه "درصورت معلوم بودن m، فرض کنیم $w=a^M$، که در آن M یک عدد اول بزرگتر از m است" آغاز کنیم.

آخرین و شاید معمول‌ترین اشتباه، تصور هر نوع فرضی درباره تجزیه xyz است. در مورد تجزیه‌ها، تنها می‌توان به پیام لم تزریق به این صورت اشاره کرد که: "y تهی نبوده و بعلاوه، $|xy|\le m$؛ بنابراین، y حتماً در بین m سمبل از الفباء رشته قرار دارد". هر تصوری غیر از این، استدلال را بی‌اعتبار می‌کند. یکی از اشتباهات معمول در اثبات منظم نبودن زبان معادله (۴-۳) ناشی از فرض $y=a^k$ و فرد بودن k است. در نتیجه، بطور قطع $w=xz$ رشته‌ای با طول زوج بوده و بنابراین، عضو L نخواهد بود. این فرض برای k نادرست است و اثبات اشتباه است.

ممکن است با آنکه برای استفاده از لم تزریق و مسائل آن تجربه‌ای کسب کرده باشیم، باز هم تعیین دقیق نحوه کاربرد آن مشکل باشد. لم تزریق مانند وارد شدن به یک بازی با قوانین پیچیده است. هرچند اطلاع از این قوانین ضروری است، اما تنها دانستن این قوانین برای انجام یک بازی خوب کافی نبوده و باید در جریان بازی از راهکار خوبی هم استفاده کنیم. اگر بتوانید لم تزریق را در مورد برخی از مثال‌های مشکل‌تر این کتاب به درستی اعمال کنید، باید واقعاً به شما تبریک گفت.

تمرین‌ها

۱. بیان دیگری از لم تزریق مطرح شده، آن را اثبات کنید؟

اگر L منظم باشد، آنگاه یک m وجود دارد که هر $w\in L$ با طول بزرگتر از m را می‌توان به صورت

$$w=xyz,$$

تجزیه کرد، بطوریکه $|y|\ge 1$ و $|yz|\le m$ و به ازای تمام i ها، xy^iz عضو L وجود باشد.

۲. حالت تعمیم یافته برای لم تزریق به صورت زیر را ثابت کنید که در آن، از قضیه ۴-۸ و همچنین تمرین ۱ بالا بعنوان شرایط ویژه استفاده شده است.

اگر L منظم باشد، آنگاه یک m وجود دارد که شرایط زیر در مورد هر $w\in L$ با طول کافی و هر یک از تجزیه‌های $w=u_1vu_2$ آن برقرار است و $u_1,u_2\in\Sigma^*, |v|\ge m$. رشته میانی v را می‌توان به صورت $v=xyz$ نوشت که در آن، $|y|\ge 1$، $|xy|\le m$، بطوریکه برای تمامی $i=0,1,2,\ldots$، $u_1xy^izu_2\in L$.

۳. نشان دهید که زبان $L=\{w : n_a(w)=n_b(w)\}$ منظم نیست. آیا L^* منظم است؟

۴. اثبات کنید که زبان‌های زیر منظم نمی‌باشند:

الف) $L=\{a^nb^ka^k : k\ge n+1\}$.

ب) $L=\{a^nb^ka^l : k\ne n+1\}$.

ج) $L=\{a^nb^ka^k : n=l \text{ یا } l\ne k\}$.

د) $L=\{a^nb^l : n\le l\}$.

ه) $L=\{w : n_a(w)\ne n_b(w)\}$.

و) $L=\{wtw : w\in\{a,b\}^*\}$.

ز) $L=\{wtwtw^R : w\in\{a,b\}^*\}$.

۵. تعیین کنید آیا زبان‌های زیر روی $\Sigma=\{a\}$ منظم هستند یا خیر.

الف) $L=\{a^n : n\ge 2 \text{ و } n \text{ یک عدد اول است}\}$

ب) $L=\{a^n : n \text{ یک عدد اول نیست}\}$

ج) $L=\{a^n : n=k^3 \text{ به ازای برخی } k\ge 0\}$

د) $L=\{a^n : n=2^k \text{ به ازای برخی } k\ge 0\}$

ه) $L=\{a^n : n \text{ حاصل دو عدد اول است}\}$

و) $L=\{a^n : n \text{ عددی اول است یا حاصل دو یا چند عدد اول است}\}$

ز) L^* با فرض اینکه L همان زبان بخش (الف) است.

۶. تعیین کنید آیا زبان‌های زیر منظم هستند یا خیر.

الف) $L=\{a^nb^m : n\ge 1\}\cup\{a^nb^m : n\ge 1, m\ge 1\}$.

ب) $L=\{a^nb^n : n\ge 1\}\cup\{a^nb^{n+2} : n\ge 1\}$.

۷. نشان دهید که زبان

$$L=\{a^nb^n : n\ge 0\}\cup\{a^nb^{n+1} : n\ge 0\}\cup\{a^nb^{n+2} : n\ge 0\}$$

منظم نیست.

۸. *نشان دهید که زبان

$$L=\{a^nb^{n+k} : n\ge 0, k\ge 1\}\cup\{a^{n+k}b^n : n\ge 0, k\ge 3\}$$

منظم نیست.

۹. آیا زبان $L=\{w\in\{a,b,c\}^* : |w|=3n_a(w)\}$ منظم است؟

۱۰. زبان زیر را در نظر بگیرید

$L=\{a^n : n \text{ عدد جذر کامل نیست}\}$.

*الف) نشان دهید که این زبان در صورت بررسی توسط لم تزریق، منظم نیست.

ب) سپس با استفاده از خواص بسته بودن زبان‌های منظم، نتیجه بدست آمده از قسمت الف را اثبات کنید.

۱۱. *نشان دهید که زبان

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

منظم نیست.

۱۲. با بکار بردن لم تزریق، نتیجه مثال ۴-۱۲ را ثابت کنید.

۱۳. نشان دهید که زبان زیر منظم نیست.

$$L=\{a^nb^k:n>k\}\cup\{a^nb^k:n\neq k-1\}.$$

۱۴. درستی یا نادرستی این عبارت را اثبات کنید. اگر L_1 و L_2 زبان‌های نامنظمی باشند، آنگاه $L_1\cup L_2$ هم نامنظم خواهد بود.

۱۵. زبان‌های زیر را در نظر بگیرید. با استفاده از لم تزریق، منظم بودن یا نبودن هر یک را ثابت کنید. سپس ادعای خود را ثابت کنید.

الف) $L=\{a^nb^la^k:n+l+k>5\}$.

ب) $L=\{a^nb^la^k:n>5,l>3,k\leq l\}$.

ج) $L=\{a^nb^l:n/l$ یک عدد صحیح است$\}$.

د) $L=\{a^nb^l:n+l$ یک عدد اول است$\}$.

ه) $L=\{a^nb^l:n\leq l\leq 2n\}$.

و) $L=\{a^nb^l:n\geq 100,l\leq 100\}$.

ز) $L=\{a^nb^l:|n-l|=2\}$.

۱۶. آیا زبان زیر منظم است؟

$$L=\{w_1cw_2:w_1,w_2\in\{a,b\}^*,w_1\neq w_2\}.$$

۱۷. فرض کنیم L_1 و L_2 زبان‌های منظمی باشند. آیا زبان $L=\{w:w\in L_1,w^R\in L_2\}$ لزوماً منظم است؟

۱۸. اصل لانه کبوتری را برای زبان موجود در مثال ۴-۸ بکار برید.

۱۹. آیا زبان‌های زیر منظم هستند؟

الف) $L=\{uvw^Rv:u,v,w\in\{a,b\}^+\}$.

ب)* $L=\{uvw^Rv:u,v,w\in\{a,b\}^+,|u|\geq|v|\}$.

۲۰. آیا زبان زیر منظم است؟

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

۲۱. فرض کنید P یک مجموعه متناهی اما شمارش‌پذیر باشد، به هر $p\in P$ یک زبان L_p نسبت می‌دهیم. کوچکترین مجموعه شامل هر L_p اجتماعی برروی مجموعه نامتناهی P است که بوسیله $\bigcup_{p\in P}L_p$ نمایش داده می‌شود. با یک مثال نشان دهید که خانواده زبان‌های منظم تحت

اجتماع نامتناهی بسته نیست.

۲۲. در بخش ۳-۲ نشان دادیم که زبان متناظر با هر گراف انتقال تعمیم داده شده، منظم است. زبان متناظر با این چنین گرافی

$$L=\bigcup_{p\in P}L(r_p)$$

است که در آن، P مجموعه تمام قدم‌ها در گراف و r_p عبارت متناظر با قدم p است. مجموعه قدم‌ها نامتناهی است؛ بطوریکه براساس تمرین ۲۱، نمی‌توان نتیجه گرفت که L منظم است. نشان دهید که در این مورد، به دلیل ماهیت خاص P ، اجتماع نامتناهی، منظم است.

۲۳. *- آیا خانواده زبانهای منظم تحت اشتراک نامتناهی بسته است؟

۲۴. فرض کنید می‌دانیم که $L_1\cup L_2$ و L_1 منظم هستند. آیا می‌توان نتیجه گرفت که L_2 هم منظم است؟

۲۵. در زبان کد زنجیره‌ای تمرین ۲۴ بخش ۳-۱، L را مجموعه تمام $w\in\{u,r,l,d\}^*$ در نظر می‌گیریم که مستطیل‌ها را توصیف می‌کنند. نشان دهید که زبان L منظم نیست.

۲۶. فرض کنیم $L=\{a^nb^m:n\geq 100,m\leq 50\}$.

الف) آیا می‌توان با استفاده از لم تزریق منظم بودن L را ثابت کرد؟

ب) آیا می‌توان با استفاده از لم تزریق منظم نبودن L را ثابت کرد؟

در مورد پاسخ‌های خود توضیح دهید.

فصل ۱

زبان‌های مستقل‌ازمتن

در فصل قبل، به این نتیجه رسیدیم که لزوماً تمامی زبان‌ها، منظم نیستند. همچنین خواهیم دید که، زبان‌های منظم در توصیف فقط برخی از الگوهای ساده مفید هستند و نمونه‌های کاربرد زبان‌های نامنظم در همه‌جا به چشم می‌خورد. با تفسیر مجدد برخی مثال‌ها می‌توان به نقش این مجدودیت‌ها در زبان‌های برنامه‌سازی پی برد. اگر در زبان $L = \{a^n b^n : n \geq 0\}$، به جای a از پرانتز چپ و به جای b از پرانتز راست استفاده کنیم، آنگاه رشته‌های پرانتزی از قبیل (()) و ((())) عضو L بوده ولی ()) عضو L نمی‌باشد. بنابراین، زبان فوق فرم ساده‌ای از یک ساختار پرانتزبندی تودرتو را توصیف می‌کند که در زبان‌های برنامه‌سازی وجود دارد، به این معنا که برخی ویژگی‌های زبان‌های برنامه‌سازی نیازمند چیزی غیر از زبان‌های منظم هستند. برای پوشش دادن این ویژگی و سایر ساختارهای پیچیده‌تر باید خانواده زبان‌ها را گسترش دهیم. به همین دلیل زبان‌ها و گرامرهای مستقل‌ازمتن را مطرح می‌کنیم.

این فصل را با تعریف گرامرها و زبان‌های مستقل‌ازمتن آغاز کرده و تعاریف را با ارائه مثال‌های ساده‌ای از آنها به تصویر می‌کشیم. سپس، مبحث مهم عضویت را بررسی کرده و علی‌الخصوص پاسخی برای مسأله مهم اشتقاق‌پذیربودن یا نبودن یک رشته مفروض از گرامر موردنظر ارائه می‌کنیم. بسیاری از ما با زبان‌های طبیعی و توصیف جملات با استفاده از اشتقاق بوسیله گرامرها، که اصطلاحاً تجزیه خوانده می‌شود، آشنا هستیم. تجزیه روشی برای توصیف ساختار جملات است. این شیوه در درک معنای جملات، مثلاً برای ترجمه از یک زبان به زبان دیگر، اهمیت دارد. بعلاوه، درعلوم کامپیوتر، در مفسرها، کامپایلرها و دیگر برنامه‌های مترجم مورد استفاده قرار می‌گیرد.

شاید بتوان زبان‌های مستقل‌ازمتن را بعنوان مهمترین جنبه کاربرد نظریه زبان‌های صوری در زبان‌های برنامه‌سازی مطرح کرد. بسیاری از ویژگی‌های زبان‌های برنامه‌سازی واقعی به بهترین صورت،

بوسیله زبانهای مستقل‌ازمتن قابل توصیف هستند. آنچه نظریه زبان‌های صوری در مورد زبان‌های مستقل‌ازمتن، به ما یاد می‌دهد، کاربردهای مهم در طراحی زبان‌های برنامه‌سازی و همچنین ساخت کامپایلرهای مؤثر و کارآمد می‌باشد. در بخش ۵-۳ مختصراً به این موضوع می‌پردازیم.

۵-۱ گرامرهای مستقل‌ازمتن

قوانین در گرامرهای منظم، به دو فرم مشاهده می‌شوند: در طرف چپ فقط یک متغیر وجود دارد، درحالی‌که طرف راست از فرم خاصی تبعیت می‌کند. برای ساخت برنامه‌های قدرتمندتر، باید تا حدودی از قید این محدودیت‌ها رها شویم. گرامرهای مستقل‌ازمتن در طرف چپ همان محدودیت گرامرهای منظم را دارند ولی در طرف راست آزادتر هستند.

تعریف ۵-۱

گرامر مفروض $G=(V,T,S,P)$ در صورتی **مستقل‌ازمتن** خوانده می‌شود که تمام قوانین P به فرم

$$A \to x,$$

باشند که در آن، $A \in V$ و $x \in (V \cup T)^*$.

زبان L مستقل‌ازمتن نامیده می‌شود اگر و تنها اگر گرامر مستقل‌ازمتن G وجود داشته باشد بطوریکه $L=L(G)$.

تمامی گرامرهای منظم، مستقل‌ازمتن هستند و بنابراین، هر زبان منظمی، یک زبان مستقل‌ازمتن نیز می‌باشد. اما، از مثال‌های ساده‌ای مثل $\{a^nb^n\}$، می‌توان دریافت، برخی از زبان‌ها، نامنظم هستند. قبلاً در مثال ۱-۱۱ نشان دادیم که این زبان بوسیله یک گرامر مستقل‌ازمتن قابل تولید است، بنابراین می‌توان نتیجه گرفت که خانواده زبان‌های منظم یکی از زیرمجموعه‌های محض خانواده زبان‌های مستقل‌ازمتن هستند.

مشخصهٔ گرامرهای مستقل از متن این است که هریک از متغیرهای موجود در سمت چپ قوانین می توانند در فرم های جمله‌ای سمت راست قوانین به هر تعداد و ترتیبی بیایند و سمبل‌های پایانی نقشی در این امر ندارند. بطور کلی شرط مستقل‌ازمتن بودن این است که در سمت چپ قوانین فقط یک متغییر وجود داشته باشد.

مثال‌هایی از زبان‌های مستقل‌ازمتن

مثال ۵-۱

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

$$S \to aSa,$$
$$S \to bSb,$$
$$S \to \lambda.$$

مستقل‌ازمتن است. یک اشتقاق این گرامر بصورت زیر است:

$$S \Rightarrow aSa \Rightarrow aaSaa \Rightarrow aabSbaa \Rightarrow aabbaa.$$

براساس این اشتقاق و نظایر آن به راحتی می‌توان گفت که

$$L(G)=\{vw^R : w \in \{a,b\}^*\}.$$

مطابق مثال ۴-۸ این زبان مستقل‌ازمتن، ولی نامنظم است.

■

مثال ۵-۲

گرامر G با قوانین

$$S \to abB,$$
$$A \to aaBb,$$
$$B \to bbAa,$$
$$A \to \lambda.$$

مستقل‌ازمتن است. زبان تولیدشده از این گرامر بصورت زیر می‌باشد:

$$L(G)=\{ab(bbaa)^n bba(ba)^n : n \geq 0\}.$$

بعنوان تمرین درستی رابطه بالا را ثابت کنید.

■

هردو مثال بالا نمونه گرامرهای مستقل‌ازمتن و در عین حال خطی هستند. ناگفته پیداست که گرامرهای منظم و خطی مستقل‌ازمتن هستند ، ولی گرامرهای مستقل‌ازمتن لزوماً خطی نیستند.

مثال ۵-۳

زبان

$$L=\{a^nb^m : n \neq m\}$$

مستقل‌ازمتن است.

برای اثبات، باید یک گرامر مستقل‌ازمتن برای این زبان بسازیم. حالت $n=m$ را در مثال ۱-۱۱ حل کردیم و با اقتباس از همین راه‌حل، حالت $n>m$ را در نظر می‌گیریم. ابتدا رشته‌ای با تعداد a ها و b های مساوی تولید کرده و سپس، a های دیگری را به طرف چپ رشته های تولیدی اضافه می‌کنیم. گرامر حاصل بصورت

$$S \to AS_1,$$
$$S_1 \to aS_1b \mid \lambda,$$
$$A \to aA \mid a.$$

خواهد بود. برای حالت $n < m$ هم می‌توان از استدلالی مشابه استفاده کرد. در نهایت گرامر حاصل بصورت زیر خواهد بود:

$$S \to AS_1 \mid S_1B,$$
$$S_1 \to aS_1b \mid \lambda,$$
$$A \to aA \mid a,$$
$$B \to bB \mid b.$$

گرامر حاصل مستقل‌ازمتن بوده و بنابراین L هم یک زبان مستقل‌ازمتن است. با این وجود، این گرامر خطی نیست.

برای زبان ارائه شده در بالا، گرامرهای مختلفی را می‌توان ارائه داد. گرامر فوق یک نمونه از این گرامرها می‌باشد. در واقع، به ازای این زبان تعدادی گرامر خطی ساده وجود دارند. یکی دیگر از این گرامرها در تمرین ۲۶ پایان همین بخش از شما خواسته شده است. ■

مثال ۵-۴

گرامری با قوانین زیر را در نظر بگیرید.

$$S \to aSb \mid SS \mid \lambda.$$

این گرامر نمونه دیگری از گرامرهای مستقل‌ازمتن و غیرخطی است. برخی از رشته‌های عضو $L(G)$ عبارتند از: $ababab, aababb, abaabb$

به راحتی می‌توان عبارت:

$$L = \{w \in \{a,b\}^* : n_a(v) \geq n_b(v) \text{ و برای هر } v \text{ پیشوند } w\ n_a(w) = n_b(w)\} \quad (۱-۵)$$

را ارائه و اثبات کرد. با جایگذاری پرانتزهای باز و بسته به ترتیب به جای a و b به راحتی می‌توان به ارتباط این زبان با زبان‌های برنامه‌سازی پی برد. زبان L شامل رشته‌هایی مثل (()) و () () () می‌باشد و در واقع، به عنوان مجموعه تمامی ساختارهای پرانتزهای تودرتوی متداول در زبان‌های برنامه‌سازی رایج، استفاده می‌شوند.

برای این مثال گرامرهای متناظر متعددی وجود دارند. اما بر خلاف مثال ۵-۳، به راحتی نمی‌توان گرامرهای خطی برای آن طراحی کرد. برای یافتن پاسخ این سوال باید تا فصل هشت منتظر بمانیم. ■

اشتقاق‌های سمت راست‌ترین و سمت چپ‌ترین

در گرامرهای مستقل‌ازمتن غیرخطی، یک اشتقاق ممکن است دارای فرم‌های جمله‌ای با بیش از یک متغیر باشد. در این موارد، در ترتیب جایگزینی متغیرها آزاد هستیم. بعنوان مثال، گرامر $G = (\{A,B,S\},\{a,b\},S,P)$ با قوانین

۱. $S \to AB$
۲. $A \to aaA$
۳. $A \to \lambda$
۴. $B \to Bb$
۵. $B \to \lambda$

را در نظر بگیرید. این گرامر زبان $L(G) = \{a^{2n}b^m : n \geq 0, m \geq 0\}$ را تولید می‌کند. می‌توانید با گرفتن چند اشتقاق از درستی این نتیجه مطمئن شوید. اینک دو اشتقاق

$$S \overset{1}{\Rightarrow} AB \overset{2}{\Rightarrow} aaAB \overset{3}{\Rightarrow} aaB \overset{4}{\Rightarrow} aaBb \overset{5}{\Rightarrow} aab$$

و

$$S \overset{1}{\Rightarrow} AB \overset{4}{\Rightarrow} ABb \overset{2}{\Rightarrow} aaABb \overset{5}{\Rightarrow} aaAb \overset{3}{\Rightarrow} aab$$

را در نظر بگیرید. برای مشخص شدن قوانین استفاده شده در هر مورد، آنها را شماره‌گذاری کرده و شماره مربوط به هریک را روی علامت $\Rightarrow$ درج کرده‌ایم. مشاهده می‌کنید که هر دو اشتقاق نه تنها یک رشتهٔ یکسان را تولید می‌کنند، بلکه قوانین دقیقاً یکسانی هم دارند. تنها تفاوت، مربوط به ترتیب اعمال قوانین است. برای حذف این دست عوامل نامرتبط، بهتر است متغیرها با ترتیب خاصی جایگزین شوند.

تعریف ۵-۲

یک اشتقاق در صورتی **سمت چپ‌ترین** نامیده می‌شود که در هر مرحله، آخرین متغیر از سمت چپ فرم جمله‌ای جایگزین شود. اما اگر در هر مرحله آخرین متغیر از سمت راست جایگزین شود، این اشتقاق را اصطلاحاً **سمت راست‌ترین** می‌خوانیم.

مثال ۵-۵

گرامری با قوانین

$$S \to aAB,$$
$$A \to bBb,$$
$$B \to A \mid \lambda.$$

را در نظر بگیرید. آنگاه

$$S \Rightarrow aAB \Rightarrow abBbB \Rightarrow abAbB \Rightarrow abbBbbB \Rightarrow abbbbB \Rightarrow abbbb$$

یکی از اشتقاق‌های سمت چپ‌ترین رشته $abbbb$ می‌باشد و

$$S \Rightarrow aAB \Rightarrow aA \Rightarrow abBb \Rightarrow abAb \Rightarrow abbBbb \Rightarrow abbbb$$

یکی از اشتقاق‌های سمت راست‌ترین همین رشته است.

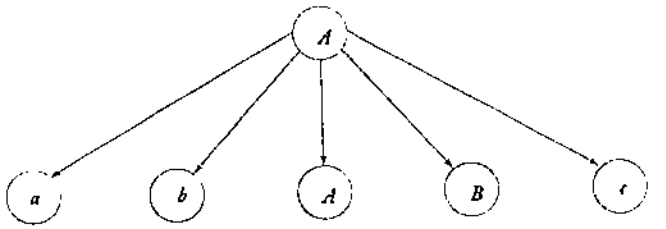
■

درخت‌های اشتقاق

روش دوم برای نمایش اشتقاق‌ها، استفاده از درخت تجزیه یا اشتقاق است (توجه داشته باشید که در این روش ترتیب بکارگیری قوانین اهمیتی ندارد). درخت اشتقاق در واقع درخت مرتبی است که در آن گره‌ها با سمت چپ قوانین نامگذاری می‌شوند و فرزندان یک گره به معرفی سمت راست آن گره می‌پردازند و به ترتیب از چپ به راست، سمبل‌ها و متغیرهای سمت راست می‌آیند. بعنوان مثال، در شکل ۵-۱ بخشی از درخت اشتقاق نمایش داده شده، که معادل قانون زیر است:

$$A \rightarrow abABc$$

در تمامی درخت‌های اشتقاق، با شروع از ریشه که با متغیر شروع گرامر نامگذاری می‌شود و خاتمه یافتن به برگ هایی که پایانی‌ها و یا λ هستند، درخت تکمیل می‌شود. این روش در قالب تعریف کلی زیر بیان می‌شود:



شکل ۵-۱

تعریف ۵-۳

فرض کنید $G = (V, T, S, P)$ یک گرامر مستقل‌ازمتن باشد. یک درخت مرتب، درخت اشتقاقی برای G خواهد بود اگر و تنها اگر خواص زیر را داشته باشد:

۱. ریشه دارای نام S است.

۲. هریک از برگها دارای نامی از $T \cup \{\lambda\}$ باشد.

۳. هریک از گره‌های میانی (گره‌ای که برگ نباشد) دارای نامی از V است.

۴. اگر یکی از گره‌های دارای نام $A \in V$ بوده و فرزندان آن هم (از چپ به راست) به صورت $a_1, a_2, \ldots, a_n$ نامگذاری شوند، آنگاه P باید قانونی به فرم

$$A \rightarrow a_1 a_2 \ldots a_n$$

داشته باشد.

۵. برگ‌های دارای نام λ هیچ خواهر و برادری ندارند؛ به این معنا که، گره‌ای که فرزند آن با λ نامگذاری شده باشد، فاقد فرزند دیگری است.

درختی که ویژگی‌های ۳، ۴ و ۵ را دارا باشد، اما خاصیت شماره ۲ لزوماً در آن برقرار نباشد. و خاصیت شماره ۲ در آن بصورت زیر جایگزین شده باشد.

۲ (الف). هر یک از برگها دارای نامی از $V \cup T \cup \{\lambda\}$ باشد، درخت **اشتقاق‌جزئی** نام دارد.

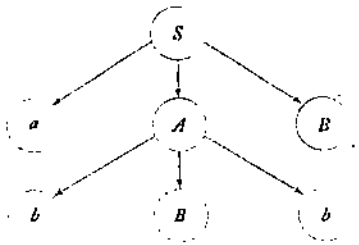
رشته‌ای از سمبل‌ها که با خواندن برگهای درخت از چپ به راست و حذف تمامی λ های مسیر ایجاد شود، اصطلاحاً تولید یا تولید درخت نامیده می‌شود. می‌توان به عبارت توصیفی چپ به راست تعریف دقیق‌تری نسبت داد. تولید، رشته‌ای از پایانی‌ها را نمایش می‌دهد که با پیمایش جستجوی اول عمق قابل ارائه می‌باشد. در این روش پیمایش، همواره سمت چپ‌ترین شاخه پیمایش نشده، دنبال می‌شود.

مثال ۵-۶

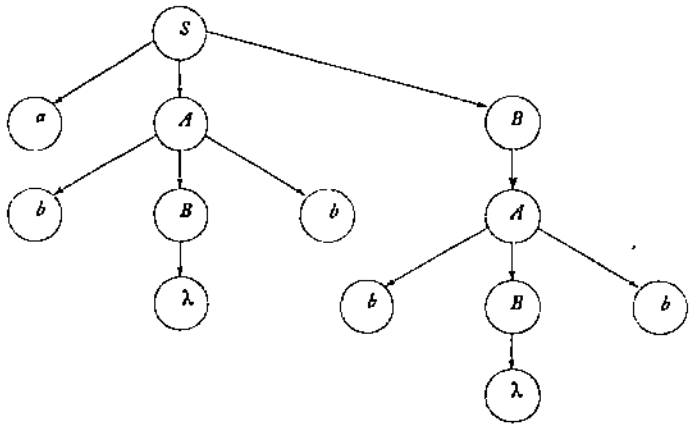
گرامر G با قوانین زیر را نظر بگیرید:

$$S \rightarrow aAB,$$
$$A \rightarrow bBb,$$
$$B \rightarrow A \mid \lambda.$$

درخت شکل ۵-۲ یک درخت اشتقاق جزئی برای G است در حالی که درخت شکل ۵-۳ یک درخت اشتقاق برای این گرامر می‌باشد. رشته $abBbB$ تولید درخت اول بوده و یک فرم جمله‌ای از G محسوب می‌شود. تولید درخت دوم، یعنی $abbbb$ ، رشته‌ای از $L(G)$ است.



شکل ۵-۲



شکل ۵-۳ ■

ارتباط بین فرم‌های جمله‌ای و درخت‌های اشتقاق

به کمک درخت‌های اشتقاق می‌توان اشتقاق مربوطه را به صورتی بسیار آشکار و قابل فهم تشریح کرد. همانند گراف‌های انتقال در اتوماتای متناهی، این صراحت کمک بزرگی در استدلال‌ها به ما می‌نماید. با این وجود، قبل از هر چیز باید ارتباطی بین اشتقاق‌ها و درخت‌های اشتقاق ایجاد کنیم.

قضیه ۵-۱

فرض کنید $G=(V,T,S,P)$ یک گرامر مستقل‌ازمتن باشد. آنگاه، به ازای هر $w\in L(G)$ یک درخت اشتقاق G برای تولید w وجود خواهد داشت. بالعکس، تولید تمامی درخت‌های اشتقاق عضو $L(G)$ می‌باشد. همچنین، اگر t_G یک درخت اشتقاق جزئی برای G با ریشه‌ای با برچسب S باشد، آنگاه تولید t_G یک فرم جمله‌ای از G خواهد بود.

اثبات: ابتدا نشان می‌دهیم که به ازای هر فرم جمله‌ای $L(G)$، یک درخت اشتقاق جزئی متناظر وجود دارد. این‌کار را با استقراء روی تعداد مراحل اشتقاق انجام می‌دهیم. بدین منظور به عنوان پایه استقراء، در نظر می‌گیریم که، نتیجه ادعا شده برای فرم جمله‌ای اشتقاق‌پذیر در یک مرحله، درست است. از آنجایی‌که اشتقاق $S\Rightarrow u$ نشان می‌دهد که قانون $S\to u$ وجود داشته است، این نتیجه به راحتی از روی تعریف ۵-۳ بدست می‌آید.

تصور کنید که به ازای هر فرم‌جمله‌ای اشتقاق‌پذیر طی n مرحله، یک درخت اشتقاق جزئی متناظر وجود داشته باشد. در اینصورت، هر w اشتقاق پذیر طی $n+1$ مرحله باید به گونه‌ای باشد که در n مرحله داشته باشیم:

$$S\Rightarrow xAy,\quad x,y\in(V\cup T)^*,\quad A\in V,$$

و

$$xAy\Rightarrow xa_1a_2..a_my=w,\quad a_i\in V\cup T,$$

از آنجائی‌که براساس فرض استقراء یک درخت اشتقاق جزئی با تولید xAy وجود دارد، و چون گرامر باید قانون تولید $A\to a_1a_2\cdots a_m$ را داشته باشد، با بسط گره‌ای دارای برچسب A، یک درخت اشتقاق جزئی با تولید $xa_1a_2\cdots a_my=w$ بدست خواهیم آورد. بنابراین، براساس استقراء ادعا می‌کنیم که نتیجه به ازای هر فرم‌جمله‌ای برقرار است.

به همین ترتیب، می‌توان نشان داد که هریک از درختهای اشتقاق جزئی، نماینده یکی از فرم‌های‌جمله‌ای هستند. این‌کار را به عنوان تمرین به خودتان واگذار می‌کنیم.

از آنجایی‌که یک درخت اشتقاق، درخت اشتقاق جزئی می‌باشد که برگها پایانی با λ هستند، می‌توان نتیجه گرفت که رشته عضو $L(G)$، تولید درخت اشتقاق مفروض G بوده و تولید هر یک از درخت‌های اشتقاق، عضو $L(G)$ است. ■

درخت‌های اشتقاق بیانگر قوانین مورد استفاده در تولید رشته‌ها هستند، اما هیچ ایده‌ای راجع به ترتیب استفاده قوانین ارائه نمی‌کنند. این درخت‌ها توانایی قابل ملاحظه‌ای در نمایش انواع اشتقاق‌ها دارند. همین ویژگی بیانگر بی‌تأثیربودن مسأله ترتیب استفاده قوانین و از این‌رو، بعنوان کلیدی برای پرکردن خلأ موجود در بحث قبل مورد استفاده قرار گرفته است. براساس تعریف، هر $w\in L(G)$ یک اشتقاق دارد، اما هرگز ادعا نکردیم که دارای اشتقاق چپ‌ترین یا راست‌ترین، نیز می‌باشد. هرچند با داشتن یک درخت اشتقاق، همواره می‌توان با این فرض که ابتدا آخرین متغیر از سمت چپ بسط یافته است، یک اشتقاق چپ‌ترین برای این درخت‌ها بدست آورد. با اضافه کردن برخی جزئیات، به این نتیجه مشخص می‌رسیم که هر $w\in L(G)$، دارای یک اشتقاق چپ‌ترین و یک اشتقاق راست‌ترین می‌باشد(برای درک بهتر مطلب به تمرین ۲۵ پایان همین بخش مراجعه کنید).

تمرین‌ها

۱. ضمن تکمیل استدلال مثال ۵-۲ نشان دهید که گرامر داده شده زبان مورد نظر را تولید می‌کند.

۲. درخت اشتقاق متناظر با اشتقاق مثال ۵-۱ را رسم کنید.

۳. یک درخت اشتقاق برای $w=abbbaabbaba$ را براساس گرامر مثال ۵-۲ رسم کنید. با استفاده از این درخت اشتقاق، یک اشتقاق چپ‌ترین ارائه دهید.

۴. نشان دهید که گرامر مثال ۵-۴ در حقیقت زبان توصیف شده در معادله ۵-۱ را تولید می‌کند.

۵. آیا زبان مثال ۵-۲ منظم است؟

۶. اثبات قضیه ۵-۱ را با استفاده از این مفهوم که، تولید هر درخت اشتقاق جزئی با ریشه S یکی از فرم‌های جمله‌ای G است، کامل کنید.

۷. گرامرهای مستقل‌ازمتنی را برای زبان‌های زیر بنویسید. (بطوریکه $n\geq 0, m\geq 0$).

الف) $L=\{a^nb^m : n \le m+3\}$. ⊖

ب) $L=\{a^nb^m : n \ne m-1\}$.

ج) $L=\{a^nb^m : n \ne 2m\}$.

د) $L=\{a^nb^m : 2n \le m \le 3n\}$. ⊖

ه) $L=\{w \in \{a,b\}^* : n_a(w) \ne n_b(w)\}$.

و) $L=\{v \in \{a,b\}^*$: $n_a(v) \ge n_b(v)$ که در آن v پیشوندی از w است$\}$.

ز) $L=\{w \in \{a,b\}^* : n_a(w)=2n_b(w)+1\}$.

۸. گرامرهای مستقل‌ازمتنی را برای زبان‌های زیر بنویسید. (بطوریکه $n \ge 0, m \ge 0, k \ge 0$).

الف) $L=\{a^nb^mc^k : n=m$ یا $m \le k\}$. ⊖

ب) $L=\{a^nb^mc^k : n=m$ یا $m \ne k\}$.

ج) $L=\{a^nb^mc^k : k=n+m\}$.

د) $L=\{a^nb^mc^k : n+2m=k\}$.

ه) $L=\{a^nb^mc^k : k=|n-m|\}$. ⊖

و) $L=\{w \in \{a,b,c\}^* : n_a(w)+n_b(w) \ne n_c(w)\}$.

ز) $L=\{a^nb^mc^k : k \ne n+m\}$.

ح) $L=\{a^nb^nc^k : k \ge 3\}$.

۹. نشان دهید که $L=\{w \in \{a,b,c\}^* : |w|=3n_a(w)\}$ یک زبان مستقل‌ازمتن است.

۱۰. گرامر مستقل از متنی را برای $head(L)$ بنویسید که در آن، L زبان تمرین ۷(الف) بالا می‌باشد. برای تعریف $head$، به تمرین ۱۸ بخش ۴-۱ مراجعه کنید.

۱۱. گرامر مستقل‌ازمتنی را به ازای $\Sigma=\{a,b\}$ برای زبان $L=\{a^nww^Rb^n : w \in \Sigma^*, n \ge 1\}$ بنویسید.

۱۲. *یا داشتن گرامر مستقل‌ازمتن G برای زبان مفروض L، نشان دهید که چگونه می‌توان گرامر $\hat{G}$ را از G بدست آورد بطوریکه $L(\hat{G})=head(L)$.

۱۳. فرض کنید $L=\{a^nb^n : n \ge 0\}$:

الف) نشان دهید که L^2 مستقل از متن است. ⊖

ب) نشان دهید که به ازای هر $k \ge 1$، L^k مستقل‌ازمتن است.

ج) نشان دهید که $\overline{L}$ و L^* مستقل‌ازمتن هستند.

۱۴. فرض کنید L_1 زبان تمرین ۸ (الف) و L_2 زبان تمرین ۸ (د) باشد. نشان دهید که $L_1 \cup L_2$ زبان مستقل‌ازمتن است.

۱۵. نشان دهید که زبان زیر مستقل‌ازمتن است.

$$L=\{uvwv^R : u,v,w \in \{a,b\}^+, |u|=|v|=2\}.$$

۱۶. *نشان دهید که مکمل زبان مثال۵-۱، مستقل‌ازمتن است. ⊖

۱۷. نشان دهید که مکمل زبان تمرین ۸ (ج)، مستقل‌ازمتن است.

۱۸. نشان دهید که زبان $L=\{w_1cw_2 : w_1,w_2 \in \{a,b\}^+, w_1 \ne w_2^R\}$ که در آن، $\Sigma=\{a,b,c\}$، مستقل‌ازمتن است.

۱۹. یک درخت اشتقاق برای رشته $aabbbb$ با گرامر زیر رسم کنید.

$$S \to AB \mid \lambda,$$
$$A \to aB,$$
$$B \to Sb.$$

به طور شهودی زبان تولید شده بوسیله این گرامر را شرح دهید.

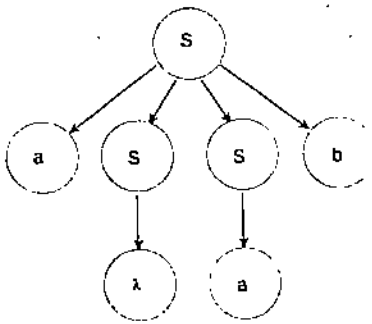
۲۰. گرامری با قوانین

$$S \to aaB,$$
$$A \to bBb \mid \lambda,$$
$$B \to Aa.$$

را در نظر بگیرید.

نشان دهید که رشته $aabbabba$ عضو زبان تولید شده بوسیله این گرامر نمی‌باشد. ⊖

۲۱. درخت اشتقاق زیر را در نظر بگیرید.



گرامر ساده G را پیدا کنید بطوریکه درخت بالا، درخت اشتقاق رشته aab باشد. سپس دو رشتهٔ دیگر از $L(G)$ را بنویسید.

۲۲. منظور از پرانتزگذاری تودرتوی درست، که شامل دو نوع پرانتز، مثل () و []، می‌باشند را تعریف کنید. به صورت شهودی، رشته‌های درست پرانتزبندی تودرتو بصورت [0]{0}([[]])،([]) هستند، نه [(]) یا ([]). با استفاده از این تعریف، گرامر مستقل‌ازمتنی را برای تولید تمام پرانتزبندی‌های تودرتوی درست، بنویسید.

۲۳. یک گرامر مستقل‌ازمتن برای مجموعه تمام عبارات منظم روی الفبای $\{a,b\}$ بنویسید. ⊖

۲۴. گرامر مستقل‌ازمتنی بنویسید که تمام قوانین تولید مربوط به گرامرهای مستقل‌ازمتنی را تولید کند که $T=\{a,b\}$ و $V=\{A,B,C\}$.

۲۵. اثبات کنید که اگر G یک گرامر مستقل‌ازمتن باشد، آنگاه هر $w \in L(G)$ دارای یک اشتقاق چپ‌ترین و راست‌ترین خواهد بود. الگوریتمی برای بدست آوردن این اشتقاق‌ها از یک درخت اشتقاق ارائه دهید.

۲۶. گرامری خطی برای زبان مثال ۵-۳ بنویسید.

۲۷. فرض کنید $G = (V, T, S, P)$ گرامر مستقل‌ازمتنی باشد که هر یک از قوانین آن به فرم $A \to v$ بوده و $|v| = k > 1$. نشان دهید که درخت اشتقاق به ازای تمامی $w \in L(G)$ ارتفاعی معادل h خواهد داشت، بطوریکه

$$\log_k |w| \le h \le \frac{(|w|-1)}{k-1}.$$

تجزیه و ابهام

تا به اینجا روی جنبه‌های تولیدی گرامرها متمرکز شدیم. بعنوان مثال، با داشتن گرامر G، مجموعه رشته‌هایی را مطالعه کردیم که با استفاده از G اشتقاق‌پذیر هستند. درکاربردهای عملی، با جنبه تحلیلی گرامرها مواجه می‌شویم؛ مثلاً با داشتن رشته w ای از پایانی‌ها، می‌خواهیم وجود یا عدم وجود w در زبان $L(G)$ را بررسی کنیم. در این موارد، گاهی اوقات لازم است یکی از اشتقاق‌های w را پیدا کنیم. برای بررسی وجود w در $L(G)$ از الگوریتمی به نام الگوریتم عضویت استفاده می‌شود. اصطلاح تجزیه به معنای یافتن دنباله‌ای از قوانین است که یک $w \in L(G)$ از آن مشتق شده است.

تجزیه و عضویت

با داشتن رشته w عضو زبان $L(G)$، می‌توان به ترتیب زیر اقدام به تجزیه آن کرد: به روشی دارای نظم می‌توان تمامی اشتقاق‌های (مثلاً چپ‌ترین) ممکن را بدست آورده و هر یک را از نظر برابری با w بررسی کنیم. به خصوص در مرحلهٔ اول با نگاه به همه قوانین، با در نظر گرفتن S به عنوان سمبل شروع گرامر، همه قوانین به فرم

$$S \to x,$$

را در نظر گرفته و تمامی x هایی را پیدا می‌کنیم که در یک مرحله از S اشتقاق‌پذیر هستند. چنانچه هیچ‌یک از این نتایج با w برابر نباشند، به مرحلهٔ بعد رفته و تمامی قوانین قابل انجام بر روی سمت چپ‌ترین متغیر هر یک از x ها، را بکار می بریم. به این ترتیب، مجموعه‌ای از فرم‌های جمله‌ای بدست می‌آید که ممکن است برخی از آنها به w ختم شوند. در هر یک از مرحله‌های بعدی، مجدداً تمامی متغیرهای سمت چپ‌ترین را در نظر گرفته و تمام قوانین ممکن را بکار می‌بریم. می‌توان برخی از این فرم‌های جمله‌ای را که w هیچگاه از آنها اشتقاق نخواهد شد، حذف کرد. اما بطورکلی، در هر مرحله، مجموعه‌ای از فرم‌های‌جمله‌ای ممکن بدست خواهد آمد. در پایان مرحله اول، فرم های جمله‌ای داریم که با اعمال فقط یک قانون بدست آمده‌اند؛ اما پس از دور دوم، فرم‌های جمله‌ای با

استفاده از دو مرحله اشتقاق را داریم و الی آخر. اگر $w \in L(G)$، آنگاه w باید یک اشتقاق چپ‌ترین با طول متناهی داشته باشد. در نتیجه، این روش نهایتاً یک اشتقاق چپ‌ترین را از w تولید خواهد کرد.

این روش، تجزیه جستجوی کامل نامیده می‌شود. که مدلی از تجزیه بالابه‌پایین است و می‌توان آنرا نتیجه یک درخت اشتقاق از ریشه به سمت پایین در نظرگرفت.

مثال ۵-۱۱

گرامر

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

و رشته $w = aabb$ را در نظر بگیرید. در مرحلهٔ اول داریم،

$$۱. S \Rightarrow SS,$$
$$۲. S \Rightarrow aSb,$$
$$۳. S \Rightarrow bSa,$$
$$۴. S \Rightarrow \lambda.$$

دو اشتقاق آخر که قادر به تولید رشته w نمی باشند را حذف می‌کنیم. به این ترتیب، در مرحلهٔ دوم با فرم‌های جمله‌ای

$$S \Rightarrow SS \Rightarrow SSS,$$
$$S \Rightarrow SS \Rightarrow aSbS,$$
$$S \Rightarrow SS \Rightarrow bSaS,$$
$$S \Rightarrow SS \Rightarrow S,$$

مواجه خواهیم شد که با جایگزینی چپ‌ترین S در فرم جمله‌ای ۱، با تمامی جایگزینی‌های قابل استفاده، بدست آمده است. به طور مشابه، از فرم جمله‌ای ۲ فرم‌های جمله‌ای دیگر

$$S \Rightarrow aSb \Rightarrow aSSb,$$
$$S \Rightarrow aSb \Rightarrow aaSbb,$$
$$S \Rightarrow aSb \Rightarrow abSab,$$
$$S \Rightarrow aSb \Rightarrow ab,$$

بدست می‌آید. مجدداً می‌توان چند فرم جمله‌ای دیگر را نادیده گرفت. در مرحلهٔ بعدی، از دنباله

$$S \Rightarrow aSb \Rightarrow aaSbb \Rightarrow aabb.$$

رشته خواسته شده را بدست می‌آوریم. بنابراین، $aabb$ بوسیلهٔ گرامر مورد نظر، عضو زبان تولید شده آن گرامر خواهد بود.

■

تجزیه جستجوی کامل، دچار نقاط ضعف مهمی می‌باشد که استفاده از آن خسته کننده است. به همین دلیل، برای تجزیه‌های کارا قابل استفاده نمی‌باشد. حتی در مواردی هم که کارآیی، اولویت دوم محسوب شود، بازهم چندان مفید نیست. هرچند به کمک این روش همواره یک $w \in L(G)$ تجزیه می‌شود، ولی ممکن است برای رشته‌های غیر موجود در $L(G)$، هرگز پایان‌پذیر نباشد. مثال قبل نمونه بارزی از این رخداد بود؛ بکار بردن این روش برای $w = abb$ دائماً و به مدت نامحدود به تولید فرم‌های جمله‌ای میانی می‌پردازد و اگر راهی برای متوقف‌کردن تولید فرم‌های جمله‌ای پیدا نکنیم، این کار به صورت نامتناهی ادامه پیدا می‌کند.

با محدود کردن فرم گرامری مجاز، به راحتی مشکل پایان‌ناپذیری در تجزیه جستجوی کامل بطورنسبی قابل حل خواهد بود. براساس مثال ۵-۷، مشکل فوق ناشی از قانون $S \to \lambda$ است؛ از این قانون می‌توان جهت کاهش طول فرم‌های جمله‌ای متوالی استفاده کرد که تشخیص محل توقف در آنها به سختی انجام می‌شود. با حذف این قوانین مشکلات کمتر می‌شود. در واقع، باید راجع به دو نوع قانون تصمیم‌گیری کنیم، یکی قوانینی که به فرم $A \to \lambda$ هستند و دیگری ، قوانین به فرم $A \to B$. همانطور که در فصل بعد خواهیم دید، در نظر گرفتن این محدودیت تأثیر چندان چشمگیری بر قدرت گرامرهای بدست آمده ندارد.

گرامر

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

با شرایط بالا همخوانی دارد. این گرامر زبان مثال ۵-۷ را، به غیر از رشته تهی، برای $w \in \{a,b\}^+$ تولید می‌کند.

روش تجزیه جستجوی کامل حداکثر در $2|w|$ مرحله متوقف خواهد شد. برای اثبات این ادعا باید به خاطر داشته باشیم که طول فرم جمله‌ای به اندازه حداقل یک سمبل در هر مرحله رشد می‌کند. پس از $2|w|$ مرحله، یا یک تجزیه داریم و یا به طور قطع می‌دانیم که $w \notin L(G)$.

■

ایده زیربنایی مثال بالا را می‌توان تعمیم داده و در قالب قضیه‌ای برای تمامی زبان‌های مستقل‌ازمتن مطرح نمود.

فرض کنید که $G = (V, T, S, P)$ گرامر مستقل‌ازمتنی باشد که هیچ قانونی به فرم

$$A \to \lambda,$$

یا

$$A \to B,$$

در آن وجود ندارد و $A, B \in V$. بنابراین، روش تجزیه جستجوی کامل می‌تواند بصورت الگوریتمی بکار رود که برای هر $w \in \Sigma^*$، یا تجزیه‌ای از w را تولید کند و یا به ما بگوید که هیچ تجزیه‌ای ممکن نیست.

اثبات: برای هر یک از فرم‌های جمله‌ای، طول و تعداد سمبل‌های پایانی را در نظر می‌گیریم. در هر یک از مراحل اشتقاق‌گیری، حداقل یکی از این‌دو فاکتور افزایش می‌یابد. از آنجایی‌که طول و تعداد سمبل‌های پایانی فرم‌های جمله‌ای هرگز بیش از $2|w|$ نمی‌شود، اشتقاق هم نمی‌تواند بیش از $2|w|$ مرحله داشته باشد. وقتی به این مرحله می‌رسیم، یا تجزیه موفقیت‌آمیز را پشت سر گذاشته‌ایم و یا دیگرکاملاً متقاعد شده‌ایم که گرامر مذکور قادر به تولید w نمی‌باشد. ■

هر چند روش جستجوی کامل تضمین نظری در مورد امکان‌پذیر بودن تجزیه محسوب می‌شود، اما چون تعداد فرم‌های جمله‌ای قابل تولید بوسیله آن، گاهی بسیار زیاد است، کاربرد عملی محدودی دارد. تعداد دقیق فرم‌های جمله‌ای تولید شده بر حسب مورد متفاوت بوده و هیچ قاعده کلی را نمی‌توان برای آن ارائه داد. اما می‌توان کرانه‌های بالایی را در این مورد در نظر گرفت. در اشتقاق‌های چپ‌ترین، پس از یک مرحله حداکثر $|P|$ فرم جمله‌ای، پس از مرحله دوم حداکثر $|P|^2$ فرم جمله‌ای خواهیم داشت و الی آخر. در اثبات قضیه ۵-۲، مشاهده کردیم که تجزیه، حداکثر طی $2|w|$ مرحله انجام می‌شود و بنابراین، تعداد کل فرم‌های جمله‌ای بیش از

$$M = |P| + |P|^2 + \cdots + |P|^{2|w|} \qquad (۲-۵)$$
$$= O(P^{2|w|+1}).$$

نخواهد بود. به این معنا که، تجزیه جستجوی کامل به صورت توانی بر حسب طول رشته تغییرکرده و هزینهٔ زمانی روش مورد نظر را بسیار زیاد می‌کند. البته، معادله (۵-۲) تنها یک حد بالا است و در اغلب موارد، تعداد فرم‌های جمله‌ای تولید شده بسیار کمتر از آن است. با این وجود، در عمل مشاهده کرده‌ایم که تجزیه جستجوی کامل در اغلب موارد بسیار ناکارا است.

ایجاد روش‌های تجزیه کاراتر برای گرامرهای مستقل‌ازمتن، موضوع پیچیده‌ای است و باید پیش از آن کامپایلرها را به دقت مورد بررسی قرار دهیم. به همین دلیل، بجز برای رسیدن به یک سری نتایج خاص، فعلاً از انجام این‌کار صرف‌نظر می‌کنیم.

قضیه ۵-۳

به ازای هر گرامر مستقل‌ازمتن، الگوریتمی وجود دارد که هر $w \in L(G)$ را در تعداد مراحلی متناسب با $|w|^3$ تجزیه می‌کند. ■

هرچند روش‌های مختلفی برای انجام این کار وجود دارد، اما همگی آنقدر پیچیده هستند که

توصیف آنها بدون ایجاد برخی نتایج دیگر امکانپذیر نیست. در بخش ۶-۳، مجدداً مسأله فوق را مطرح کرده و تا حدودی به آن می‌پردازیم. برای مطالعه بیشتر به کتابهای Harrison (۱۹۷۸) و (۱۹۷۹) Hopcroft and Ullman مراجعه کنید. یکی از دلایل عدم دنبال کردن این الگوریتم‌ها در این قسمت درس این است که الگوریتمهای مذکور به هیچ وجه قانع کننده نمی‌باشند. استفاده از روشهای دیگر، بطور نمونه معرفی رابطه فوق برحسب توان سوم طول رشته برای یک الگوریتم، هر چند بهتر از الگوریتمهای توانی است، ولی بازهم کارا نمی‌باشد؛ چون تجزیه‌ای که براساس آن انجام شود، حتی برای پردازش برنامه‌های تقریبا کوچک، زمان زیادی صرف می‌کند. بهتر است مدت زمان مورد استفاده روشها برای تجزیه، با طول رشته متناسب باشد. این روش به نام الگوریتم تجزیه **زمان خطی** می‌باشد. هرچند تا به امروز هیچ روش تجزیه‌ای که برای تمامی زبان‌های مستقل‌ازمتن در حالت کلی در زمان خطی کار کند، را نمی‌شناسیم، از این دست الگوریتم‌ها برای گرامرهای محدودی اما مهم و خاص استفاده می‌شود.

تعریف ۵-۴

گرامر مستقل‌ازمتن $G=(V,T,S,P)$ در صورتی **گرامر ساده** یا **s-گرامر** نامیده می‌شود که تمامی قوانین آن به فرم

$$A \to ax,$$

باشند که در آن، $a \in T, x \in V^*, A \in V$، و هر زوج (A,a) حداکثر یک بار در P وجود داشته باشد.

مثال ۵-۹

گرامر

$$S \to aS \mid bSS \mid c$$

یک s-گرامر است. در حالی که گرامر

$$S \to aS \mid bSS \mid aSS \mid c$$

s-گرامر نیست، چون زوج (S,a) در دو قانون $S \to aS$ و $S \to aSS$ وجود دارد. ■

با اینکه s-گرامرها بسیار محدود هستند اما حاوی نکات جالبی می‌باشند. همانطور که در بخش بعدی درس خواهیم دید، بسیاری از ویژگی‌های زبانهای برنامه‌سازی رایج بوسیله s-گرامرها قابل توصیف هستند.

اگر G یک s-گرامر باشد، آنگاه هر رشته w عضو $L(G)$ را می‌توان با مجموعه عملیاتهای متناسب با $|w|$ تجزیه کرد. برای اثبات، روش جستجوی کامل را در رشته $w=a_1a_2\cdots a_n$ بکار می‌گیریم. از آنجایی‌که به دلیل آغاز رشته در طرف راست با a_1 و قرار داشتن S در طرف چپ، حداکثر فقط یک

قانون برقرار می‌شود، اشتقاق باید با

$$S \Rightarrow a_1A_1A_2\cdots A_m$$

آغاز شود. سپس متغیر A_1 را جایگزین می‌کنیم، اما مجدداً به علت محدودیت تعداد انتخاب‌ها، داریم:

$$S \Rightarrow a_1a_2B_1B_2\cdots A_2\cdots A_m.$$

مشاهده می‌کنید که در هر مرحله یک سمبل پایانی تولید شده و به همین دلیل، کل فرآیند حداکثر طی $|w|$ مرحله تکمیل می‌شود.

ابهام در گرامرها و زبان‌ها

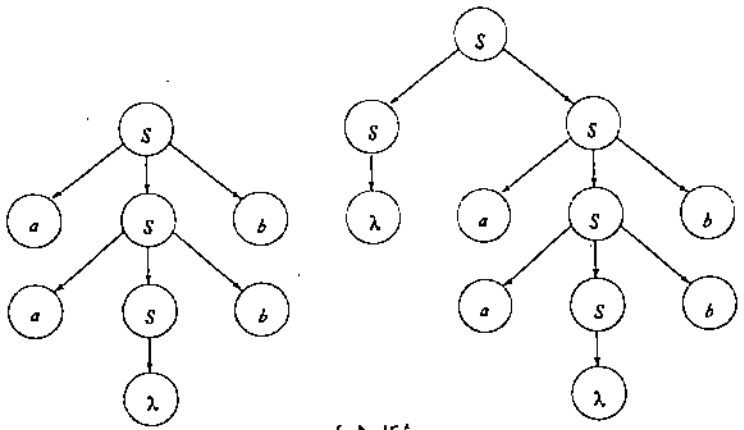
براساس استدلال فوق می‌توان ادعا کرد که با داشتن هر $w \in L(G)$، می‌توان با بکاربردن روش تجزیه جستجوی کامل، یک درخت اشتقاق برای w ایجاد نمود. در این مورد حتماً از عبارت "یک" استفاده می‌کنیم، چون ممکن است چندین درخت اشتقاق متفاوت هم وجود داشته باشد. این حالت اصطلاحاً ابهام خوانده می‌شود.

تعریف ۵-۵

گرامر مستقل‌ازمتن G در صورتی مبهم خوانده می‌شود که یک رشته $w \in L(G)$ وجود داشته باشد که حداقل دو درخت اشتقاق مجزا داشته باشد. به بیان دیگر، ابهام به طور ضمنی به معنای وجود دو یا چند اشتقاق چپ‌ترین یا راست‌ترین، نیز می‌باشد.

مثال ۵-۱۰

گرامر مثال ۵-۴ با قوانین $S \to aSb \mid SS \mid \lambda$ مبهم است. جمله $aabb$ دو درخت اشتقاق دارد که در شکل ۵-۴ مشاهده می‌کنید.



شکل ۵-۴

■

ابهام یکی از ویژگی‌های معمول زبان‌های طبیعی است و در زبان‌های مذکور به شیوه‌های مختلف با آن برخورد می‌شود. در زبان‌های برنامه‌سازی که قاعدتاً فقط یک تفسیر برای هر عبارات وجود دارد، در صورت امکان باید ابهام را حذف کرد. این کار غالباً از طریق بازنویسی گرامر به فرمی فاقد ابهام و معادل با گرامر اولیه انجام می‌شود.

مثال ۵-۱۱

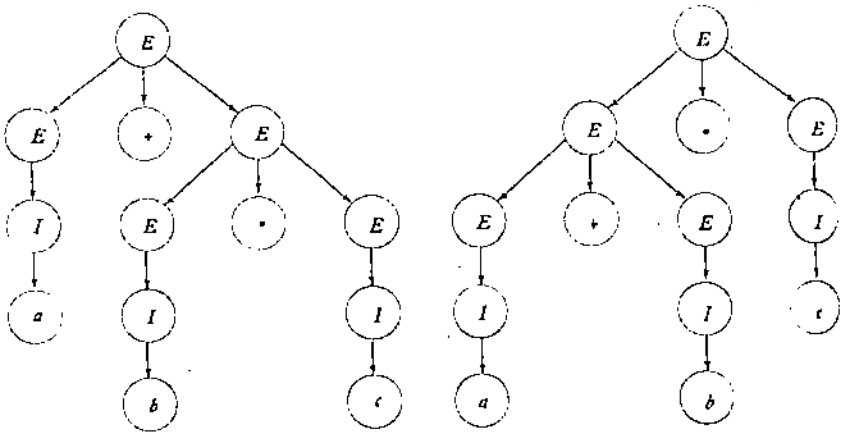
در گرامر $G=(V,T,E,P)$ که

$$V=\{E,I\},$$
$$T=\{a,b,c,+,*,(,)\},$$

و قوانین

$$E\to I,$$
$$E\to E+E,$$
$$E\to E*E,$$
$$E\to (E),$$
$$I\to a\mid b\mid c.$$

رشته‌های $c*(a+b)$ و $a*b+c$ عضو $L(G)$ هستند. به راحتی می‌توان مشاهده کرد که این گرامر زیرمجموعه محدودی از عبارات ریاضی را برای زبان‌های برنامه‌سازی شبیه C تولید می‌کند. بعلاوه، از آنجایی‌که مطابق شکل ۵-۵، بطورنمونه رشته $a+b*c$ دو درخت اشتقاق متفاوت دارد، بنابراین گرامر G مبهم است.



شکل ۵-۵ دو درخت اشتقاق برای $(a+b*c)$.

■

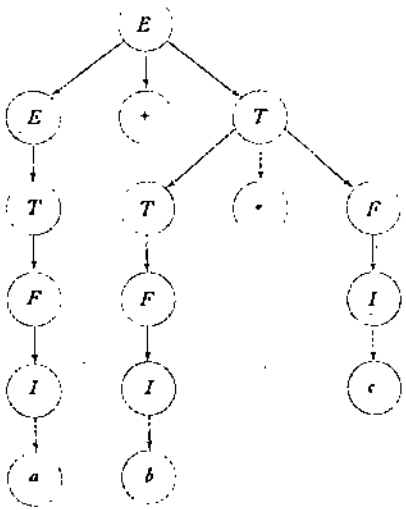
یک روش برای رفع ابهام، که در بسیاری از کتاب‌های آموزش برنامه‌سازی آورده شده است، تعریف قوانین تقدم عملگرها در هنگام استفاده از عملگرهای $+$، $*$ و غیره است. از آنجایی‌که عملگر $*$ اولویت بیشتری نسبت به عملگر $+$ دارد، می‌توان شکل ۵-۵ (الف) را به عنوان تجزیه درست پذیرفت. براساس این شکل، زیرعبارت $b*c$ را باید زودتر از انجام عمل جمع ارزیابی نمود. اما چون براساس گرامر چنین امکانی وجود ندارد، بهتر است گرامر را به صورت زیر بازنویسی کنیم تا فقط یک تجزیه امکان‌پذیر باشد.

مثال ۵-۱۲

برای بازنویسی گرامر مثال ۵-۱۱، متغیرهای جدیدی را معرفی می‌کنیم. به این منظور، V را به صورت $\{E,T,F,I\}$ در نظر گرفته و قوانین را با

$$E\to T,$$
$$T\to F,$$
$$F\to I,$$
$$E\to E+T,$$
$$T\to T*E,$$
$$F\to (E),$$
$$I\to a\mid b\mid c.$$

جایگزین می‌کنیم. درخت اشتقاق برای رشته $a+b*c$، در شکل ۵-۶ ارائه شده است.



شکل ۵-۶

هیچ درخت اشتقاق دیگری برای این رشته وجود ندارد و بنابراین، گرامر فوق غیرمبهم است همچنین هم‌ارز با گرامر مثال ۵-۱۱ می‌باشد. هرچند اثبات اینگونه ادعاها در این نمونه خاص تقریباً به راحتی انجام شد، ولی بطورکلی، پاسخگویی به پرسش‌هایی از قبیل مبهم‌بودن یا نبودن یک گرامر مستقل‌ازمتن و یا هم‌ارزبودن یا نبودن دو گرامر مستقل‌ازمتن، بسیار مشکل است. بعداً نشان خواهیم داد که هیچگونه الگوریتم کلی برای حل این قبیل سؤالات وجود ندارد.

■

در مثال اخیر، ابهام ناشی از گرامر بود. بطوریکه توانستیم با پیدا کردن یک گرامر فاقد ابهام معادل با آن، ابهام را به راحتی رفع کنیم. با این وجود، در برخی موارد، چون ابهام در دل زبان است، از بین بردن آن چندان آسان نیست.

تعریف ۵-۶

اگر L یک زبان مستقل‌ازمتن بوده و یک گرامر غیرمبهم به ازای آن وجود داشته باشد، آنگاه L غیرمبهم خوانده می‌شود. اگر هر گرامر تولیدکننده L مبهم باشد، آنگاه این زبان ذاتاً مبهم خوانده می‌شود.

اثبات اینکه زبانی ذاتاً مبهم است قدری مشکل است. در این موارد، بهترین کار ارائه یک مثال و ادعای ذاتاً مبهم بودن زبان براساس آن است.

مثال ■-۱۳

زبان

$$L=\{a^nb^nc^m\}\cup\{a^nb^mc^m\},$$

که در آن n و m غیرمنفی هستند، یک زبان مستقل‌ازمتن ذاتاً مبهم است.

مستقل‌ازمتن بودن L براحتی قابل اثبات است. توجه کنید که

$$L=L_1\cup L_2,$$

که در آن، L_1 بوسیله

$$S_1\to S_1c\mid A,$$
$$A\to aAb\mid\lambda.$$

تولید شده و L_2 با گرامری مشابه گرامر فوق با متغیر شروع S_2 و قوانین

$$S_2\to aS_2\mid B,$$
$$B\to bBc\mid\lambda.$$

تولید می‌شود. سپس، با ترکیب این‌دو گرامر با قانون زیر:

$$S\to S_1\mid S_2,$$

زبان L تولید می‌شود. به دلیل وجود دو اشتقاق مجزا برای رشته $a^nb^nc^n$، که یکی با $S\Rightarrow S_1$ و دیگری با $S\Rightarrow S_2$ شروع می‌شود، این گرامر مبهم است. البته، بر این اساس نمی‌توان به ذاتاً مبهم‌بودن L پی برد، چون ممکن است یک گرامر فاقد ابهام دیگر برای آن وجود داشته باشند. اما چون L_1 تعداد a ها و L_2 تعداد b ها را محدود می‌کند، این دو زبان وضعیت متفاوتی دارند. نمی‌توان این شروط را با هم ترکیب و در مجموعه‌ای از قوانین قرار داد که منحصراً $n=m$ را شامل شوند. در هر صورت، ارائه یک استدلال دقیق عملیاتی، بسیار تکنیکی است. یکی از این اثباتها را می‌توانید در کتاب (۱۹۷۸) Harrison مطالعه کنید.

■

تمرین‌ها

۱. یک s-گرامر برای $L(aaa^*b+b)$ بنویسید.

۲. یک s-گرامر برای $L=\{a^nb^n:n\geq 1\}$ بنویسید ●

۳. یک s-گرامر برای $L=\{a^nb^{n+1}:n\geq 2\}$ بنویسید

۴. نشان دهید که تمامی s-گرامرها فاقد ابهام هستند.

۵. فرض کنید $G=(V,T,S,P)$ یک s-گرامر باشد. یک عبارت برای حداکثر اندازه P برحسب $|V|$ و $|T|$ ارائه دهید.

۶. نشان دهید که گرامر زیر مبهم است. ●

$$S\to AB\mid aaB,$$
$$A\to a\mid Aa,$$
$$B\to b.$$

۷. یک گرامر غیر مبهم هم‌ارز با گرامر تمرین ۶ بسازید.

۸. با استفاده از گرامر مثال ۵-۱۲، یک درخت اشتقاق برای $(((a+b)*c))+a+b$ رسم کنید.

۹. نشان دهید که زبان‌های منظم نمی‌توانند ذاتاً مبهم باشند. ●

۱۰. یک گرامر غیر مبهم بنویسید که مجموعه تمام عبارات منظم روی $\Sigma=\{a,b\}$ را تولید کند.

۱۱. آیا گرامرهای منظم می‌توانند مبهم باشند؟

۱۲. نشان دهید که زبان $L=\{ww^R:w\in\{a,b\}^*\}$ ذاتاً مبهم نیست.

۱۳. نشان دهید که گرامر زیر مبهم است.

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

۱۴. نشان دهید که گرامر مثال ۵-۴ مبهم است، اما زبان تولید شده بوسیله آن مبهم نیست. ●

۱۵. نشان دهید که گرامر مثال ۱-۱۳ مبهم است.

۱۶. نشان دهید که گرامر مثال ۵-۵ غیرمبهم است.

۱۷. با استفاده از روش تجزیه جستجوی کامل، رشته $abbbbbb$ را با گرامر مثال ۵-۵، تجزیه کنید. بطور کلی، چه تعداد مرحله برای تجزیه هریک از رشته‌های w این زبان لازم است؟

۱۸. نشان دهید که گرامر مثال ۱-۱۴ غیرمبهم است.

۱۹. نتیجه زیر را اثبات کنید:

فرض کنید $G = (V, T, S, P)$ گرامر مستقل‌ازمتن باشد که بطوریکه هر $A \in V$ در سمت چپ حداکثر، یک قانون رخ می‌دهد. بنابراین، G غیرمبهم است.

۲۰. گرامری هم‌ارز با گرامر مثال ۵-۵ بنویسید، که شرایط قضیه ۵-۲ برای آن برقرار باشد.

۵ گرامرهای مستقل‌ازمتن و زبان‌های برنامه‌سازی

یکی از کاربردهای مهم نظریه زبان‌های صوری در تعریف زبان‌های برنامه‌سازی و همچنین ساخت مفسرها و کامپایلرها برای آنها است. مشکل اصلی در این موارد، ارائه تعریفی دقیق و روشن از زبان برنامه‌سازی و استفاده از این تعریف بعنوان نقطه آغاز نوشتن برنامه‌های مترجم کارآمد و قابل اطمینان است. زبان‌های منظم و مستقل‌ازمتن هر دو در کسب این موفقیت نقش کلیدی دارند. همانطور که قبلاً هم مشاهده کردیم، از زبانهای منظم جهت شناخت برخی الگوهای ساده و ویژه در زبانهای برنامه‌سازی استفاده می‌شود. اما براساس مقدمه همین فصل، برای مدل‌سازی جنبه‌های پیچیده‌تر، به زبان‌های مستقل‌ازمتن نیاز داریم.

زبان‌های برنامه‌سازی، مانند اغلب زبان‌های دیگر، بوسیله گرامر توصیف می‌شوند. استفاده از یک قرارداد برای تعریف گرامرها به منظور نوشتن زبان‌های برنامه نویسی مرسوم است، به این قرارداد فرم *Backus-Naur* یا BNF گفته می‌شود. این فرم بطور خلاصه، همان مجموعه نشانه‌هایی است که قبلاً استفاده می‌کردیم، اما با کمی تفاوت ظاهری. برای نوشتن سمبل‌های پایانی نیاز به نشانه‌گذاری خاصی نداریم. بعلاوه، BNF از یک نشانهٔ جایگزینی(یا) از قبیل | تقریباً مشابه قبل استفاده می‌کند. بنابراین، گرامر مثال ۵-۱۲ در BNF به صورت زیر نمایش داده می‌شود:

$$< expression > ::= < term > | < expression > + < term >,$$
$$< term > ::= < factor > | < term > * < factor >,$$

سمبل‌های + و * پایانی هستند. از نشانه | برای جایگزینی و مشابه نشانه‌گذاری استفاده می‌شود، اما ::= به جای $\rightarrow$ استفاده می‌شود. در توصیف‌های BNF از زبان‌های برنامه‌سازی بهتر است از شناسه‌های متغیری صریح‌تر برای روشن‌کردن مفهوم قانون تولید استفاده کنیم هیچ تفاوت مهمی بین این دو تعریف گرامر و BNF وجود ندارد.

بسیاری از بخشهای زبانهای برنامه‌سازی شبیه C بوسیله قوانینی خاص و محدود شده گرامرهای مستقل‌ازمتن قابل تعریف هستند. بعنوان مثال، عبارت *while* را می‌توان در C به صورت زیر تعریف کرد:

$$< while - statement > ::= while < expression > < statement > .$$

در اینجا، کلمه کلیدی *while* یک سمبل پایانی می‌باشد. عبارات دیگر متغیرهایی هستند که باید

تعریف شوند. با مقایسه عبارت فوق با تعریف ۵-۴ به شباهت این عبارت با قانون s-گرامرها پی می‌بریم. متغیر $< while - statement >$ در طرف چپ قانون، همواره با سمبل پایانی *while* در طرف راست مرتبط است. به همین دلیل، این‌گونه عبارات به راحتی و به صورتی کارآمد قابل تجزیه هستند. یکی از دلایل استفاده از کلمات کلیدی آن است که، کلمات کلیدی با ایجاد یک ساختار دیداری، خواننده برنامه را راهنمایی کرده و همچنین، باعث آسانی بسیار عملکرد کامپایلرها می‌شوند.

متأسفانه، تمامی ویژگی‌های زبان‌های برنامه‌سازی بوسیله s-گرامرها قابل ارائه نیستند. قوانین مربوط به $< expression >$ در بالا از این نوع نبوده و به همین دلیل، تجزیه کمتر مشخص می‌شود. این پرسش مطرح می‌شود که از کدام قوانین گرامری استفاده کنیم تا تجزیه کارآیی خود را حفظ کند. در کامپایلرها، بیشتر از گرامرهای LL و LR استفاده می‌شود. این گرامرها قادر به بیان ویژگی‌های نه چندان بارز زبانهای برنامه‌سازی بوده و در عین حال، امکان تجزیه در زمان خطی را فراهم می‌کنند. بررسی این موضوع نه چندان ساده تا حد زیادی خارج از مباحث این کتاب قرار دارد. در فصل ۶ دوباره این مسأله را مطرح خواهیم کرد، اما فعلاً فقط لازم است بدانیم که این دست گرامرها وجود داشته و بسیار هم مورد مطالعه قرار گرفته‌اند.

در این ارتباط، مسأله ابهام اهمیت بیشتری پیدا می‌کند. ساختارهای موجود در زبان‌های برنامه‌سازی باید غیرمبهم باشد؛ در غیراینصورت برنامه، در صورت پردازش بوسیله کامپایلرهای مختلف یا اجرا روی سیستم‌های مختلف، نتایج بسیار مختلفی را ایجاد می‌کند. بر اساس مثال ۵-۱۱، استفاده از روشهای نامعتبر ممکن است باعث ابهام در گرامر شود. برای جلوگیری از بروز این دست مشکلات، باید ابهامات را شناسایی کرده و آنها را رفع کنیم. گرامرهای ذاتاً مبهم برای این منظور مناسب هستند. به همین دلیل، نیاز به الگوریتم‌هایی جهت شناسایی و رفع ابهام در گرامرهای مستقل‌ازمتن و همچنین تصمیم‌گیری راجع به وجود یا عدم وجود ابهام ذاتی در زبان‌های مستقل‌ازمتن، را باید مهم دانست. اما همانطور که بعداً هم خواهیم دید، این قبیل تصمیم‌گیری‌ها بسیار مشکل بوده و در حالت کلی، حتی غیرممکن هستند.

ابعادی از زبان‌های برنامه‌سازی که بوسیله گرامرهای مستقل‌ازمتن، قابل مدل‌سازی هستند، بطور معمول نحو آن زبان خوانده می‌شوند. با این وجود، معمولاً همهٔ برنامه‌های نوشته شده به یک زبان برنامه‌سازی هرچند از نظر نحوی دارای خطا نمی‌باشند. درحقیقت از نظر معنایی قابل قبول نمی‌باشند. بطور نمونه در زبان برنامه‌نویسی C ، براساس تعریف متداول BNF ، جملاتی مانند

$$char \quad a,b,c;$$

که مقدار

$$c = 3.2;$$

است، بدون خطای نحوی هستند. در حالیکه دو جملهٔ فوق با هم دیگر، به دلیل نادیده گرفتن محدودیت "نمی‌توان به یک متغیر کاراکتری مقدار اعشاری نسبت داد"، برای کامپایلرهای C قابل قبول نیست. بوسیله گرامرهای مستقل‌ازمتن نمی‌توان مسأله غیرمجاز بودن عدم برابری نوع داده را

مطرح کرد. این دسته قوانین، از آنجایی‌که با چگونگی تفسیر معنای یک ساختار خاص ارتباط دارند، بخشی از معناشناسی زبان برنامه‌سازی محسوب می‌شوند.

معناشناسی زبان‌های برنامه‌سازی موضوعی پیچیده است. برای تشخیص معناشناسی زبان‌های برنامه‌سازی تنها ابزار، استفاده از گرامرهای مستقل‌ازمتن می‌باشد که چندان برای این کار دقیق نخواهد بود. و به همین دلیل، برخی ویژگی‌های معنایی ممکن است به خوبی تعریف نشده و یا مبهم باقی بمانند. این مشکل در هر دو حوزه زبان‌های برنامه‌سازی و نظریه زبان‌های صوری هنگامی ظاهر می‌شود که بخواهیم به کمک آنها روشهای مفید و مؤثری برای تعریف معنای زبان‌های برنامه‌سازی ارائه دهیم. هرچند تاکنون روشهای مختلفی برای این کار پیشنهاد شده، اما هیچ یک مورد اقبال عمومی قرار نگرفته است. بعلاوه، مقبولیت این روش‌ها به اندازه مقبولیت گرامرهای مستقل‌ازمتن برای تعریف نحو در تعریف معناشناسی موفق نبوده‌اند.

تمرین‌ها

۱. با مراجعه به یکی از منابع زبان برنامه‌نویسی C ، برای ساختارهای زیر یک گرامر مستقل‌ازمتن بنویسید.

الف) اعداد ثابت دوتایی

ب) عبارت for

ج) عبارت if-else

د) عبارت انجام بده

ه) عبارت ترکیبی

و) عبارت بازگشتی

۲. مثالهایی از ویژگی‌های زبان برنامه‌نویسی C ارائه دهید که بوسیله گرامرهای مستقل‌ازمتن قابل توصیف نباشد.

فصل ۶

ساده‌کردن گرامرهای مستقل‌ازمتن و فرم‌های نرمال

قبل از مطالعه عمیق زبان‌های مستقل‌ازمتن، باید به برخی از مسائل فنی اشاره کنیم. در تعریف گرامرهای مستقل‌ازمتن، هیچ محدودیتی برای سمت راست قانون در نظر گرفته نشده است. با این وجود، آزادی کامل هم پسندیده نیست و در برخی استدلال‌ها این آزادی، حتی ایجاد مشکل می‌کند. در قضیه ۵-۲، دیدیم که می‌توان برخی فرم‌های جمله‌ای را به صورتی محدود کرد؛ بعنوان مثال، حذف قوانین به فرم $A \to \lambda$ و $A \to B$ استدلال‌ها را ساده‌تر می‌کنند. در بسیاری از موارد، بهتر است حتی گام را، از این فراتر نهاده و محدودیت‌های شدیدتری در مورد گرامرها قائل شویم. به همین دلیل، به بررسی روش‌های تبدیل گرامرهای مستقل‌ازمتن به گرامرهای معادلی می‌پردازیم که در آنها از محدودیت‌های خاصی روی فرم قوانین استفاده می‌شود. در این فصل، ضمن طرح و تشریح برخی تبدیل‌ها و جایگزینی‌ها، از آنها در مباحث بعدی استفاده خواهیم کرد.

همچنین، **فرم‌های نرمال** گرامرهای مستقل‌ازمتن را بررسی خواهیم کرد. فرم نرمال، فرمی است که در عین محدودیت، آنقدر جامع است که هر گرامری یک نسخه فرم نرمال معادل دارد. در این فصل دو فرم مفید به نامهای **فرم نرمال چامسکی** و **فرم نرمال گریباخ** را معرفی می‌کنیم که کاربردهای عملی و نظری فراوانی پیدا کرده‌اند. یکی از کاربردهای فرم نرمال چامسکی برای استفاده در تجزیه رشته‌ها توسط گرامر می‌باشد، که در بخش ۶-۲ ارائه شده است.

ماهیت تقریباً یکنواخت مطالب این فصل ناشی از فرضی بودن بسیاری از استدلال‌هاست و به همین دلیل، شهود کمی را برای خواننده ایجاد می‌کنند. در بحث فعلی، این نقطه ضعف تکنیکی نسبتاً بی‌اهمیت و همچنین پذیرفته شده است. از نتایج مختلف و مهم مطرح شده به دفعات در مباحث بعدی استفاده شده است.

روش‌های تبدیل گرامرها

قبل از هر مطلبی، مسأله وجود رشته‌های λ را مطرح می‌کنیم که آفت جان تمامی گرامرها و زبان‌ها محسوب می‌شوند. رشته‌های λ در بسیاری از قضایا و اثبات‌ها نقش قابل ملاحظه‌ای را ایفا کرده و به همین دلیل باید بطور ویژه مورد توجه قرار گیرند. ترجیح می‌دهیم بحث در مورد رشته‌های λ را فعلاً نادیده گرفته و فقط زبان‌های فاقد λ را بررسی کنیم. برای انجام این کار، همان طور که در فرض‌های بعدی نیز مشهود است، عمومیت مسأله را نادیده نمی‌گیریم. بعنوان مثال، فرض کنید که L یک زبان مستقل‌ازمتن بوده و $G=(V,T,S,P)$ گرامر مستقل‌ازمتن برای $L-\{\lambda\}$ باشد. آنگاه، گرامری که با افزودن متغیر جدید S_0 به V، در نظر گرفتن S_0 بعنوان متغیر شروع و افزودن قوانین

$$S_0 \to S \mid \lambda$$

به P ایجاد می‌شود، زبان L را ایجاد خواهد کرد. بنابراین، هر نتیجه کلی که بتوانیم برای $L-\{\lambda\}$ داشته باشیم، همواره قابل تبدیل به L خواهد بود. همچنین، با داشتن هر گرامر مستقل‌ازمتن G، روشی برای بدست‌آوردن $\hat{G}$ وجود دارد، بطوریکه $L(\hat{G})=L(G)-\{\lambda\}$ (بعنوان نمونه، به تمرین‌های ۱۳ و ۱۴ پایان همین بخش مراجعه کنید). در نتیجه، در تمامی مسائل واقعی، هیچ تفاوتی بین زبان‌های مستقل‌ازمتن دارای λ و فاقد λ، وجود ندارد. در ادامه فصل، جز در برخی موارد، فقط در مورد زبان‌های مستقل‌ازمتن فاقد λ، بحث خواهیم کرد.

یک قانون جایگزینی مناسب

بسیاری از قوانین می‌توانند با استفاده از جایگزینی، گرامرهای معادلی را برای تولید زبان بوجود آورند. در ادامه، یکی از همین قوانین بسیار مفید را ارائه می‌دهیم که در ساده‌سازی گرامرها به صورت‌های مختلف مورد استفاده قرار می‌گیرند. هرچند اصطلاح ساده‌سازی را به طور دقیق تعریف نمی‌کنیم، اما از آن استفاده خواهیم کرد. بطور خلاصه، منظور ما از ساده‌سازی، حذف برخی انواع قوانین نامطلوب است؛ ولی این فرآیند لزوماً تعداد قوانین را کاهش نمی‌دهد.

قضیه ۶-۱

فرض کنید $G=(V,T,S,P)$ یک گرامر مستقل‌ازمتن باشد، که در P قانونی به فرم

$$A \to x_1 B x_2$$

باشد. همچنین، فرض کنید A و B متغیرهای متفاوتی بوده و همچنین،

$$B \to y_1 \mid y_2 \mid \cdots \mid y_n$$

مجموعه تمام قوانین عضو P باشد که B در طرف چپ آنها قرار گرفته است. فرض کنید $\hat{G}=(V,T,S,\hat{P})$ گرامری باشد که در آن، $\hat{P}$ با حذف

$$A \to x_1 B x_2 \qquad (۱-۶)$$

از P و اضافه کردن

$$A \to x_1 y_1 x_2 \mid x_1 y_2 x_2 \mid \cdots \mid x_1 y_n x_2$$

به گرامر مذکور ایجاد می‌شود. آنگاه

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

اثبات: فرض کنید که $w \in L(G)$ باشد، بطوریکه

$$S \Rightarrow_G w.$$

از زیرنویس علامت اشتقاق $\Rightarrow$ جهت تمایز بین اشتقاق‌ها و انواع گرامرها استفاده شده است. اگر قانون (۶-۱) در این اشتقاق وجود نداشته باشد، آنگاه مشخص است که

$$S \Rightarrow_{\hat{G}} w.$$

در غیر اینصورت، اولین اشتقاق را بررسی می‌کنیم که در اثر استفاده از (۶-۱) ایجاد شده است. B که به این ترتیب بدست آمده، نهایتاً باید جایگزین شود؛ فرض کنید که این کار بلافاصله انجام شود (مراجعه شود به تمرین ۱۸ انتهای همین بخش). بنابراین،

$$S \Rightarrow_G u_1 A u_2 \Rightarrow_G u_1 x_1 B x_2 u_2 \Rightarrow_G u_1 x_1 y_j x_2 u_2.$$

اما با استفاده از گرامر $\hat{G}$ می‌نویسیم:

$$S \Rightarrow_{\hat{G}} u_1 A u_2 \Rightarrow_{\hat{G}} u_1 x_1 y_j x_2 u_2.$$

بنابراین G و $\hat{G}$ هر دو فرم‌های جمله‌ای یکسانی را ایجاد می‌کنند. اگر بعداً دوباره از (۶-۱) استفاده شود، همین استدلال را دوباره برای آن تکرار می‌کنیم. بنابراین، با استقراء روی تعداد دفعات بکار بردن قوانین، داریم:

$$S \Rightarrow_{\hat{G}} w.$$

در نتیجه، اگر $w \in L(G)$ باشد، آنگاه $w \in L(\hat{G})$.

با استدلالی مشابه، می‌توان نشان داد که اگر $w \in L(\hat{G})$ باشد، آنگاه $w \in L(G)$ و اثبات قضیه به پایان می‌رسد. ■

قضیه ۶-۱ یک قانون جایگزینی ساده و کاملاً شهودی را مطرح می‌کند: قانون $A \to x_1 B x_2$ در صورتی از گرامر قابل حذف است که به جای آن مجموعه قوانینی قرار دهیم که در آن، به جای B از تمامی فرم‌های جمله‌ای استفاده می‌شود که در یک مرحله اشتقاق شده‌اند. برای این منظور، A و B باید متغیرهای متفاوتی باشند. در مورد حالت خاص $A=B$ در تمرین‌های ۲۳ و ۲۴ انتهای همین بخش بحث کرده‌ایم.

مثال ۶-۱

گرامر $G=(\{A,B\},\{a,b,c\},A,P)$ را با قوانین

$$A \to a \mid aaA \mid abBc,$$
$$B \to abbA \mid b.$$

در نظر بگیرید. با استفاده از جایگزینی پیشنهادی برای متغیر B، گرامر $\hat{G}$ با قوانین:

$$A \to a \mid aaA \mid ababbAc \mid abbc,$$
$$B \to abbA \mid b.$$

بدست می‌آید. گرامر جدید $\hat{G}$ متناظر با G است. اشتقاق رشته $aaabbc$ در G

$$A \Rightarrow aaA \Rightarrow aaabBc \Rightarrow aaabbc$$

و اشتقاق نظیر آن در $\hat{G}$

$$A \Rightarrow aaA \Rightarrow aaabbc$$

می‌باشد.

توجه داشته باشید که در این حالت، متغیر B و قوانین مربوط به آن، هر چند نقشی در هیچکدام از اشتقاق‌ها ندارند، ولی هنوز در گرامر وجود دارند. در ادامه نحوه حذف این قوانین غیرضروری از گرامر را توضیح خواهیم داد.

■

حذف قوانین بی‌فایده

برخی گرامرها دارای قوانینی هستند که نقشی در اشتقاق‌ها نداشته و به همین دلیل قابل حذف هستند. بعنوان مثال، گرامری که مجموعه همهٔ قوانین آن بصورت:

$$S \to aSb \mid \lambda \mid A,$$
$$A \to aA.$$

می‌باشند، را در نظر بگیرید. بدلیل آنکه A نمی‌تواند رشته‌ای از پایانی‌ها را تولید کند، قانون $S \to A$ نیز هیچ نقشی ایفا نمی‌کند. هرچند ممکن است A در فرم‌های جمله‌ای مشتق‌شده از S حضور داشته باشد، اما هرگز به رشته‌ای از پایانی‌ها ختم نخواهد شد. با حذف این قانون هیچ تغییری در زبان ایجاد نشده و تنها، نوعی ساده‌سازی محسوب می‌شود.

تعریف ۶-۱

فرض کنید $G=(V,T,S,P)$ یک گرامر مستقل‌ازمتن باشد. متغیر $A \in V$ مفید خوانده می‌شود اگر و تنها اگر حداقل یک $w \in L(G)$ وجود داشته باشد که

$$S \Rightarrow xAy \Rightarrow w, \qquad (۶-۲)$$

بطوریکه x,y عضو $(V \cup T)^*$ می‌باشند. به عبارت دیگر، یک متغیر مفید است اگر و تنها اگر در حداقل یک اشتقاق حضور داشته باشد. متغیری هم که مفید نباشد، بی‌فایده یا غیرمفید خوانده می‌شود. یک قانون در صورتی بی‌فایده است که یکی از متغیرهای بی‌فایده در آن حضور داشته باشد.

مثال ۶-۲

یکی از دلایل بی‌فایده بودن یک متغیر آن است که به هیچ وجه نتوان یک رشته پایانی از آن بدست آورد. این مورد یکی از عوامل بی‌فایده‌بودن متغیرها است. عامل دیگر برای بی‌فایده بودن متغیرها، در گرامر زیر ارائه شده است. در گرامری با سمبل شروع S و قوانین

$$S \to A,$$
$$A \to aA \mid \lambda,$$
$$B \to bA.$$

متغیر B و همچنین قانون $B \to bA$ بی‌فایده هستند. هرچند متغیر B قادر به اشتقاق یک رشته پایانی می‌باشد، ولی $S \Rightarrow xBy$ هیچگاه استفاده نمی شود یعنی B از طریق S قابل دسترس نمی‌باشد.

■

مثال بالا، دو عامل بی‌فایده بودن متغیرها، یعنی قابل‌دستیابی‌نبودن متغیر از طریق متغیر شروع گرامر و ناتوانی در اشتقاق یک رشته پایانی، را نشان داد. به محض شناسایی یکی از این‌دو عامل، امکان حذف متغیرها و قوانین بی‌فایده بوجود می‌آید. پیش از ارائه حالت کلی این روال و قضیه مربوطه، مثال دیگری را بررسی می‌کنیم.

مثال ۶-۳

متغیرها و قوانین بی‌فایده را از $G=(V,T,S,P)$ حذف کنید. در این گرامر، $V=\{S,A,B,C\}$ و $T=\{a,b\}$ و P شامل

$$S \to aS \mid A \mid C,$$
$$A \to a,$$
$$B \to aa,$$
$$C \to aCb.$$

می‌باشد.

ابتدا، مجموعه متغیرهایی را شناسایی می‌کنیم که می‌توانند یک رشته پایانی تولید کنند. از آنجایی‌که $A \to a$ و $B \to aa$، متغیرهای A و B به این مجموعه تعلق دارند. همین ادعا در مورد S صدق می‌کند، چون $S \Rightarrow A \Rightarrow a$. در حالیکه این استدلال در مورد متغیر C صحیح نبوده و بنابراین متغیر C بی‌فایده است. با حذف C و قوانین مربوط به آن، گرامر G_1 با متغیرهای $V_1=\{S,A,B\}$،

پایانی‌های $T = \{a\}$ و قوانین

$$S \to aS \mid A,$$
$$A \to a,$$
$$B \to aa.$$

بدست می‌آید. سپس، اقدام به حذف متغیرهایی می‌کنیم که این متغیرها از متغیر شروع گرامر قابل دستیابی نمی باشند. به این منظور، می‌توان **گراف وابستگی** این متغیرها را رسم کرد. گراف‌های وابستگی روشی برای تجسم روابط پیچیده بوده و کاربردهای فراوانی دارند. در گراف‌های وابستگی مربوط به گرامرهای مستقل‌ازمتن، رئوس با متغیرها برچسب‌دار می‌شوند و تنها در صورتی یک یال بین رئوس C و D قرار می‌گیرد که قانونی به فرم

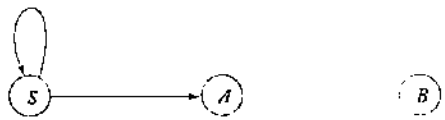
$$C \to xDy$$

وجود داشته باشد.

گراف وابستگی به ازای V_1 را در شکل ۶-۱ مشاهده می‌کنید. یک متغیر فقط هنگامی مفید است که مسیری از رأس دارای برچسب S به رأس برچسب‌دار شده با همان متغیر وجود داشته باشد. بنابراین دز مثال حاضر و براساس شکل ۶-۱، B بی فایده است. با حذف B و قانون‌ها و پایانی‌های متأثر از آن، گرامر نهایی $\hat{G} = (\hat{V}, \hat{T}, S, \hat{P})$ با $\hat{T} = \{a\}$، $\hat{V} = \{S, A\}$ و قوانین

$$S \to aS \mid A,$$
$$A \to a.$$

بدست می‌آید. با فرموله کردن این قضیه می‌توان روش کلی ساخت الگوریتم و قضیه مربوط به آن را ارائه داد.



شکل ۶-۱

■

قضیه ۶-۲

$G = (V, T, S, P)$ را یک گرامر مستقل‌ازمتن فرض کنید. آنگاه گرامر هم‌ارز $\hat{G} = (\hat{V}, \hat{T}, S, \hat{P})$ وجود دارد که شامل هیچ متغیر یا قانون بی‌فایده نمی‌باشد.

اثبات: گرامر $\hat{G}$ را می‌توان با استفاده از یک الگوریتم دو فازی از G تولید کرد. در فاز اول، گرامر میانی $G_1 = (V_1, T_1, S, P_1)$ را به گونه‌ای می‌سازیم که V_1 فقط حاوی متغیرهای A باشد که برای آن

$$A \overset{*}{\Rightarrow} w \in T^*$$

امکان‌پذیر باشد. مرحله‌های این الگوریتم به ترتیب زیر هستند:

۱. V_1 را معادل $\varnothing$ قرار دهید.

۲. این مرحله را آنقدر تکرار کنید تا هیچ متغیر دیگری به V_1 اضافه نشود. به ازای هر $A \in V$ که در آن، P قانونی به فرم:

$$A \to x_1 x_2 \cdots x_n$$ برای تمام x_iهای متعلق به $V_1 \cup T$

باشد، A را به V_1 اضافه کنید.

۳. P_1 را تمامی قوانین موجود در P، در نظر بگیرید که تمام سمبل‌های آن در $(V_1 \cup T)$ قرار داشته باشند.

ناگفته پیداست که این روال خاتمه‌پذیر است. همچنین واضح است که اگر $A \in V_1$، آنگاه $A \overset{*}{\Rightarrow} w \in T^*$ یکی از اشتقاق‌های ممکن از G_1 است. بعلاوه، این پرسش مطرح می‌شود که آیا تمامی Aهای با ضابطه $A \overset{*}{\Rightarrow} w = ab \cdots$، پیش از توقف روال، به V_1 اضافه می‌شود یا نه. برای پاسخ به این سؤال، هر A تعریفی متناظر با این اشتقاق را به همراه درخت اشتقاق جزئی متناظر با آن اشتقاق در نظر می‌گیریم (شکل ۶-۲). در سطح k، فقط پایانی‌ها وجود دارند و بنابراین تمامی متغیرهای A_i واقع در سطح k-1، در اولین گذر از مرحله ۲ الگوریتم، به سطح k-1 افزوده خواهند شد. سپس، هر متغیر در سطح k-2 در دومین عبور از مرحله ۲ به V_1 افزوده خواهد شد. در سومین عبور از مرحله ۲، تمامی متغیرهای واقع در سطح k-3 اضافه خواهند شد و الی آخر. و تا زمانی که متغیرهایی وجود داشته باشند که در درخت V_1 وجود ندارند، الگوریتم متوقف نخواهد شد. به همین دلیل سرانجام، A به V_1 اضافه می‌شود.

در فاز دوم ساخت، جواب نهایی $\hat{G}$ را از G_1 بدست می‌آوریم. برای این منظور، گراف وابستگی متغیر را به ازای G_1 رسم کرده و با بررسی آن، تمامی متغیرهایی را پیدا می‌کنیم که از S قابل دسترسی نیستند. این متغیرها به همراه قوانین مربوط به آنها از مجموعه متغیرها حذف می‌شوند. همچنین می‌توان پایانی‌هایی را حذف کرد که در قوانین مفید وجود ندارند. گرامر حاصل $\hat{G} = (\hat{V}, \hat{T}, S, \hat{P})$ می‌باشد.

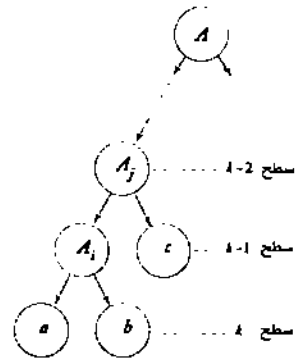
به دلیل نوع ساخت، $\hat{G}$ حاوی هیچ سمبل و یا قانون بی‌فایده‌ای نخواهد بود. همچنین، به ازای هر اشتقاق $w \in L(G)$

$$S \overset{*}{\Rightarrow} xAy \overset{*}{\Rightarrow} w$$

را داریم. بدلیل آنکه A و تمام قوانین مربوط به آن در جریان ساخت $\hat{G}$ حفظ می‌شود، بنابراین هر آنچه را که برای گرفتن اشتقاق

$$S \overset{*}{\Rightarrow} \hat{G}\ xAy \overset{*}{\Rightarrow} \hat{G}\ w$$

نیاز داریم، اکنون نیز وجود دارد. گرامر $\hat{G}$ با حذف قانون‌ها از G بدست می‌آید، بطوریکه $\hat{P} \subseteq P$. در نتیجه $L(\hat{G}) \subseteq L(G)$. سپس با کنار هم قرار دادن و مقایسه این‌دو نتیجه، به هم‌ارز بودن G و $\hat{G}$ پی می‌بریم.



شکل ۶-۲

■

حذف قوانین λ

یکی از انواع قوانین گاهاً نامطلوب، قوانینی هستند که در طرف راست آنها رشته تهی قرار دارد.

تعریف ۶-۲

هر قانونی از یک گرامر مستقل‌ازمتن به فرم

$$A \to \lambda$$

قانون λ خوانده می‌شود. هر متغیر A که اشتقاق

$$A \overset{*}{\Rightarrow} \lambda \qquad (۶-۳)$$

برای آن امکان‌پذیر باشد، میرا (توانا برای تولید λ) نامیده می‌شود.

برخی گرامرها زبان‌هایی را تولید می‌کنند که هرچند فاقد λ هستند، تعدادی متغیر میرا یا قانون λ در آنها وجود دارند. در این موارد، می‌توان قوانین λ را حذف کرد.

مثال ۶-۴

گرامر

$$S \to aS_1b,$$

$$S_1 \to aS_1b \mid \lambda.$$

با متغیر شروع S را در نظر بگیرید. این گرامر زبان $\{a^nb^n : n \geq 1\}$ که فاقد λ می‌باشد را تولید می‌کند. می‌توان پس از افزودن قوانین جدیدی که با جایگزینی λ در S_1های واقع در طرف راست بدست آمده‌اند، قانون $S_1 \to \lambda$ را حذف نمود. به این ترتیب، گرامر

$$S \to aS_1b \mid ab,$$
$$S_1 \to aS_1b \mid ab.$$

بدست می‌آید. به راحتی می‌توان نشان داد که این گرامر جدید زبان اصلی را تولید می‌کند.
در موارد کلی‌تر، جایگزینی قوانین λ به روشی مشابه، اما با کمی دقت و پیچیدگی، انجام می‌شود.

■

قضیه ۶-۳

G را یک گرامر مستقل‌ازمتن با شرط عدم وجود λ در $L(G)$ در نظر بگیرید. آنگاه، گرامر هم‌ارز $\hat{G}$ وجود دارد که فاقد قانون λ است.

اثبات: ابتدا براساس مراحل زیر، مجموعه V_N شامل تمام متغیرهای میرا در G را پیدا می‌کنیم.

۱. به ازای تمام قوانین $A \to \lambda$، A را در V_N قرار دهید.
۲. مرحله زیر را آنقدر تکرار کنید که هیچ متغیر دیگری به V_N اضافه نشود.
به ازای تمام قوانین

$$B \to A_1A_2 \cdots A_n,$$

که در آن $A_1, A_2, \ldots, A_n$ در V_N قرار دارند، B را در V_N قرار دهید.

پس از یافتن مجموعه V_N، دیگر ساخت $\hat{P}$ چندان مشکل نیست. برای این کار، تمام قوانین موجود در P به فرم

$$A \to x_1x_2 \cdots x_m \ , \quad m \geq 1,$$

را در نظر می‌گیریم که در آن، هر $x_i \in V \cup T$. به ازای هر یک از این قوانین P، قانون مذکور و همچنین تمام قوانینی که با جایگزینی متغیرهای میرا به جای λ در تمام ترکیبات ممکن بوجود آمده‌اند، را در $\hat{P}$ قرار می‌دهیم. بعنوان مثال، اگر x_i و x_j هر دو توانا برای تولید λ باشند، یک قانون در $\hat{P}$ با ضابطه جایگزینی x_i به λ، یکی با ضابطه جایگزینی x_j به λ و دیگری با ضابطه جایگزینی x_i و x_j به λ موجود است. فقط یک استثناء وجود دارد:
اگر تمامی x_iها توانا برای تولید λ باشند، قانون $A \to \lambda$ را در $\hat{P}$ قرار نمی‌دهیم.
بررسی معادل بودن گرامر $\hat{G}$ با G بسیار آسان و قابل فهم بوده و به همین دلیل، آنرا به خواننده واگذار می‌کنیم. ■

مثال ۶-۵

یک گرامر مستقل‌ازمتن فاقد قانون λ و متناظر با گرامر زیر پیدا کنید:

$$S \to ABaC,$$
$$A \to BC,$$
$$B \to b \mid \lambda,$$
$$C \to D \mid \lambda,$$
$$D \to d.$$

از همان ابتدای ساخت گرامر مورد نظر براساس قضیه ۶-۳، متوجه می‌شویم که A, B, C متغیرهای میرا هستند. سپس، براساس مرحله دوم ساخت، داریم:

$$S \to ABaC \mid BaC \mid AaC \mid ABa \mid aC \mid Aa \mid Ba \mid a,$$
$$A \to B \mid C \mid BC,$$
$$B \to b,$$
$$C \to D,$$
$$D \to d.$$

حذف قوانین واحد

همانطور که در قضیه ۵-۲ دیدیم، قوانینی که در هر دو طرف آنها فقط یک متغیر وجود داشته باشد، گاهی اوقات نامطلوب هستند.

تعریف ۶-۲

هر قانونی از یک گرامر مستقل‌ازمتن به فرم

$$A \to B,$$

که در آن $A, B \in V$، **قانون یکه یا واحد** نامیده می‌شود.

برای حذف قوانین واحد، از قانون جایگزینی مطرح شده در قضیه ۶-۱ استفاده می‌کنیم. اینکار مطابق قضیه بعد به راحتی قابل انجام است.

قضیه ۶-۴

$G = (V, T, S, P)$ را یک گرامر مستقل‌ازمتن فاقد قانون λ در نظر بگیرید. آنگاه گرامر مستقل‌ازمتن $\hat{G} = (\hat{V}, \hat{T}, S, \hat{P})$ وجود دارد که هیچ قانون واحدی نداشته و هم‌ارز با G می‌باشد.

اثبات: مشخص است که هر قانون واحد به فرم $A \to A$ را می‌توان به راحتی از گرامر حذف کرد؛ فقط باید دقت کنیم که در $A \to B$، A و B متغیرهای متفاوتی باشند. در نگاه اول، ممکن است به نظر برسد که می‌توان مستقیماً از قضیه ۶-۱ با ضابطه $x_1 = x_2 = \lambda$ جهت جایگزینی

$$A \to B$$

در

$$A \to y_1 \mid y_2 \mid \cdots \mid y_n.$$

استفاده کرد. اما این روش همیشه امکان‌پذیر نیست؛ مثلاً در حالت خاص

$$A \to B,$$
$$B \to A,$$

قوانین واحد حذف نمی‌شوند. برای بررسی این مسأله، ابتدا به ازای هر A، تمام متغیرهای B را پیدا می‌کنیم که

$$A \overset{*}{\Rightarrow} B. \qquad (۶-۴)$$

این‌کار را می‌توان با ترسیم یک گراف وابستگی با یال (C, D) به شرط وجود یک قانون واحد $C \to D$ در گرامر انجام داد؛ آنگاه اگر یک قدم بین A و B وجود داشته باشد، (۶-۴) صدق می‌کند. گرامر جدید $\hat{G}$، ابتدا با قراردادن تمام قوانین غیر واحد موجود در P به $\hat{P}$ بوجود می‌آید. سپس، به ازای تمامی A و B که در رابطه (۶-۴) برقرار می‌شود، قانون:

$$A \to y_1 \mid y_2 \mid \cdots \mid y_n$$

را به $\hat{P}$ اضافه می‌کنیم که در آن، $B \to y_1 \mid y_2 \mid \cdots y_n$ مجموعه تمام قوانینی عضو $\hat{P}$ است که B را در طرف چپ خود دارند. توجه داشته باشید که چون $B \to y_1 \mid y_2 \mid \cdots \mid y_n$ عضو $\hat{P}$ می‌باشد، هیچ کدام از y_iها تک‌متغیری نبوده و به همین دلیل، هرگز قانون‌واحد در مرحله آخر تولید نمی‌شود.

برای اثبات معادل بودن گرامر بدست آمده با گرامر اولیه، می‌توان از استدلال قضیه ۶-۱ استفاده کرد. ■

مثال ۶-۶

تمام قوانین واحد را از گرامر

$$S \to Aa \mid B,$$
$$B \to A \mid bb,$$
$$A \to a \mid bc \mid B.$$

حذف کنید. گراف وابستگی به ازای قوانین واحد در شکل ۶-۳ داده شده است؛ مشاهده می‌کنید که $S \overset{*}{\Rightarrow} A, S \overset{*}{\Rightarrow} B, B \overset{*}{\Rightarrow} A, A \overset{*}{\Rightarrow} B$. به همین دلیل، به قوانین غیر واحد گرامر اولیه

۳. قوانین واحد را حذف کنید.
۴. قوانین بی‌فایده را حذف کنید.

به این ترتیب، هیچ یک از قوانین مذکور در گرامر نهایی وجود ندارند. و بنابراین، قضیه ثابت می‌شود. □

تمرین‌ها

۱. اثبات قضیه ۶-۱ را کامل کرده و نشان دهید که

$$S \overset{*}{\Rightarrow}_G w$$

به طور ضمنی به معنای

$$S \overset{*}{\Rightarrow}_G w$$

است.

۲. در مثال ۶-۱، با استفاده از گرامر اولیه و نهایی بدست آمده، یک درخت اشتقاق به ازای رشته $ababbac$ رسم کنید.

۳. نشان دهید که دو گرامر

$$S \to abAB \mid ba,$$
$$A \to aaa,$$
$$B \to aA \mid bb.$$

و

$$S \to abAaA \mid abAbb \mid ba,$$
$$A \to aaa.$$

هم‌ارز هستند.

۴. در قضیه ۶-۱، چرا باید متغیرهای A و B را متفاوت فرض کرد؟

۵. تمامی قوانین بی‌فایده را از گرامر

$$S \to aS \mid AB,$$
$$A \to bA,$$
$$B \to AA.$$

حذف کنید. این گرامر چه زبانی را تولید می‌کند؟

۶. قوانین بی‌فایده را از گرامر

$$S \to a \mid aA \mid B \mid C,$$
$$A \to aB \mid \lambda,$$
$$B \to Aa,$$
$$C \to cCD,$$
$$D \to ddd.$$

$$S \to Aa,$$
$$A \to a \mid bc,$$
$$B \to bb.$$

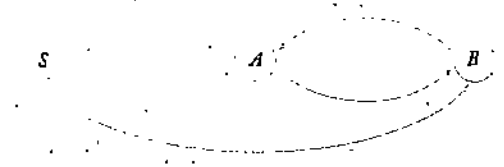
قوانین جدید

$$S \to a \mid bc \mid bb,$$
$$A \to bb,$$
$$B \to a \mid bc,$$

را اضافه می‌کنیم تا گرامر هم‌ارز

$$S \to a \mid bc \mid bb \mid Aa,$$
$$A \to a \mid bb \mid bc,$$
$$B \to a \mid bb \mid bc.$$

بوجود آید. توجه کنید که در اثر حذف قوانین واحد، B و قوانین مربوط به آن بی‌فایده شده‌اند.



شکل ۶-۳

□

در قضیه بعد تلاش کرده‌ایم تا با کنار هم قرار دادن تمامی این نتایج، امکان حذف قوانین بی‌فایده، قوانین λ وقوانین واحد، از گرامرهای زبان‌های مستقل‌ازمتن، را اثبات کنیم.

قضیه ۶-۵

L را یک زبان مستقل‌ازمتن فاقد λ فرض کنید. آنگاه یک گرامر مستقل‌ازمتن وجود خواهد داشت که L را تولید کرده و فاقد هرگونه قانون بی‌فایده، قانون λ و قانون واحد باشد.

اثبات: توسط روال‌های داده شده در قضایای ۶-۲، ۶-۳ و ۶-۴ این نوع قانون‌ها را به ترتیب حذف کنید. تنها نکته مهم اینکه، حذف یکی از انواع این قانون‌ها ممکن است باعث بوجود آمدن قوانین نوع دیگر شود؛ بعنوان مثال، روال حذف قوانین λ می‌تواند قوانین‌واحد جدیدی ایجاد کند. همچنین، قضیه ۶-۴ ایجاب می‌کند که گرامر فاقد هرگونه قوانین λ باشد. اما توجه داشته باشید که حذف قوانین‌واحد باعث تولید قوانین λ (به تمرین ۱۶ در انتهای این بخش مراجعه شود). و حذف قوانین بی‌فایده باعث تولید قوانین λ یا قوانین واحد نمی‌شود (به تمرین ۱۷ در انتهای این بخش مراجعه شود). بنابراین، می‌توان به ترتیب زیر، تمام قوانین نامطلوب را حذف کرد:

۱. قوانین λ را حذف کنید.

حذف کنید.

۷. تمامی قوانین λ را از گرامر

$$S \to AaB \mid aaB,$$
$$A \to \lambda,$$
$$B \to bbA \mid \lambda.$$

حذف کنید.

۸. تمامی قوانین‌واحد، قوانین بی‌فایده و قوانین λ را، از گرامر

$$S \to aA \mid aBB,$$
$$A \to aaA \mid \lambda,$$
$$B \to bB \mid bbC,$$
$$C \to B.$$

حذف کنید. این گرامر چه زبانی را تولید می‌کند؟

۹. تمامی قوانین‌واحد را، از گرامر تمرین ۶ حذف کنید.

۱۰. اثبات قضیه ۶-۳ را کامل کنید.

۱۱. اثبات قضیه ۶-۴ را کامل کنید.

۱۲. با استفاده از روش ساخت موجود در قضیه ۶-۳، قوانین λ را از گرامر مثال ۵-۴ حذف کنید. گرامر حاصل چه زبانی را تولید می‌کند؟

۱۳. گرامر G با قوانین

$$S \to A \mid B,$$
$$A \to \lambda,$$
$$B \to aBb,$$
$$B \to b.$$

را در نظر بگیرید. با اعمال الگوریتم قضیه ۶-۳ در G، گرامر $\hat{G}$ را بسازید. چه تفاوتی بین $L(G)$ و $L(\hat{G})$ وجود دارد؟

۱۴. فرض کنید که G یک گرامر مستقل‌ازمتن با شرط $\lambda \in L(G)$ باشد. نشان دهید که با بکاربردن روش ساخت موجود در قضیه ۶-۳، می‌توان گرامر جدیدی به نام $\hat{G}$ بدست آورد بطوری که $L(\hat{G}) = L(G) - \{\lambda\}$.

۱۵. نمونه‌ای از حالتی را ذکر کنید که حذف قوانین λ باعث تولید قوانین واحد می‌شود که قبلاً وجود نداشته‌اند.

۱۶. فرض کنید G گرامر فاقد قوانین λ باشد که احتمالاً تعدادی قانون‌واحد دارد. نشان دهید که ساخت گرامر بوسیله قضیه ۶-۴، باعث تولید هیچ قانون λ نخواهد شد.

۱۷. نشان دهید که اگر یک گرامر فاقد هرگونه قانون λ و قانون واحد باشد، آنگاه برای آن گرامر

حذف قوانین بی‌فایده به کمک ساختار قضیه ۶-۲، باعث ایجاد این قانون‌ها نخواهد شد.

۱۸. در اثبات قضیه ۶-۱ ادعا شد که متغیر B را می‌توان به محض ایجاد حذف کرد. این ادعا را ثابت کنید.

۱۹. فرض کنید که در گرامر $G = (V, T, S, P)$ قانونی به فرم

$$A \to xy,$$

وجود داشته باشد که در آن، $x, y \in (V \cup T)^+$. اثبات کنید که اگر این قانون با

$$A \to By,$$
$$B \to x,$$

با شرط $B \notin V$ جایگزین شود، آنگاه گرامر حاصل معادل با گرامر اولیه خواهد بود.

۲۰. روال مطرح شده در قضیه ۶-۲ در مورد حذف قوانین بی‌فایده را در نظر بگیرید. ترتیب دو بخش را معکوس کنید؛ یعنی ابتدا متغیرهایی را حذف کنید که بوسیله S قابل دسترسی نیستند و سپس آنهایی را که هیچ رشته پایانی بوجود نمی‌آورند. آیا روال جدید بازهم درست کار می‌کند؟ در صورت مثبت بودن جواب، آنرا اثبات کنید. در غیر اینصورت، یک مثال نقض ذکر کنید.

۲۱. می‌توان با ارائه مفهوم **پیچیدگی** گرامر، اصطلاح ساده‌سازی را دقیق‌تر تعریف کرد. این می‌تواند به روش‌های متعددی انجام بگیرد، یکی از این روشها روی طول رشته‌های بکار رفته در قوانین بکار می‌رود به عنوان نمونه می‌توان از رابطهٔ زیر استفاده کرد:

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

نشان دهید که حذف قوانین بی‌فایده همواره باعث کاهش پیچیدگی در این مورد می‌شود. آیا این ادعا درباره حذف قوانین λ و قوانین واحد هم برقرار است؟

۲۲. گرامر مستقل‌ازمتن G در صورتی برای زبان مفروض L کمینه خوانده می‌شود که به ازای هر $\hat{G}$ تولید کننده L، $complexity(G) \le complexity(\hat{G})$. به کمک مثال نشان دهید که حذف قوانین بی‌فایده لزوماً موجب تولید گرامرهای کمینه نخواهد شد.

۲۳. * نتیجه زیر را اثبات کنید.

$G = (V, T, S, P)$ را یک گرامر مستقل‌ازمتن فرض کنید. مجموعه قوانینی را که در سمت چپ آنها یک متغیر (مثل A) وجود داشته باشد، به دو زیرمجموعه مجزای

$$A \to Ax_1 \mid Ax_2 \mid \cdots \mid Ax_n,$$

$$A \to y_1 \mid y_2 \mid \cdots \mid y_m,$$

تبدیل کنید که در آن، x_i, y_i در $(V \cup T)^+$ قرار دارد، بطوریکه A پیشوند هیچکدام از y_iها نیست. گرامر $\hat{G} = (V \cup \{Z\}, T, S, \hat{P})$ را در نظر بگیرید که در آن، $Z \notin V$ و $\hat{P}$ با جایگزینی قوانین

$$A \to y_i \mid y_i Z, \quad i = 1,2,\ldots,m,$$
$$Z \to x_i \mid x_i Z, \quad i = 1,2,\ldots,n,$$

بجای تمامی قوانین موجود در P که A در سمت چپ آنها قرار دارد، بدست می‌آید.

آنگاه $L(G) = L(\hat{G})$.

۲۴. با استفاده از نتیجه تمرین قبل، گرامر

$$A \to Aa \mid aBc \mid \lambda,$$
$$B \to Bb \mid bc.$$

را به صورتی بازنویسی کنید که دیگر قوانینی به فرم $A \to Ax$ یا $B \to Bx$ در آن وجود نداشته باشد.

۲۵. *_صورت دیگری از تمرین ۲۳ را اثبات کنید. فرض کنید مجموعه قوانینی که متغیر A در طرف چپ آنها قرار گرفته است، به دو زیرمجموعه مجزای

$$A \to x_1 A \mid x_2 A \mid \cdots \mid x_n A$$

و

$$A \to y_1 \mid y_2 \mid \cdots \mid y_m,$$

تقسیم شود که در آن، A پسوند هیچکدام از y_i نباشد. نشان دهید که گرامر حاصل از جایگزینی این قانون‌ها با

$$A \to y_i \mid Z y_i, \quad i = 1,2,\ldots,m,$$
$$Z \to x_i \mid Z x_i, \quad i = 1,2,\ldots,n,$$

هم‌ارز با گرامر اولیه است.

۶-۵ دو فرم نرمال مهم

در گرامرهای مستقل‌ازمتن می‌توان از انواع مختلف فرم‌های نرمال موجود استفاده کرد. برخی از این فرم‌ها، به دلیل کاربرد گسترده، بسیار مورد مطالعه قرار گرفته‌اند. در ادامه دو مورد از آنها را بطور مختصر بررسی خواهیم کرد.

فرم نرمال چامسکی

در این فرم نرمال، تعداد سمبل‌ها و متغییرهای طرف راست قانون، بسیار محدود است. در یک حالت خاص، فرم جمله‌ای واقع در طرف راست قانون، حداکثر دو سمبل دارد. بعنوان یکی از نمونه‌های این نوع می‌توان به فرم نرمال چامسکی اشاره کرد.

تعریف ۶-۴

یک گرامر مستقل‌ازمتن در صورتی در فرم نرمال چامسکی خواهد بود که تمام قوانین آن به فرم

$$A \to BC$$

یا

$$A \to a,$$

باشند که در آن، A, B, C عضو V بوده و a عضو T است.

مثال ۶-۱۱

گرامر

$$S \to AS \mid a,$$
$$A \to SA \mid b$$

در فرم نرمال چامسکی است، اما گرامر

$$S \to AS \mid AAS,$$
$$A \to SA \mid aa$$

نیست. شرایط موجود در تعریف ۶-۴ در هیچکدام از قوانین $S \to AAS$ و $A \to aa$ برقرار نیست. □

قضیه ۶-۶

هر گرامر مستقل‌ازمتن $G = (V, T, S, P)$ با شرط $\lambda \notin L(G)$ دارای گرامری هم‌ارز، در فرم نرمال چامسکی به صورت $\hat{G} = (\hat{V}, \hat{T}, S, \hat{P})$ است.

اثبات: ضمن حفظ کلیت بحث، براساس قضیه ۶-۵ می‌توان G را فاقد هرگونه قانون λ و قانون‌واحد در نظر گرفت. ساخت $\hat{G}$ در دو فاز انجام می‌شود.

فاز ۱: گرامر $G_1 = (V_1, T, S, P_1)$ را، از G می‌سازیم. به این منظور، تمام قوانین موجود در P به فرم

$$A \to x_1 x_2 \cdots x_n, \qquad (۶-۵)$$

را در نظر می‌گیریم که در آن، هر x_i نشانه‌ای است که یا در V قرار دارد و یا در T. اگر $n = 1$، آنگاه چون هیچ قانون‌واحدی وجود ندارد، x_1 باید پایانی باشد. در این حالت، قانون را در P_1 قرار می‌دهیم. در صورتیکه $n \geq 2$، به ازای هر $a \in T$ متغیرهای جدید B_a را معرفی می‌کنیم. به ازای هر یک از قوانین P که به فرم (۶-۵) باشند، قانون

$$A \to C_1 C_2 \cdots C_n,$$

را در P_1 قرار می‌دهیم که در آن:

$$C_i = x_i \quad \text{اگر } x_i \text{ در } V \text{ باشد،}$$

و

$$C_i = B_a \quad \text{اگر } x_i = a$$

همچنین برای هر B_a قانون

$$B_a \to a$$

را در P_1 قرار می‌دهیم. این بخش از الگوریتم باعث حذف پایانی‌های قوانینی می‌شود که طول طرف راست آنها بیشتر از یک بوده و آنها را با متغیرهای جدیداً تعریف شده، جایگزین می‌کند. در پایان این مرحله، گرامر G_1 بوجود می‌آید که تمام قوانین آن به فرم:

$$A \to a, \tag{۶-۶}$$

یا

$$A \to C_1C_2\cdots C_n, \tag{۶-۷}$$

هستند، بطوریکه $C_i \in V_1$.

به راحتی از قضیه ۶-۱ می‌توان نتیجه گرفت که

$$L(G_1) = L(G).$$

فاز ۲ : در فاز دوم، با معرفی متغیرهای بیشتر، سعی می‌کنیم تا در هر جا که لازم باشد، طول طرف راست قوانین را کاهش دهیم. ابتدا، تمام قوانین به فرم (۶-۶) و تمام قوانین به فرم (۶-۷) با شرط $n = 2$ را در $\hat{P}$ قرار می‌دهیم. برای $n > 2$، متغیرهای جدید $D_1, D_2, \ldots$ را معرفی کرده و قوانین

$$\begin{aligned} A &\to C_1D_1, \\ D_1 &\to C_2D_2, \\ &\vdots \\ D_{n-2} &\to C_{n-1}C_n \end{aligned}$$

را در $\hat{P}$ قرار می‌دهیم. مشخص است که گرامر قانون $\hat{G}$ در فرم نرمال چامسکی خواهد بود. با بکار بستن دوبارهٔ قضیه ۶-۱ می‌توان نشان داد که $L(G_1) = L(\hat{G})$ و بنابراین

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

این استدلال تقریباً غیررسمی را می‌توان کوتاه‌تر کرد. پیشنهاد می‌کنیم این کار را خودتان بعنوان تمرین انجام دهید. ■

مثال ۶-۱۱

گرامر مفروض با قوانین

$$\begin{aligned} S &\to ABa, \\ A &\to aab, \\ B &\to Ac \end{aligned}$$

را به فرم نرمال چامسکی تبدیل کنید.

براساس شرایط موجود ساخت که در قضیهٔ ۶-۶ آمده است، در این گرامر هیچ قانون λ یا قانون-واحدی وجود ندارد.

در مرحله ۱، متغیرهای جدید B_a, B_b, B_c را معرفی کرده و با استفاده از الگوریتم، قوانین:

$$\begin{aligned} S &\to ABB_a, \\ A &\to B_aB_aB_b, \\ B &\to AB_c, \\ B_a &\to a, \\ B_b &\to b, \\ B_c &\to c. \end{aligned}$$

را بدست می‌آوریم.

در مرحله دوم، با معرفی متغیرهای دیگر، دو قانون اول را به فرم نرمال چامسکی تبدیل کرده و گرامر نهایی را به صورت زیر بدست می‌آوریم.

$$\begin{aligned} S &\to AD_1, \\ D_1 &\to BB_a, \\ A &\to B_aD_2, \\ D_2 &\to B_aB_b, \\ B &\to AB_c, \\ B_a &\to a, \\ B_b &\to b, \\ B_c &\to c. \end{aligned}$$

■

فرم نرمال گریباخ

فرم نرمال گریباخ یکی دیگر از این فرم‌های مفید برای گرامرهای مستقل‌از متن است. در این فرم نرمال، برای طول سمت راست قوانین محدودیتی نداریم بلکه در مورد محل بروز پایانی‌ها و متغیرها. محدودیت خاصی قائل می‌شویم. استدلال‌های مربوط به فرم نرمال گریباخ کمی پیچیده است و خیلی

شفاف نیست. به همین دلیل، ساخت گرامر متناظر با یک گرامر مستقل‌ازمتن در فرم نرمال گریباخ کمی مشکل‌تر است. بنابراین، زیاد روی این موضوع تمرکز نمی‌کنیم. اما باید متذکر شد، که فرم نرمال گریباخ دارای نتایج نظری و کاربردی بسیاری است.

[redacted]

گوییم یک گرامر مستقل‌ازمتن در فرم نرمال گریباخ است هرگاه تمام قوانین آن به فرم

$$A \to ax,$$

باشند که در آن، $a \in T$ و $x \in V^*$.

با مقایسه این تعریف، با تعریف ۵-۴ مشاهده می‌کنیم که فرم $A \to ax$ در هر تا فرم نرمال گریباخ و s-گرامرها به چشم می‌خورد؛ اما در فرم نرمال گریباخ محدودیت وقوع حداکثر یک مرتبه‌ای زوج (A, a) وجود ندارد. این آزادی، کلیتی به فرم نرمال گریباخ می‌بخشد که در s-گرامرها وجود ندارد.

اگر گرامری که در اختیار داریم به فرم نرمال گریباخ نباشد، می‌توان آنرا به کمک روش‌های فوق‌الذکر به فرم نرمال گریباخ تبدیل کرد. در دو مثال ساده زیر از این روش‌ها استفاده شده است.

گرامر

$$S \to AB,$$
$$A \to aA \mid bB \mid b,$$
$$B \to b$$

در فرم نرمال گریباخ نیست. اما می‌توان با استفاده از جایگزینی قضیه ۶-۱، گرامر معادل

$$S \to aAB \mid bBB \mid bB,$$
$$A \to aA \mid bB \mid b,$$
$$B \to b$$

را به فرم نرمال گریباخ بدست آورد.

■

گرامر

$$S \to abSb \mid aa$$

را به فرم نرمال گریباخ تبدیل کنید.

در این مورد می‌توان از ابزاری مشابه آنچه در ساخت فرم نرمال چامسکی استفاده شد، بهره گرفت. ما متغیرهای جدید A و B را معرفی می‌کنیم که اساساً به ترتیب، مترادف با a و b هستند. جایگزینی پایانی‌ها با متغیرهای مربوط به آنها، باعث ایجاد گرامر هم‌ارز

$$S \to aBSB \mid aA,$$
$$A \to a,$$
$$B \to b,$$

می‌شود که در فرم نرمال چامسکی است.

■

اما در مجموع باید گفت که تبدیل یک گرامر مفروض به فرم نرمال گریباخ و اثبات انجام‌پذیر بودن همیشگی آن، به راحتی قابل انجام نیست. در این فصل سعی کردیم تا با معرفی فرم نرمال گریباخ، بحث تکنیکی یکی از نتایج مهم در فصل بعد را ساده کنیم. اما چون مفهوم فرم نرمال گریباخ در بحث بعدی ما نقشی ندارد، فقط نتیجه کلی زیر را بیان کرده و از اثبات آن صرف‌نظر می‌کنیم.

قضیه ۶-۷

به ازای هر گرامر مستقل‌ازمتن G با شرط $\lambda \notin L(G)$، یک گرامر معادل $\hat{G}$ به فرم نرمال گریباخ وجود دارد. ■

تمرین‌ها

۱. جزئیات اثبات قضیه ۶-۶ را کامل کنید.

۲. گرامر $S \to aSb \mid ab$ را به فرم نرمال چامسکی تبدیل کنید.

۳. گرامر $S \to aSaA \mid A, \quad A \to abA \mid b$، را به فرم نرمال چامسکی تبدیل کنید.

۴. گرامری با قوانین

$$S \to abAB,$$
$$A \to bAB \mid \lambda,$$
$$B \to BAa \mid A \mid \lambda$$

را به فرم نرمال چامسکی تبدیل کنید.

۵. گرامر

$$S \to AB \mid aB,$$
$$A \to aab \mid \lambda,$$
$$B \to bbA$$

را به فرم نرمال چامسکی تبدیل کنید.

۶. فرض کنید $G=(V,T,S,P)$ یک گرامر مستقل‌ازمتن بدون هیچ قانون λ یا قانون‌واحدی باشد. بعلاوه، فرض کنید k حداکثر تعداد سمبل‌های موجود در طرف راست قوانین P باشد. نشان دهید که گرامر معادلی در فرم نرمال چامسکی و با حداکثر $|T|+|P|(k-1)$ قانون تولید وجود دارد.

۷. گراف وابستگی گرامر تمرین ۴ را رسم کنید.

۸. زبان خطی، زبانی است که به ازای آن یک گرامر خطی وجود داشته باشد(برای تعریف گرامر خطی، به مثال ۳-۱۴ مراجعه شود). فرض کنید L یک زبان خطی فاقد λ باشد. نشان دهید که گرامر $G=(V,T,S,P)$ وجود دارد که تمام قوانین آن به یکی از فرم‌های

$$A\to aB,$$
$$A\to Ba,$$
$$A\to a$$

است که در آن، $a\in T$، $A,B\in V$، بطوریکه $L=L(G)$.

۹. نشان دهید که به ازای هر گرامر مستقل‌ازمتن $G=(V,T,S,P)$ گرامر هم‌ارزی وجود دارد که در آن، تمام قوانین به فرم

$$A\to aBC,$$

یا

$$A\to\lambda,$$

هستند، بطوریکه $a\in\Sigma\cup\{\lambda\}$، $A,B,C\in V$.

۱۰. گرامر

$$S\to aSb\,|\,bSa\,|\,a\,|\,b$$

را به فرم نرمال گریباخ تبدیل کنید.

۱۱. گرامر زیر را به فرم نرمال گریباخ تبدیل کنید.

$$S\to aSb\,|\,ab.$$

۱۲. گرامر

$$S\to ab\,|\,aS\,|\,aaS$$

را به فرم نرمال گریباخ تبدیل کنید.

۱۳. گرامر

$$S\to ABb\,|\,a,$$
$$A\to aaA\,|\,B,$$
$$B\to bAb$$

را به فرم نرمال گریباخ تبدیل کنید.

۱۴. آیا قوانین تمام گرامرهای خطی را می‌توان به فرم $A\to ax$ تبدیل کرد، بطوریکه $a\in T$، $x\in V\cup\{\lambda\}$؟

۱۵. یک گرامر مستقل‌ازمتن، را در صورتی در فرم دوم استاندارد می‌گوئیم که تمام قوانین تولید در فرم $S\to aSA$، $C\to aBC$

$$A\to aBC,$$
$$A\to aB,$$
$$A\to a.$$

صدق کنند، بطوریکه $A,B,C\in V$ و $a\in T$.

گرامر $G=(\{S,A,B,C\},\{a,b\},S,P)$ را که در آن، P از روابط

$$A\to bABC,$$
$$B\to b,$$

بدست می‌آید، به فرم دوم استاندارد تبدیل کنید.

۱۶.* فرم دوم استاندارد قابل تعمیم است. اثبات کنید به ازای هر گرامر مستقل‌ازمتن G که شامل λ نمی‌باشد، یک گرامر معادل در فرم دوم استاندارد وجود دارد.

یک الگوریتم عضویت برای گرامرهای مستقل‌ازمتن*

در بخش ۵-۲، بدون هیچ توضیح بیشتری، ادعا کردیم که تجزیه رشته مفروض w توسط برخی الگوریتم‌های عضویت و تجزیه موجود برای زبانهای مستقل از متن تقریباً طی $|w|^3$ مرحله انجام می‌شود. اینک دانسته‌های ما در حدی است که می‌توانیم این ادعا را ثابت کنیم. الگوریتمی که در اینجا توصیف می‌کنیم به نام طراحان خود، یعنی D.H.Younger J.Cocke و T.Kasami، الگوریتم CYK خوانده می‌شود. این الگوریتم به صورت زیر، مسأله را به بخش‌های کوچکتر تقسیم کرده و فقط در صورتی کار می‌کند که گرامر در فرم نرمال چامسکی باشد. فرض کنید که ما گرامر $G=(V,T,S,P)$ در فرم نرمال چامسکی و رشته

$$w=a_1a_2\cdots a_n$$

را داریم.

زیررشته‌های

$$w_{ij}=a_i\cdots a_j,$$

و زیرمجموعه‌هایی از V را بصورت زیر:

$$V_{ij}=\{A\in V : A \overset{*}{\Rightarrow} w_{ij}\}$$

تعریف می‌کنیم. مشخص است که $w\in L(G)$ اگر و تنها اگر $S\in V_{1n}$.

برای محاسبه V_{ij}، مشاهده می‌کنیم که $A\in V_{ij}$ اگر و تنها اگر یکی از قوانین G به فرم $A\to a_i$ باشد. بنابراین، می‌توان با بررسی w و قوانین گرامر، V_{ii} را به ازای تمام $1\le i\le n$ محاسبه نمود. در ادامه، توجه داشته باشید که به ازای $j>i$، A می‌تواند w_{ij} را اشتقاق کند اگر و تنها اگر یک قانون $A\to BC$ با فرض اینکه $B\overset{*}{\Rightarrow} w_{ik}$ و $C\overset{*}{\Rightarrow} w_{k+1,j}$ به ازای برخی k ها با شرط $i\le k<j$ وجود داشته باشد. به بیان دیگر،

$$V_{ij}=\bigcup_{k\in\{i,i+1,\dots,j-1\}}\{A: B\in V_{ik}, C\in V_{k+1,j} \text{ با شرط } A\to BC\} \qquad (۸-۶)$$

با رعایت دنباله قدم‌های زیر می‌توان از رابطه فوق برای محاسبه تمام V_{ij}ها استفاده کرد :

۱. $V_{11},V_{22},\dots,V_{nn}$، را محاسبه کنید.

۲. $V_{12},V_{23},\dots,V_{n-1,n}$، را محاسبه کنید.

۳. $V_{13},V_{24},\dots,V_{n-2,n}$، را محاسبه کنید.

و الی آخر.

مثال ۶-۱۱

وجود یا عدم وجود رشته $w=aabbb$ در زبان تولید شده بوسیله گرامر زیر را تحقیق کنید.

$$S\to AB,$$
$$A\to BB\mid a,$$
$$B\to AB\mid b.$$

ابتدا یادآوری می‌کنیم که $w_{11}=a$، بنابراین V_{11} مجموعه تمام متغیرهایی است که فوراً a را اشتقاق می‌کنند، یعنی $V_{11}=\{A\}$. از آنجاییکه $w_{22}=a$، داریم که $V_{22}=\{A\}$ و به همین ترتیب،

$$V_{11}=\{A\},V_{22}=\{A\},V_{33}=\{B\},V_{44}=\{B\},V_{55}=\{B\}.$$

اینک از (۶-۸) می‌توان نتیجه گرفت که

$$V_{12}=\{A: A\to BC, B\in V_{11}, C\in V_{22}\}.$$

از آنجایی‌که $V_{11}=\{A\}$ و $V_{22}=\{A\}$، این مجموعه شامل تمام متغیرهای واقع در طرف چپ قوانینی است که در طرف راست آنها AA قرار دارد. چون هیچ قانونی به این فرم وجود ندارد، V_{12} تهی است. سپس،

$$V_{23}=\{A: A\to BC, B\in V_{22}, C\in V_{33}\}.$$

بنابراین، AB باید در طرف راست قرار داشته باشد و از اینرو $V_{23}=\{S,B\}$ راداریم. در نهایت، با یک استدلال ساده و صریح مشابه بالا

$$V_{12}=\varnothing, V_{23}=\{S,B\}, V_{34}=\{A\}, V_{45}=\{A\},$$
$$V_{13}=\{S,B\}, V_{24}=\{A\}, V_{35}=\{S,B\},$$
$$V_{14}=\{A\}, V_{25}=\{S,B\},$$
$$V_{15}=\{S,B\}.$$

بدست می‌آید و بنابراین، $w\in L(G)$.

به کمک الگوریتم CYK که در بالا شرح داده شد، می‌توان راجع به عضویت یا عدم عضویت رشته‌ها در تمامی زبان‌هایی که بوسیله گرامرهای در فرم نرمال چامسکی تولید می‌شوند، تصمیم‌گیری کرد. با افزودن برخی جزئیات درباره نحوه محاسبهٔ اعضای V_{ij}، می‌توان از این الگوریتم به عنوان روش تجزیه استفاده نمود. برای اثبات اینکه الگوریتم عضویت CYK طی $O(n^3)$ مرحله انجام می‌شود، توجه کنید که دقیقاً باید $n(n+1)/2$ مجموعه از V_{ij}ها محاسبه شوند. در هر مرحله حداکثر n بار (۶-۸) بررسی می‌شود و بنابراین نتیجه ادعا شده، اثبات می‌شود.

تمرین‌ها

۱. با استفاده از الگوریتم CYK تعیین کنید آیا رشته‌های $aabb$ ، $aabba$ و $abbbb$ در زبان تولید شده توسط گرامر مثال ۶-۱۱ وجود دارند یا خیر.

۲. با استفاده از الگوریتم CYK و گرامر مثال ۶-۱۱، یکی از تجزیه‌های رشته aab را پیدا کنید.

۳. با استفاده از شیوه بکار رفته در تمرین ۲، نشان دهید که چگونه می‌توان از الگوریتم عضویت CYK بعنوان روش تجزیه استفاده نمود؟

۴. ** - با استفاده از نتیجه تمرین۳، یک برنامه کامپیوتری برای تجزیه تمامی گرامرهای مستقل‌ازمتن به فرم نرمال چامسکی بنویسید.

فصل ۷

اتوماتای پشته‌ای

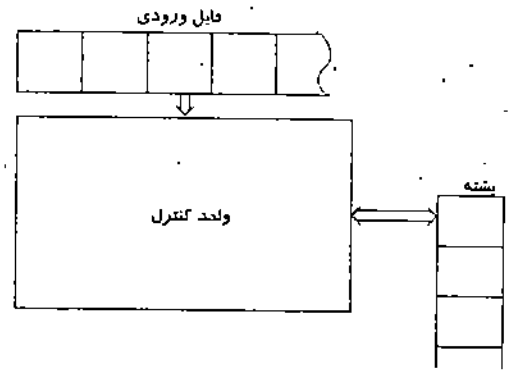
توصیف زبان‌های مستقل‌ازمتن بوسیله گرامرهای مستقل‌ازمتن روشی مناسب است و به همین دلیل، در تعریف زبان‌های برنامه‌سازی از BNF استفاده می‌شود. حال این سؤال مطرح می‌شود که آیا اتوماتایی وجود دارند که بتوان به کمک آنها زبان‌های مستقل‌ازمتن را تعریف کرد؟ همانطور که دیدیم، اتوماتای متناهی قادر به شناخت تمامی زبان‌های مستقل‌ازمتن نیستند. از نظر شهودی، دلیل این امر را می‌توان به حافظه‌های بسیار متناهی این نوع اتوماتا نسبت داد، در حالی که تشخیص رشته‌های یک زبان مستقل-ازمتن گاهی اوقات مستلزم ذخیره‌سازی مقدار نامحدودی اطلاعات است. بعنوان مثال، برای بررسی یکی از رشته‌های زبان $L = \{a^n b^n : n \geq 0\}$، علاوه بر بررسی تمام a هایی که قبل از اولین b قرار می‌گیرند، باید تعداد a ها را نیز شمارش کنیم. بدلیل آنکه n نامحدود است، انجام کار شمارش با یک حافظه متناهی ممکن نیست. بلکه نیازمند اتوماتی هستیم که بتواند بدون هیچ محدودیتی این شمارش را انجام دهد. در عین حال، با نگاه به مثال‌هایی از قبیل $\{ww^R\}$ در می‌یابیم که ماشین مذکور، علاوه بر نامحدود بودن قدرت شمارش، باید قدرت ذخیره‌سازی و همچنین تطابق دنباله‌ای از سمبل‌ها با ترتیب معکوس را، داشته باشد. بنابراین، می‌توان از یک پشته بعنوان مکانیزم ذخیره‌سازی استفاده کرده و از امکان ذخیره‌سازی نامحدود، که از مزایای منحصربفرد عملیات پشته‌ای محسوب می‌شود، بهره گرفت. به این ترتیب دسته‌ای از مکانیزم‌ها به نام اتوماتای پشته‌ای (pda) بوجود می‌آید.

در این فصل، به بررسی ارتباط بین اتوماتای پشته‌ای و زبان‌های مستقل‌ازمتن می‌پردازیم. از اینرو، ابتدا نشان می‌دهیم که اگر اتوماتای پشته‌ای به صورت نامعین عمل کنند، اتوماتای حاصل فقط خانواده زبان‌های مستقل‌ازمتن را می‌پذیرند. سپس، مشاهده می‌کنیم که دیگر مفهومی به نام هم‌ارزی بین اتوماتای پشته‌ای معین و نامعین وجود ندارد. این دسته اتوماتای پشته‌ای معین، ضمن معرفی خانواده جدیدی از زبان‌ها به نام زبان‌های مستقل‌ازمتن معین، زیرمجموعه مناسبی برای زبان‌های مستقل‌ازمتن

ایجاد می‌کند. از آنجایی‌که شناخت این خانواده مهم، نقش اساسی برای درک رفتار زبان‌های برنامه‌سازی ایفا می‌کند، فصل را با تعریف مختصری از گرامرهای مربوط به زبان‌های مستقل‌ازمتن معین به پایان خواهیم رساند.

۷-۱ اتوماتای پشته‌ای نامعین

در شکل۷-۱، مدل کلی از یک اتومات پشته‌ای نمایش داده شده است. با هر حرکت واحد کنترل، یکی از سمبل‌های فایل ورودی خوانده شده و همزمان با آن، محتویات پشته با انجام عملیات معمول بر روی پشته تغییر می‌کند. تمامی حرکات واحد کنترل برحسب سمبل ورودی جاری و همچنین، سمبلی که هم‌اکنون در بالای پشته قرار گرفته، تعیین می‌شود. در اثر این حرکت، واحد کنترل در حالت جدیدی قرار گرفته و یک تغییر در بالای پشته انجام می‌شود. فعلاً فقط روی مدل خاصی از اتوماتای پشته‌ای تمرکز می‌کنیم که بعنوان پذیرنده عمل می‌کنند.



شکل ۷-۱

تعریف اتومات پشته‌ای

با تدوین‌کردن ایده شهودی که در بالا ارائه شد، به تعریف دقیقی از یک اتومات پشته‌ای می‌رسیم.

تعریف ۷-۱

پذیرنده پشته‌ای نامعین (npda) بوسیله هفت‌تایی

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

تعریف می‌شود که در آن،

Q مجموعه متناهی از حالت‌های داخلی واحد کنترل است،

Σ الفبای ورودی است،

Γ مجموعه متناهی از سمبل‌ها به نام الفبای پشته است،

زیرمجموعه‌های متناهی از $Q\times\Gamma^*$ است، $\delta:Q\times(\Sigma\cup\{\lambda\})\times\Gamma\to Q\times\Gamma^*$ تابع انتقال است،

$q_0\in Q$ حالت شروع واحد کنترل است،

$z\in\Gamma$ سمبل ته پشته است،

و $F\subseteq Q$ مجموعه حالات پایانی است.

به دلیل ظاهر صوری پیچیده دامنه و برد δ، باید بیشتر در مورد آن توضیح دهیم. در آرگومانهای δ باید حالت جاری واحد کنترل، سمبل ورودی جاری و سمبل جاری در بالای پشته مشخص شود. به این ترتیب، مجموعه‌ای از زوج‌های (q,x) بوجود می‌آید که در آن، q حالت بعدی واحد کنترل و x رشته‌ای است که به جای سمبل قبلی، در بالای پشته قرارگرفته است. توجه داشته باشید که مولفهٔ دوم δ ممکن است λباشد؛ به این معنا که می‌توانیم در یکی از حرکات از سمبل ورودی استفاده نکنیم. این دست حرکت‌ها اصطلاحاً انتقال λ خوانده می‌شوند. همچنین توجه داشته باشید که δ به گونه‌ای تعریف می‌شود که حتماً یک سمبل پشته در آن وجود دارد؛ بنابراین در صورت تهی بودن پشته، هیچ حرکتی امکان‌پذیر نخواهد بود. نکته آخر اینکه، چون $Q\times\Gamma^*$ یک مجموعه نامتناهی بوده و در نتیجه زیرمجموعه‌های نامتناهی دارد، شرط زیرمجموعه نامتناهی بودن δ یک شرط لازم است. هرچند npdaها ممکن است انتخاب‌های مختلفی برای حرکت داشته باشند، ولی این انتخاب‌های مختلف باید به مجموعه‌ای متناهی از رخدادها محدود گردد.

مثال ۷-۱

فرض کنید که مجموعه قوانین انتقال یک npda به صورت

$$\delta(q_1,a,b)=\{(q_2,cd),(q_3,\lambda)\}$$

باشد. هرگاه واحد کنترل در حالت q_1 قرار گیرد، سمبل ورودی خوانده شده a و سمبل واقع در بالای پشته b باشد، آنگاه یکی از دو حالت زیر اتفاق می‌افتد: (الف) واحد کنترل به حالت q_2 رفته و رشته cd، جایگزین b در بالای پشته می‌شود، یا (ب) واحد کنترل به حالت q_3 رفته و سمبل b از بالای پشته حذف می‌شود. در نشانه‌گذاری این عملیات فرض می‌کنیم که درج رشته در پشته، از انتهای سمت راست رشته شروع شده و در ضمن، به صورت سمبل به سمبل انجام می‌شود.

■

مثال ۷-۲

یک npda با فرض

$$Q=\{q_0,q_1,q_2,q_3\},$$
$$\Sigma=\{a,b\},$$
$$\Gamma=\{0,1\},$$
$$z=0,$$
$$F=\{q_3\},$$

با حالت شروع q_0 و

$$\delta(q_0,a,0)=\{(q_1,10),(q_3,\lambda)\},$$
$$\delta(q_0,\lambda,0)=\{(q_3,\lambda)\},$$
$$\delta(q_1,a,1)=\{(q_1,11)\},$$
$$\delta(q_1,b,1)=\{(q_2,\lambda)\},$$
$$\delta(q_2,b,1)=\{(q_2,\lambda)\},$$
$$\delta(q_2,\lambda,0)=\{(q_3,\lambda)\}$$

را در نظر بگیرید. نحوه عملکرد این اتومات را تشریح کنید.

اولاً، توجه داشته باشید که در حالت کلی تابع انتقال مثال فوق برای برخی سمبل‌ها مشخص نیست. بعنوان مثال، هیچ حرکتی برای $\delta(q_0,b,0)$ داده نشده است. این حالت را می‌توان همانند آنچه قبلاً در مورد اتوماتای متناهی نامعین گفتیم، تفسیرکرد: انتقال‌های تخصیص‌نیافته را انتقال به مجموعه تهی و پیکربندی مرده‌ای برای npda در نظر می‌گیریم.

بعنوان مهمترین انتقالات این npda می‌توان به

$$\delta(q_1,a,1)=\{(q_1,11)\},$$

اشاره کرد که پس از خواندن یک سمبل a از ورودی، یک سمبل ۱ را به پشته اضافه می‌کند، و همچنین

$$\delta(q_2,b,1)=\{(q_2,\lambda)\},$$

که به محض برخورد با یک b از ورودی ، یک سمبل ۱ را از پشته حذف می‌کند. به کمک این دو مرحله، تعداد a ها شمارش شده و تعداد aها با تعداد b ها تطبیق داده می‌شود. واحد کنترل در حالت q_1 قرار داشته و پس از برخورد با اولین b ، به حالت q_2 می‌رود. این تضمین می‌کند که، هیچ b پیش از آخرین a قرار نخواهد گرفت. پس از تجزیه و تحلیل بقیه انتقالات، مشاهده می‌کنیم که تنها در صورتی npda به حالت نهایی q_3 ختم می‌شود که رشته ورودی جزء زبان

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

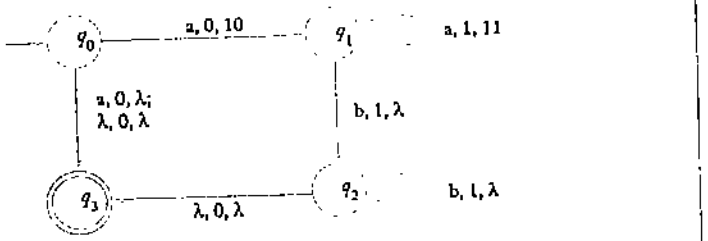
باشد. به دلیل شباهت npda فوق با اتوماتای متناهی، می‌توان گفت که این npda زبان مذکور را می‌پذیرد. اما قبل از طرح چنین ادعایی، باید منظور دقیق خود را از عبارت"پذیرش زبان توسط npda " بیان کنیم.

▫

در ادامه، از گراف‌های انتقال به عنوان روشی دیگر برای نمایش npda ها استفاده خواهیم کرد. در این شیوه، برای برچسب‌دار کردن یال‌های گراف از سه مؤلفه استفاده می‌شود: سمبل ورودی جاری، سمبل واقع در بالای پشته و رشته‌ای که در بالای پشته جایگزین می‌شود.

مثال ۲-۷

npda مثال ۲-۷ بوسیله گراف انتقال شکل ۲-۷ نمایش داده شده است.



شکل ۲-۷

هرچند گراف‌های انتقال برای توصیف npda ها بسیار مناسب هستند، کارآیی چندانی در ارائه استدلال‌ها ندارند. لزوم مراقبت از حالت‌های داخلی و محتویات پشته، اثربخشی و بکارگیری مناسب گراف‌های انتقال برای استدلال دقیق را کاهش می‌دهد. به همین دلیل، نمادگذاری مناسبی را برای توصیف پیکربندی‌های متوالی npda ها در حین پردازش رشته‌ها معرفی می‌کنیم. در هر مورد باید به عواملی از قبیل حالت جاری واحد کنترل، بخش خوانده نشده رشته ورودی و محتویات جاری پشته توجه کرد. این‌ها با هم بطور کامل تعیین کنندهٔ همهٔ راه‌های ممکنی است که npda می‌تواند ادامه دهد. سه تایی

$$(q,w,u),$$

که در آن، q حالت جاری واحد کنترل، w بخش خوانده نشده رشته ورودی و u محتویات پشته (چپ‌ترین سمبل بیانگر بالای پشته) است، پیکربندی لحظه‌ای یک اتومات پشته‌ای خوانده می‌شود. انتقال از یک پیکربندی لحظه‌ای به پیکربندی لحظه‌ای دیگر بوسیله نشانهٔ $\vdash$ نشان داده می‌شود؛ بنابراین

$$(q_1,aw,bx)\vdash(q_2,w,yx)$$

امکان‌پذیر است اگر و تنها اگر

$$(q_2,y)\in\delta(q_1,a,b).$$

برای نمایش حرکت طی تعداد دلخواهی مرحله، از نشانهٔ $\vdash^*$ استفاده می‌کنیم. عبارت

$$(q_1,w_1,x_1)\vdash^*(q_2,w_2,x_2)$$

بیانگر تغییر پیکربندی در طی چند مرحله است[1].

۱- البته به دلیل نامعین‌بودن، نیازی به انجام این دست تغییرات وجود ندارد.

اگر چندین اتوماتا بطور همزمان مورد بررسی قرار بگیرند، با استفاده از $\vdash_M$ تأکید می‌کنیم که حرکت مورد نظر بوسیله اتومات M انجام شده است.

زبان موردپذیرش در یک اتومات پشته‌ای

تعریف ۷-۲

فرض کنید $M=(Q,\Sigma,\Gamma,\delta,q_0,z,F)$ یک اتومات پشته‌ای نامعین باشد. زبان مورد پذیرش بوسیله M، توسط مجموعه

$$L(M)=\{w\in\Sigma^*:(q_0,w,z)\vdash_M(p,\lambda,u),p\in F,u\in\Gamma^*\}$$

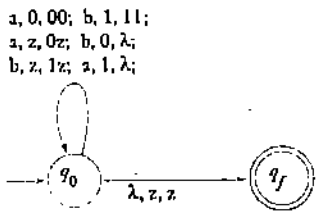
نمایش داده می‌شود. به بیان دیگر، زبان مورد پذیرش توسط M مجموعه تمام رشته‌هایی است که M با رسیدن به انتهای آن رشته‌ها می‌تواند در حالت پایانی قرار گیرد. توجه داشته باشید که محتوای نهائی پشته u ارتباطی با این تعریف ندارد.

مثال ۷-۴

یک npda برای زبان زیر بسازید.

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

مشابه مثال ۷-۲، برای حل این مسأله باید به کمک یک پشته، تعداد a ها و b ها را شمارش کنیم. تنها تفاوت این است که، در این مورد ترتیب a ها و b ها اهمیتی ندارد. به این منظور، می‌توان به محض خواندن یک a یک سمبل 0 را برای شمارش در پشته درج کرده و پس از خواندن یک b یک سمبل شمارش از پشته بیرون آورد. تنها مشکل این است که اگر در یکی از پیشوندهای w، تعداد b ها بیشتر از a باشد، نمی‌توان از 0 استفاده کرد. برای حل این مشکل می‌توان از یک سمبل شمارش منفی، مثل 1، برای شمارش b هایی استفاده کرد که باید بعداً با a ها تطابق داده شوند. جواب کامل مسأله در گراف انتقال شکل ۷-۳ آمده است.



شکل ۷-۳

برای پردازش رشته $baab$، npda حرکت‌های زیر را انجام می‌دهد:

$$(q_0,baab,z)\vdash(q_0,aab,1z)\vdash(q_0,ab,z)$$
$$\vdash(q_0,b,0z)\vdash(q_0,\lambda,z)\vdash(q_f,\lambda,z)$$

به همین دلیل رشته پذیرفته می‌شود.

■

مثال ۷-۵

برای ساخت npda که بتواند زبان

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

را بپذیرد، از این ویژگی بهره می‌گیریم که سمبل‌ها به ترتیب معکوس درج خود، از پشته برداشته می‌شوند. در حین خواندن بخش اول رشته یعنی w، سمبل بعدی را در پشته درج می‌کنیم(Push می‌کنیم). برای بخش دوم رشته یعنی w^R، سمبل ورودی جاری را با سمبل بالای پشته مقایسه کرده و اینکار را تا زمانیکه دو سمبل برابر هستند، ادامه می‌دهیم. از آنجایی که رشته‌ها برعکس ترتیب درج خود در پشته، از آن حذف می‌شوند، تست برابری کامل امکان‌پذیر است اگر و تنها اگر ورودی به فرم ww^R باشد.

یکی از نقاط ضعف آشکار این روش، عدم امکان تشخیص وسط رشته است؛ یعنی نمی‌دانیم که w در کجا به پایان رسیده و w^R از کجا آغاز می‌شود. می‌توان از ماهیت نامعین‌بودن اتومات برای حل این مشکل استفاده کرد. به کمک این ویژگی، npda قادر خواهد بود تا به درستی وسط رشته را حدس زده و حالت را در این نقطه عوض کند. یکی از جواب‌های مسأله فوق بوسیله $M=(Q,\Sigma,\Gamma,\delta,q_0,z,F)$ ارائه می‌شود که در آن،

$$Q=\{q_0,q_1,q_2\},$$
$$\Sigma=\{a,b\},$$
$$\Gamma=\{a,b,z\},$$
$$F=\{q_2\}.$$

تابع انتقال شامل چندین بخش است:

یک مجموعه برای درج کردن w به بالای پشته،

$$\delta(q_0,a,a)=\{(q_0,aa)\},$$
$$\delta(q_0,b,a)=\{(q_0,ba)\},$$
$$\delta(q_0,a,b)=\{(q_0,ab)\},$$
$$\delta(q_0,b,b)=\{(q_0,bb)\},$$
$$\delta(q_0,a,z)=\{(q_0,az)\},$$
$$\delta(q_0,b,z)=\{(q_0,bz)\},$$

یک مجموعه برای حدس زدن وسط رشته، یعنی محل انتقال از حالت q_0 به q_1،

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

یک مجموعه برای برابری w^R با محتویات پشته،

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

و نهایتاً

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

برای شناسایی برابری نهایی لازم است.

برای نمونه، دنباله تغییرات پیکربندی برای پذیرش رشته $abba$ به صورت زیر می‌باشد:

$$(q_0,abba,z)\mapsto(q_0,bba,az)\mapsto(q_0,ba,baz)$$
$$\mapsto(q_1,ba,baz)\mapsto(q_1,a,az)\mapsto(q_1,\lambda,z)\mapsto(q_2,z).$$

از ویژگی نامعین‌بودن اتومات فوق برای مکان‌یابی وسط رشته در حرکت سوم استفاده شد. در این مرحله، pda پیکربندی لحظه‌ای (q_0,ba,baz) و همچنین، دو انتخاب برای حرکت بعد در اختیار دارد که یکی، استفاده از $\delta(q_0,b,b)=\{(q_0,bb)\}$ برای انجام حرکت

$$(q_0,ba,baz)\mapsto(q_0,a,bbaz),$$

و دیگری $\delta(q_0,\lambda,b)=\{(q_1,b)\}$ است. فقط حرکت آخر منجر به پذیرش ورودی می‌شود.

تمرین‌ها

۱. pda با کمتر از چهار حالت پیدا کنید که همان زبان pda مثال ۷-۲ را پذیرش کند.

۲. اثبات کنید که pda مثال ۷-۵ هیچ کدام از رشته‌های غیرموجود در $\{ww^R\}$ را پذیرش نمی‌کند.

۳. npda هایی بسازید که زبان‌های منظم زیر را پذیرش کند.

الف) $L_1=L(aaa^*b)$

ب) $L_2=L(aab^*aba^*)$

ج) اجتماع L_1 و L_2.

د) L_1-L_2

۴. npda هایی بسازید که زبان‌های زیر را روی الفبای $\Sigma=\{a,b,c\}$ پذیرش کند.

الف) $L=\{a^nb^{2n}:n\geq 0\}$.

ب) $L=\{wcw^R:w\in\{a,b\}^*\}$.

ج) $L=\{a^nb^mc^{n+m}:n\geq 0,m\geq 0\}$.

د) $L=\{a^nb^{n+m}c^m:n\geq 0,m\geq 1\}$.

ه) $L=\{a^nb^nc^n:n\geq 0\}$.

و) $L=\{a^nb^m:n\leq m\leq 3n\}$.

ز) $L=\{w:n_a(w)=n_b(w)+1\}$.

ح) $L=\{w:n_a(w)=2n_b(w)\}$.

ط) $L=\{w:n_a(w)+n_b(w)=n_c(w)\}$.

ی) $L=\{w:2n_a(w)\leq n_b(w)\leq 3n_a(w)\}$.

ک) $L=\{w:n_a(w)<n_b(w)\}$.

۵. npda ای بسازید که زبان $L=\{a^nb^m:n\geq 0,n\neq m\}$ را پذیرش کند.

۶. npda ای روی $\Sigma=\{a,b,c\}$ پیدا کنید که زبان زیر را پذیرش کند.

$L=\{w_1cw_2:w_1,w_2\in\{a,b\}^*,w_1\neq w_2^R\}$.

۷. npda ای برای الحاق $L(a^*)$ به زبان تمرین ۶، ارائه کنید.

۸. یک npda برای زبان $L=\{ab(ab)^nb(ba)^n:n\geq 0\}$ ارائه کنید.

۹. آیا dfa ای وجود دارد که همان زبان pda

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

با فرض

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

را پذیرش کند.

۱۰. pda

$$M=(\{q_0,q_1,q_2,q_3,q_4,q_5\},\{a,b\},\{0,1,a\},\delta,z,q_0,\{q_5\}),$$

با فرض

$$\delta(q_0,b,z)=\{(q_1,1z)\},$$
$$\delta(q_1,b,1)=\{(q_1,11)\},$$
$$\delta(q_2,a,1)=\{(q_3,\lambda)\},$$
$$\delta(q_3,a,1)=\{(q_4,\lambda)\},$$
$$\delta(q_4,a,z)=\{(q_4,z),(q_5,z)\}.$$

چه زبانی را پذیرش می‌کند؟

npda

$$M=(\{q_0,q_1,q_2\},\{a,b\},\{a,b,z\},\delta,q_0,z,\{q_2\})$$

با انتقالات

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

چه زبانی را پذیرش می‌کند؟

۱۲. اگر در مثال ۷-۴ از $F=\{q_0,q_f\}$ استفاده کنیم، npda مذکور چه زبانی را پذیرش می‌کند؟

۱۳. اگر در تمرین ۱۱ از $F=\{q_0,q_1,q_2\}$ استفاده کنیم، npda مذکور چه زبانی را پذیرش می‌کند؟

۱۴. یک npda با حداکثر دو حالت داخلی پیدا کنید که زبان $L(aa^*ba^*)$ را بپذیرد.

۱۵. فرض کنید که در مثال ۷-۲ به جای مقدار داده شده برای $\delta(q_2,\lambda,0)$ از

$$\delta(q_2,\lambda,0)=\{(q_0,\lambda)\}$$

استفاده کنیم، pda جدید چه زبانی را پذیرش می‌کند؟

۱۶. npda را می‌توان به صورت مدل محدود شده تعریف کرد. یگونه‌ای که در هر حرکت، طول پشته را حداکثر به اندازه یک سمبل افزایش می‌دهد، و بر این اساس می‌توان تعریف ۷-۱ را به گونه‌ای تغییر داد که

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

برای تفسیر رابطه بالا می‌گوییم که، برد δ از مجموعه زوج‌هایی به صورت (q_i,λ) یا (q_i,a) و یا (q_i,ab)، تشکیل می‌شود. نشان دهید که به ازای هر npda به نام M ، یک npda محدود شده به نام $\hat{M}$، وجود دارد بطوریکه $L(M)=L(\hat{M})$.

۱۷. شرط دیگر برای پذیرش زبان تعریف ۷-۲ این است که وقتی به انتهای رشته ورودی می‌رسیم، پشته تهی باشد. به بیان دقیق، اگر

$$N(M)=\left\{w\in\Sigma^*:(q_0,w,z)\vdash_M(p,\lambda,\lambda)\right\},$$

بر این اساس می‌گوییم که npda M ، زبان $N(M)$ را با پشته تهی می‌پذیرد. در رابطه فوق، p عضوی از Q است. نشان دهید که این تعریف برای N(M) دز عمل معادل با تعریف ۷-۲ است؛ به این معنی که به ازای هر npda به نام M، یک npda به نام $\hat{M}$ وجود دارد بطوریکه $L(M)=N(\hat{M})$ و بالعکس.

اتوماتای پشته‌ای و زبان‌های مستقل‌ازمتن

در مثال‌های بخش قبل، مشاهده کردیم که برای برخی زبان‌های مستقل‌ازمتن معروف و آشنا، اتوماتای پشته‌ای معادلی وجود دارد. این امر تصادفی نیست و در واقع باید گفت که یک رابطه کلی بین زبان‌های مستقل‌ازمتن و پذیرنده‌های پشته‌ای نامعین وجود دارد که در قالب دو نتیجه مهم زیر گنجانده شده است. در ادامه نشان خواهیم داد که به ازای هر زبان مستقل‌ازمتن، یک npda وجود دارد که آنرا پذیرش می‌کند، و بالعکس، زبان پذیرفته شده بوسیله هر npda مستقل‌ازمتن است.

اتوماتای پشته‌ای برای زبان‌های مستقل‌ازمتن

ابتدا نشان می‌دهیم که به ازای هر زبان مستقل‌ازمتن یک npda وجود دارد که آنرا پذیرش می‌کند. در اصل، می‌خواهیم npdaای بسازیم که بتواند به طریقی، اشتقاق چپ‌ترین هر رشته‌ای عضو زبان را دنبال کند. برای ساده‌تر کردن بحث، فرض می‌کنیم که گرامر تولید کنندهٔ زبان به فرم گریباخ است.

pda مورد نظر قادر است تا برای نمایش اشتقاق، متغیرها را در بخش سمت راست فرم جمله‌ای پشته خود نگه دارد. در حالیکه سمت چپ، که تماماً از سمبل‌های پایانی تشکیل می‌شود، مشابه ورودی خوانده شده است. ابتدا، متغیر شروع گرامر را روی پشته قرار می‌دهیم. سپس، برای شبیه‌سازی کارکرد قوانین $A\to ax$، باید متغیر A را در بالای پشته و پایانی a را بعنوان سمبلی از رشته ورودی داشته باشیم. متغیر واقع در بالای پشته، حذف شده و با دنباله‌ای از متغیرها به نام x جایگزین می‌شود. به همین ترتیب در مورد δ بحث می‌کنیم. پیش از ارائه استدلال کلی، بیایید نگاهی به یک مثال ساده و مشابه داشته باشیم.

مثال ۷-۶

pdaای بسازید که زبان تولید شده بوسیله گرامر مفروض با قوانین

$$S\to aSbb\mid a$$

را پذیرش کند.

ابتدا با تغییر قوانین به

$$S\to aSA\mid a,$$
$$A\to bB,$$
$$B\to b.$$

گرامر فوق را به فرم نرمال گریباخ تبدیل می‌کنیم.

اتومات نظیر دارای سه حالت $\{q_0,q_1,q_2\}$ خواهد بود، به طوری که حالت شروع با q_0 و حالت پایانی با q_2 مشخص می‌شود . ابتدا توسط

$$\delta(q_0,\lambda,z)=\{(q_1,Sz)\}$$

متغیر شروع S در ته پشته روی z قرار داده می‌شود. برای شبیه‌سازی قانون تولید $S \to aSA$ در pda ، S را از پشته حذف کرده و با خواندن a از ورودی، S را با SA جایگزین می‌کنیم. به همین ترتیب، قانون تولید $S \to a$ باعث می‌شود که pda ، باحذف S ، اقدام به خواندن a کند. بنابراین، این دو قانون در pda با

$$\delta(q_1,a,S)=\{(q_1,SA),(q_1,\lambda)\}$$

جایگزین می‌شوند.

به روشی مشابه، بقیه قوانین را بوسیلهٔ pda ، شبیه‌سازی می‌کنیم.

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

با وقوع z در ته پشته به عنوان سمبل بالای پشته، می‌توان اعلام کرد که کار اشتقاق کامل شده است و pda بوسیله

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

در حالت پایانی خود، قرار می‌گیرد.

از ساخت این مثال می‌توان بعنوان روشی برای دیگر موارد هم استفاده کرد و به این ترتیب، نتیجه کلی زیر را بدست آورد.

قضیه ۷-۱

برای همهٔ زبانهای مستقل‌ازمتن L ، یک npda به نام M وجود دارد بطوریکه

$$L=L(M).$$

اثبات: اگر L یک زبان مستقل‌ازمتن فاقد λ باشد، یک گرامر مستقل‌ازمتن در فرم نرمال گریباخ برای آن وجود دارد. فرض کنید گرامر بصورت $G=(V,T,S,P)$ باشد. سپس npdaای می‌سازیم که اشتقاق‌های چپ‌ترین این گرامر را شبیه‌سازی می‌کند. همانطور که گفتیم، شبیه‌سازی به صورتی انجام خواهد شد که بخش پردازش نشده فرم‌جمله‌ای در پشته قرار گرفته و پیشوند پایانی تمامی فرم‌های جمله‌ای با پیشوند معادل آن در رشته ورودی همخوانی داشته باشد.

در حالت خاص، این npda بصورت

$$M=(\{q_0,q_1,q_f\},T,V\cup\{z\},\delta,q_0,z,\{q_f\}),$$

خواهد برد که درآن، $z\notin V$. توجه داشته باشید که الفبای ورودی M مشابه با مجموعه پایانی‌های گرامر G بوده و بعلاوه، الفبای پشته حاوی مجموعه متغیرهای گرامر است.

تابع انتقال به فرم

$$\delta(q_0,\lambda,z)=\{(q_1,Sz)\},\qquad(۱-۷)$$

است تا پس از اولین حرکت M پشته حاوی متغیر شروع S برای اشتقاق باشد.(سمبل شروع z در پشته سمبلی است که برای شناسایی انتهای اشتقاق و پایان رشته بکار می‌رود.) بعلاوه، مجموعه قوانین انتقال به گونه‌ای است که هرگاه

$$A\to au\quad A\to au$$

در P قرار داشته باشد، خواهیم داشت:

$$(q_1,u)\in\delta(q_1,a,A),\qquad(۲-۷)$$

به این ترتیب، ورودی a خوانده شده، متغیر A از پشته حذف می‌شود و با u جایگزین می‌شود. انتقالات رخ داده باعث می‌شوند تا pda تمامی اشتقاق‌ها را شبیه‌سازی کند. در نهایت،

$$\delta(q_1,\lambda,z)=\{(q_f,z)\},\qquad(۳-۷)$$

برای اینکه M به حالت پایانی برود، استفاده می‌شود.

برای اثبات اینکه M همه $w\in L(G)$ را پذیرش می‌کند، اشتقاق چپ‌ترین جزئی زیر را در نظر بگیرید:

$$S\overset{*}{\Rightarrow}a_1a_2\cdots a_nA_1A_2\cdots A_m$$
$$\Rightarrow a_1a_2\cdots a_nbB_1\cdots B_kA_2\cdots A_m.$$

برای اینکه M بتواند این اشتقاق را شبیه‌سازی کند، لازم است که پشته پس از خواندن $a_1a_2\cdots a_n$، حاوی $A_1A_2\cdots A_m$ باشد. در مرحله بعدی اشتقاق، باید قانون

$$A_1\to bB_1\cdots B_k.$$

در G وجود داشته باشد. اما براساس مراحل ساخت، در اینصورت M قانون انتقالی خواهد داشت که در آن

$$(q_1,B_1\cdots B_k)\in\delta(q_1,b,A_1),$$

بطوریکه پشته پس از خواندن $a_1a_2\cdots a_nb$ حاوی $B_1\cdots B_kA_2\cdots A_m$ می‌باشد، نه آنچه که ما نیاز داریم.

با یک استدلال استقرایی ساده روی تعداد مراحل اشتقاق می‌توان نشان داد که اگر

$$S\overset{*}{\Rightarrow}w,$$

آنگاه

$$(q_1,w,Sz)\overset{*}{\mapsto}(q_1,\lambda,z).$$

اینک، با استفاده از (۷-۱) و (۷-۳) داریم:

$$(q_0,w,z)\vdash(q_1,w,Sz)\vdash(q_1,\lambda,z)\vdash(q_f,\lambda,z),$$

بطوریکه $L(G)\subseteq L(M)$.

برای اثبات $L(M)\subseteq L(G)$، فرض کنید که $w\in L(M)$ باشد. آنگاه براساس تعریف داریم

$$(q_0,w,z)\vdash(q_f,\lambda,u).$$

بدلیل آنکه فقط یک راه برای انتقال از q_0 به q_1 و یک راه برای انتقال از q_1 به q_f وجود دارد. بنابراین، باید داشته باشیم:

$$(q_1,w,Sz)\vdash(q_1,\lambda,z).$$

اینک، اگر $w=a_1a_2a_3\cdots a_n$ در نظر گرفته شود، آنگاه در اولین مرحله از

$$(q_1,a_1a_2a_3\cdots a_n,Sz)\vdash(q_1,\lambda,z) \qquad (۴-۷)$$

باید قانونی به فرم (۷-۲) وجود داشته باشد تا

$$(q_1,a_1a_2a_3\cdots a_n,Sz)\vdash(q_1,a_2a_3\cdots a_n,u_1z).$$

اما در اینصورت باید قانون $S\to a_1u_1$ در گرامر وجود داشته باشد، بطوریکه

$$S\Rightarrow a_1u_1.$$

با تکرار قانون فوق و فرض $u_1=Au_2$، داریم :

$$(q_1,a_2a_3\cdots a_n,Au_2z)\vdash(q_1,a_3\cdots a_n,u_3u_2z),$$

که به طور ضمنی به معنای وجود $A\to a_2u_3$ در گرامر و همچنین

$$S\Rightarrow a_1a_2u_3u_2.$$

خواهد بود. به این ترتیب می‌بینیم که در تمامی طول زمان عملیات، محتویات پشته (به استثنای z) مشابه بخش تطابق نیافته فرم جمله‌ای است. بنابراین، با استفاده از(۷-۴) همچنین داریم:

$$S\Rightarrow a_1a_2\cdots a_n$$

در نتیجه، اثبات $L(M)\subseteq L(G)$ برای زبان‌های فاقد λ کامل می‌شود.

چنانچه $\lambda\in L$، انتقال

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

را به npda ساخته شده اضافه می‌کنیم تا رشته تهی هم پذیرش شود. ■

مثال ۷-۱۱

گرامر زیر را در نظر بگیرید.

$$S\to aA,$$
$$A\to aABC\mid bB\mid a,$$
$$B\to b,$$
$$C\to c.$$

از آنجایی که این گرامر اکنون در فرم نرمال گریباخ است، می‌توان به راحتی از روش ساخت موجود در قضیه قبل استفاده کرد. علاوه بر قوانین

$$\delta(q_0,\lambda,z)=\{(q_1,Sz)\}$$

و

$$\delta(q_1,\lambda,z)=\{(q_f,z)\},$$

همچنین pda از قوانین انتقال

$$\delta(q_1,a,S)=\{(q_1,A)\},$$
$$\delta(q_1,a,A)=\{(q_1,ABC),(q_1,\lambda)\},$$
$$\delta(q_1,b,A)=\{(q_1,B)\},$$
$$\delta(q_1,b,B)=\{(q_1,\lambda)\},$$
$$\delta(q_1,c,C)=\{(q_1,\lambda)\}.$$

برخوردار است.

برای نمونه، دنباله حرکات انجام شده بوسیله M جهت پردازش $aaabc$ به صورت زیر می‌باشد:

$$\begin{aligned}(q_0,aaabc,z)&\vdash(q_1,aaabc,Sz)\\&\vdash(q_1,aabc,Az)\\&\vdash(q_1,abc,ABCz)\\&\vdash(q_1,bc,BCz)\\&\vdash(q_1,c,Cz)\\&\vdash(q_1,\lambda,z)\\&\vdash(q_f,\lambda,z)\end{aligned}$$

که متناظر با اشتقاق

$$S\Rightarrow aA\Rightarrow aaABC\Rightarrow aaaBC\Rightarrow aaabC\Rightarrow aaabc.$$

است.

■

برای ساده کردن استدلال‌ها در اثبات قضیه ۷-۱، گرامر را در فرم نرمال گریباخ فرض کردیم. اما

همواره لزومی به انجام اینکار نبوده و می‌توان با یک گرامر مستقل‌ازمتن کلی نیز به ساختاری مشابه، اما کمی زودتر و البته پیچیده‌تر، دست یافت. بعنوان مثال، برای قوانین به فرم

$$A \rightarrow Bx,$$

A را از پشته حذف کرده و آنرا با Bx جایگزین می‌کنیم، اما از هیچ سمبل ورودی استفاده نمی‌کنیم. برای قوانین به فرم

$$A \rightarrow abCx,$$

ابتدا باید ab ورودی را با رشته مشابهی در پشته تطابق داده و سپس A را با Cx جایگزین کنیم. جزئیات مراحل ساخت npda و اثبات مربوط به این کار را بعنوان تمرین در انتهای همین بخش آورده‌ایم.

گرامرهای مستقل‌ازمتن برای اتوماتای پشته‌ای

عکس قضیه ۷-۱ نیز برقرار است. برای ساخت حالت عکس، فقط کافی است که فرآیند قضیه ۷-۱ را معکوس کنید تا گرامر بتواند حرکات pda را شبیه‌سازی کند. به این معنا که محتوای پشته باید در بخش متغیر فرم جمله‌ای منعکس شده و ورودی پردازش شده در پیشوند پایانی فرم جمله‌ای قرار گیرد. انجام اینکار با کمی دقت به جزئیات به راحتی امکان‌پذیر است.

به بیان ساده‌تر، فرض کنید که npda مورد نظر دارای شرایط زیر باشد:

۱. فقط یک حالت پایانی q_f داشته باشد که وارد آن می‌شویم اگر و تنها اگر پشته خالی باشد؛

۲. با ضابطه $a \in \Sigma \cup \{\lambda\}$، $\delta(q_i, a, A) = \{c_1, c_2, \ldots, c_n\}$ باشند که در آن،

$$c_i = (q_j, \lambda), \qquad (۷-۵)$$

یا

$$c_i = (q_j, BC). \qquad (۷-۶)$$

به این معنا که هر یک از حرکات پشته، محتوای آنرا به اندازه یک سمبل یا متغیر افزایش یا کاهش می‌دهد.

هرچند این شرایط بسیار سخت‌گیرانه به نظر می‌رسد، در واقع چنین نبوده و می‌توان اثبات کرد که به ازای هر npda، یک npda معادل با ویژگی‌های ۱ و ۲ وجود دارد. با نمونه‌هایی از این معادل بودن در تمرین‌های ۱۶و ۱۷ بخش ۷-۱ آشنا شدید. گرچه در اینجا آنرا کمی بیشتر بسط و شرح می‌دهیم، دوباره به عنوان تمرین ۱۶ در انتهای همین بخش مطرح خواهیم کرد. با فرض اثبات معادل بودن مذکور، حال یک گرامر مستقل‌ازمتن برای زبان پذیرفته شده بوسیله npda می‌سازیم.

همانطور که قبلاً هم گفته شد، فرم جمله‌ای باید محتوای پشته را نمایش دهد. اما پیکربندی npda

شامل یک حالت داخلی است که باید در فرم جمله‌ای ذکر شود. همین امر، کار را تا حدودی مشکل کرده و پیچیدگی روش ساختی که در اینجا ارائه می‌دهیم، کار را واقعاً طاقت‌فرسا می‌کند.

فعلاً فرض کنید که گرامر مورد نظر ما متغیرهایی به فرم $(q_i A q_j)$ و قوانینی به فرم

$$(q_i A q_j) \Rightarrow v,$$

دارد. اگر و تنها اگر npda، در حین خواندن v و انتقال از حالت q_0 به q_j، A را از پشته پاک کند. در اینجا، منظور ما از "پاک‌کردن" آن است که A و اثرات آن (یعنی تمامی رشته‌های متوالی که جایگزین آن می‌شود) از رشته حذف شده و نشانه‌ای را که در ابتدا زیر A قرارداشته، الان روی پشته می‌باشد. اگر بتوان چنین گرامری را پیدا کرد، و اگر $(q_0 z p_f)$ را بعنوان متغیر شروع آن انتخاب کنیم، آنگاه

$$(q_0 z p_f) \Rightarrow w$$

اگر و تنها اگر npda در حین خواندن w و انتقال از حالت q_0 به q_f، اقدام به حذف z (و ایجاد یک پشته تهی) نماید. اما npda به این ترتیب اقدام به پذیرش w می‌کند. بنابراین، زبان تولید شده بوسیله گرامر، با زبان پذیرفته شده بوسیله npda یکسان خواهد بود.

برای ساخت گرامری با این شرایط، انواع مختلف انتقالات npda را بررسی می‌کنیم. بدلیل آنکه (۷-۵) مستلزم حذف فوری A می‌باشد، قانون تولیدی به فرم

$$(q_i A q_j) \rightarrow a$$

در گرامر وجود خواهد داشت. قوانین به فرم (۷-۶)، مجموعه قوانین به فرم

$$(q_i A q_k) \rightarrow a(q_j B q_l)(q_l C q_k),$$

را می‌سازند، که در آن، q_k و q_l تمامی مقادیر ممکن در Q را اختیار می‌کنند. دلیل این‌کار هم آن است که برای پاک کردن A، ابتدا باید آنرا با BC جایگزین کرده، درحالیکه یک a را می‌خوانیم و از حالت q_i به q_j می‌رویم. در نتیجه، از حالت q_j به q_l رفته و B را حذف می‌کنیم. سپس از q_l به q_k رفته و C را حذف می‌کنیم.

از آنجایی‌که در مرحله آخر هنوز هم برخی حالت‌های q_l در حین پاک‌کردن B، از q_j قابل دسترسی نمی‌باشند، ممکن است چنین به نظر برسد که در افزودن برخی متغیرها زیاده‌روی کرده‌ایم. هرچند این امر واقعیت دارد، اما تأثیری بر گرامر حاصل نمی‌گذارد. اشتقاق‌های $(q_j B q_l)$ متغیرهای غیرمفید بوده و تأثیری بر زبان پذیرفته شده توسط گرامر ندارند.

در نهایت، $(q_0 z q_f)$ را بعنوان متغیر شروع در نظر می‌گیریم، که در آن q_f تنها حالت پایانی npda است.

مثال ۷-۸

npda با انتقالات زیر را در نظر بگیرید.

$$\delta(q_0,a,z)=\{(q_0,Az)\},$$

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

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

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

با فرض q_0 بعنوان حالت شروع و q_2 بعنوان حالت پایانی، npda فقط شرط ۱ قبل را برآورده می‌کند. برای آنکه شرط ۲ برآورده کند، حالت جدید q_3 و یک مرحله میانی را معرفی می‌کنیم تا به این ترتیب، ابتدا A از پشته حذف شده و سپس با حرکت بعدی جایگزین شود. مجموعه جدید قوانین انتقال به صورت زیر خواهد بود.

$$\delta(q_0,a,z)=\{(q_0,Az)\},$$
$$\delta(q_3,\lambda,z)=\{(q_0,Az)\},$$
$$\delta(q_0,a,A)=\{(q_3,\lambda)\},$$
$$\delta(q_0,b,A)=\{(q_1,\lambda)\},$$
$$\delta(q_1,\lambda,z)=\{(q_2,\lambda\}.$$

سه انتقال آخر به فرم (۵-۷) بوده و به همین دلیل، قوانین متناظر

$$(q_0Aq_3)\to a,\quad (q_0Aq_1)\to b,\quad (q_1zq_2)\to\lambda$$

را بوجود می‌آورند. از دو انتقال اول، مجموعه قوانین زیر بدست می‌آید.

$$\begin{aligned}(q_0zq_0)\to{}&a(q_0Aq_0)(q_0zq_0)\mid a(q_0Aq_1)(q_1zq_0)\mid\\&a(q_0Aq_2)(q_2zq_0)\mid a(q_0Aq_3)(q_3zq_0),\\(q_0zq_1)\to{}&a(q_0Aq_0)(q_0zq_1)\mid a(q_0Aq_1)(q_1zq_1)\mid\\&a(q_0Aq_2)(q_2zq_1)\mid a(q_0Aq_3)(q_3zq_1),\\(q_0zq_2)\to{}&a(q_0Aq_0)(q_0zq_2)\mid a(q_0Aq_1)(q_1zq_2)\mid\\&a(q_0Aq_2)(q_2zq_2)\mid a(q_0Aq_3)(q_3zq_2),\\(q_0zq_3)\to{}&a(q_0Aq_0)(q_0zq_3)\mid a(q_0Aq_1)(q_1zq_3)\mid\\&a(q_0Aq_2)(q_2zq_3)\mid a(q_0Aq_3)(q_3zq_3),\end{aligned}$$

$$(q_3zq_0)\to(q_0Aq_0)(q_0zq_0)\mid(q_0Aq_1)(q_1zq_0)\mid(q_0Aq_2)(q_2zq_0)\mid(q_0Aq_3)(q_3zq_0),$$

$$(q_3zq_1)\to(q_0Aq_0)(q_0zq_1)\mid(q_0Aq_1)(q_1zq_1)\mid(q_0Aq_2)(q_2zq_1)\mid(q_0Aq_3)(q_3zq_1),$$

$$(q_3zq_2)\to(q_0Aq_0)(q_0zq_2)\mid(q_0Aq_1)(q_1zq_2)\mid(q_0Aq_2)(q_2zq_2)\mid(q_0Aq_3)(q_3zq_2),$$

$$(q_3zq_3)\to(q_0Aq_0)(q_0zq_3)\mid(q_0Aq_1)(q_1zq_3)\mid(q_0Aq_2)(q_2zq_3)\mid(q_0Aq_3)(q_3zq_3).$$

ظاهر بسیار پیچیده قوانین زیر را می‌توان بصورت زیر تا حدودی ساده کرد. اگر متغیری در طرف چپ هیچ کدام از قوانین رخ ندهد، حتماً غیرمفید است. بر این اساس فوراً می‌توان (q_0Aq_0) و (q_0Aq_2) را حذف کرد. همچنین، با نگاهی به گراف انتقال npda اصلاح‌شده، مشاهده می‌کنیم که هیچ مسیری از q_1 به q_0، از q_1 به q_1، از q_1 به q_3 و همچنین از q_2 به q_2 وجود ندارد و بنابراین، متغیرهای مربوطه هم غیرمفید هستند. پس از حذف تمامی قوانین غیرمفید مشابه به این مورد، گرامر بسیار کوچک‌تر شده و به

$$\begin{aligned}&(q_0Aq_3)\to a,\\&(q_0Aq_1)\to b,\\&(q_1zq_2)\to\lambda,\\&(q_0zq_0)\to a(q_0Aq_3)(q_3zq_0),\\&(q_0zq_1)\to a(q_0Aq_3)(q_3zq_1),\\&(q_0zq_2)\to a(q_0Aq_1)(q_1zq_2)\mid a(q_0Aq_3)(q_3zq_2),\\&(q_0zq_3)\to a(q_0Aq_3)(q_3zq_3),\\&(q_3zq_0)\to(q_0Aq_3)(q_3zq_0),\\&(q_3zq_1)\to(q_0Aq_3)(q_3zq_1),\\&(q_3zq_2)\to(q_0Aq_1)(q_1zq_2)\mid(q_0Aq_3)(q_3zq_2),\\&(q_3zq_3)\to(q_0Aq_3)(q_3zq_3),\end{aligned}$$

با متغیر شروع (q_0zq_2)، تبدیل خواهد شد.

مثال ۵-۹

رشته مفروض $w=aab$ توسط pda مثال ۷-۸ با پیکربندی‌های متوالی

$$\begin{aligned}(q_0,aab,z)&\mapsto(q_0,ab,Az)\\&\mapsto(q_3,b,z)\\&\mapsto(q_0,b,Az)\\&\mapsto(q_1,\lambda,z)\\&\mapsto(q_2,\lambda,\lambda)\end{aligned}$$

پذیرفته می‌شود. اشتقاق متناظر با G

$$(q_0zq_2)\Rightarrow a(q_0Aq_3)(q_3zq_2)$$
$$\Rightarrow aa(q_3zq_2)$$
$$\Rightarrow aa(q_0Aq_1)(q_1zq_2)$$
$$\Rightarrow aab(q_1zq_2)$$
$$\Rightarrow aab$$

می‌باشد. توجه به تناظر بین پیکربندی‌های لحظه‌ای متوالی npda و فرم‌های جمله‌ای اشتقاق، درک مراحل اثبات قضیه زیر را به مراتب راحت‌تر می‌کند. اولین q_i در چپ‌ترین متغیر هریک از فرم‌های جمله‌ای، همان حالت جاری pda است. در حالیکه دنباله سمبل‌های میانی، همان محتوای پشته می‌باشد.

■

هرچند روش ساخت فوق باعث پیچیدگی گرامر می‌شود، می‌توان از آن برای ساخت تمامی pdaهایی استفاده کرد که قوانین تولید آنها در شرایط داده شده صدق می‌کند. این ویژگی اساس اثبات یک نتیجه کلی را تکمیل می‌کند.

قضیه ۷-۲

اگر در یک npda مفروض M داشته باشیم $L=L(M)$، آنگاه L یک زبان مستقل‌ازمتن است.

اثبات: فرض کنید که $M=(Q,\Sigma,\Gamma,\delta,q_0,z,\{q_f\})$ در شرایط ۱ و ۲ قبل، صدق می‌کند. با استفاده از روش ساخت قبلی، گرامر $G=(V,T,S,P)$ با فرض $T=\Sigma$ و V را با اعضایی به فرم (q_icq_j) ایجاد می‌کنیم. باید نشان دهیم گرامری که به این ترتیب بدست می‌آید چنان است که به ازای تمامی $A\in\Gamma$، $q_i,q_j\in Q$، $X\in\Gamma^*$ و $u,v\in\Sigma^*$

$$(q_i,uv,AX)\overset{*}{\mapsto}(q_j,v,X) \qquad (۷-۷)$$

به طور ضمنی به معنای

$$(q_iAq_j)\overset{*}{\Rightarrow}u,$$

می‌باشد و برعکس.

در مرحله اول نشان می‌دهیم که هرگاه npda بتواند متغیر A و اثرات آنرا از پشته، در حین خواندن u و انتقال از حالت q_i به q_j حذف کند، آنگاه متغیر (q_iAq_j) قادر به اشتقاق u می‌باشد. بدلیل آنکه با ساختن گرامر به خودی خود این نتیجه بدست می‌آید، به راحتی می‌توان به درستی عبارت بالا پی برد. فقط باید با استقراء روی تعداد حرکات، این نتیجه را کامل کرد.

برای حالت عکس، کافی است یکی از مراحل اشتقاق را در نظر بگیرید که در آن،

$$(q_iAq_k)\Rightarrow a(q_jBq_l)(q_lCq_k).$$

با استفاده از انتقال نظیر برای npda داریم:

$$\delta(q_i,a,A)=\{(q_j,BC),\ldots\}. \qquad (۸-۷)$$

بنابراین، می‌توان A را از پشته حذف کرده، BC را جایگزین آن کرد و در حین انتقال واحد کنترل از حالت q_i به q_j، اقدام به خواندن a نمود. همچنین اگر

$$(q_iAq_j)\Rightarrow a, \qquad (۹-۷)$$

آنگاه باید یک انتقال معادل به فرم

$$\delta(q_i,a,A)=\{(q_j,\lambda)\} \qquad (۱۰-۷)$$

وجود داشته باشد تا بتوان به کمک آن A را از پشته خارج کرد. در این صورت می‌بینیم که فرم‌های جمله‌ای اشتقاق شده از (q_iAq_j) دنباله‌ای از پیکربندی‌های ممکن pda را تعریف می‌کند که بواسطه آنها (۷-۷) انجام می‌شود.

توجه داشته باشید که $(q_iAq_j)\Rightarrow a(q_jBq_l)(q_lCq_k)$ به ازای $(q_jBq_l)(q_lCq_k)$ امکانپذیر است که در آنها، هیچ انتقال متناظری به فرم (۸-۷) یا (۱۰-۷) وجود نداشته باشد. اما در این صورت، حداقل یکی از متغیرهای واقع در طرف راست غیرمفید خواهد بود. استدلال فوق در مورد همهٔ فرم‌های جمله‌ای که به یک رشته پایانی ختم می‌شوند، برقرار است.

اکنون اگر نتیجه را بروی

$$(q_0,w,z)\overset{*}{\mapsto}(q_f,\lambda,\lambda),$$

بکار بریم می‌بینیم که عبارت بالا تنها در صورتی وجود خواهد داشت که

$$(q_0zq_f)\overset{*}{\Rightarrow}w.$$

در نتیجه $L(M)=L(G)$ است. ■

تمرین‌ها

۱. نشان دهید که pda ساخته‌شده در مثال ۷-۶ رشته $aaabbbb$ را پذیرش می‌کند که عضو زبان تولید شده بوسیله گرامر مذکور می‌باشد.

۲. اثبات کنید که pda مثال ۷-۶ زبان $L=\{a^{n+1}b^{2n}:n\geq 0\}$ را پذیرش می‌کند.

۳. npdaای بسازید که زبان تولید شده بوسیله گرامر زیر را پذیرش می‌کند.

$$S\rightarrow aSbb\mid aab.$$

۴. npdaای بسازید که زبان تولید شده بوسیله گرامر $S\rightarrow aSSS\mid ab$ را پذیرش می‌کند.

۵. npdaای متناظر با گرامر زیر را بسازید.

$$S \to aABB \mid a\Lambda A,$$
$$A \to aBB \mid a,$$
$$B \to bBB \mid A.$$

۶. npdaای بسازید که زبان تولید شده بوسیله گرامر $G=(\{S,A\},\{a,b\},S,P)$ با قوانین $S \to AA \mid a$ و $A \to SA \mid b$ را پذیرش کند.

۷. نشان دهید که قضایای ۷-۱ و ۷-۲ این نتیجه‌گیری را ارائه می‌کنند: به ازای هر npda به نام M، یک npda، به نام $\hat{M}$ حداکثر سه حالتی وجود دارد، بطوریکه $L(M)=L(\hat{M})$.

۸. نشان دهید چگونه تعداد حالات $\hat{M}$ در تمرین قبل می‌تواند به دو حالت کاهش یابد.

۹. یک npda دو حالتی برای زبان $L=\{a^n b^{n+1} : n \geq 0\}$ پیدا کنید.

۱۰. یک npda دو حالتی پیدا کنید که $L=\{a^n b^{2n} : n \geq 1\}$ را پذیرش کند.

۱۱. نشان دهید که npda مثال ۷-۸ زبان $L(aa^*b)$ را پذیرش می کند.

۱۲. نشان دهید که گرامر مثال ۷-۸ زبان $L(aa^*b)$ را تولید می‌کند.

۱۳. در مثال ۷-۸ نشان دهید که متغیر (q_0zq_1) غیرمفید است.

۱۴. با اقتباس از روش ساخت موجود در قضیه ۷-۱ یک npda برای زبان مثال ۷-۵ بخش ۷-۱ پیدا کنید.

۱۵. یک گرامر مستقل‌ازمتن پیدا کنید که زبان پذیرفته شده بوسیله npda مفروض $M=(\{q_0,q_1\},\{a,b\},\{A,z\},\delta,q_0,z,\{q_1\})$ با انتقالات

$$\delta(q_0,a,z)=\{(q_0,\Lambda z)\},$$
$$\delta(q_0,b,A)=\{(q_0,\Lambda\Lambda)\},$$
$$\delta(q_0,a,A)=\{(q_1,\lambda)\}.$$

را تولید کند.

۱۶. نشان دهید که به ازای هر npda، یک npda معادل وجود دارد که شرایط ۱ و ۲ موجود در ابتدای قضیه ۷-۲ برای آن برقرار است.

۱۷. جزئیات کامل اثبات قضیه ۷-۲ را بنویسید.

۱۸. روش ساختی را ارائه دهید که بتوان به کمک آن در اثبات قضیه ۷-۱ از یک گرامر مستقل‌ازمتن دلخواه استفاده نمود.

۱۹. آیا در گرامر مثال ۷-۸ هنوز متغیرهای بی‌فایده‌ای وجود دارد؟

۷-۳ اتوماتای پشته‌ای معین و زبان‌های مستقل‌ازمتن معین

یک پذیرنده پشته‌ای معین(dpda) اتومات پشته‌ایی است که انتخابی برای حرکت خود ندارد. برای تعریف dpda، تعریف ۷-۱ را اصلاح می‌کنیم.

تعریف ۷-۲ اتومات پشته‌ای $M=(Q,\Sigma,\Gamma,\delta,q_0,z,F)$ معین گفته می‌شود اگر هم در تعریف (۷-۱) صدق کند و همچنین دارای محدودیت‌هایی به این شرح باشد که به ازای هر $b \in \Gamma$، $a \in \Sigma \cup \{\lambda\}$ و $q \in Q$،

۱. $\delta(q,a,b)$ حداکثر یک عضو داشته باشد،

۲. اگر $\delta(q,\lambda,b)$ تهی نباشد، آنگاه $\delta(q,c,b)$ باید به ازای هر $c \in \Sigma$ تهی باشد.

اولین شرط فوق صرفاً مستلزم آن است که به ازای هر سمبل ورودی مفروض و هر عنصر بالای پشته، حداکثر یک حرکت قابل انجام باشد. براساس شرط دوم، چنانچه در یکی از پیکربندی‌های مفروض به یک انتقال λ برخورد کنیم، آنگاه هیچ حرکتی برای جلو بردن و مصرف ورودی امکان‌پذیر نمی‌باشد.

تفاوت بین این تعریف و تعریف متناظر با آن در مورد اتومات متناهی معین بسیار جالب است. اولاً، چون می‌خواهیم انتقال‌های λ را حفظ کنیم، دامنه تابع انتقال همانند تعریف ۷-۱ است نه $Q \times \Sigma \times \Gamma$. بدلیل آنکه بخش بالای پشته در تعیین حرکت بعدی نقش دارد، وجود انتقال‌های λ به معنای ضمنی نامعین بودن نیست. ثانیاً، برخی انتقالات dpda ممکن است به مجموعه تهی منجر شده و در نتیجه تعریف‌نشده باشند. در اینصورت، باید منتظر برخورد با پیکربندی‌های مرده باشیم. اما این امر تأثیری بر تعریف فوق نمی‌گذارد. باید بدانیم که، تنها در صورتی می‌توان با اطمینان کامل در مورد معین‌بودن اتومات صحبت کرد که همواره حداکثر فقط یک حرکت امکان‌پذیر باشد.

تعریف ۷-۳ زبان L یک زبان **مستقل‌ازمتن معین** نامیده می‌شود اگر و تنها اگر یک dpda به نام M وجود داشته باشد که $L=L(M)$ است.

مثال ۷-۱۱

زبان

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

یک زبان مستقل‌ازمتن معین است. pda ی $M=(\{q_0,q_1,q_2\},\{a,b\},\{0,1\},\delta,q_0,0,\{q_0\})$ با انتقالات

$$\delta(q_0,a,0)=\{(q_1,10)\},$$

$$\delta(q_1,a,1)=\{(q_1,11)\},$$

$$\delta(q_1,b,1)=\{(q_2,\lambda)\},$$

$$\delta(q_2,b,1)=\{(q_2,\lambda)\},$$

$$\delta(q_2,\lambda,0)=\{(q_0,\lambda)\},$$

زبان مورد نظر را می‌پذیرد. همچنین در شرایط تعریف ۷-۴ صدق کرده و بنابراین، معین است.

■

اینک به مثال ۷-۵ نگاه کنید. npda مذکور معین نمی‌باشد، چون

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

و

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

در شرط ۲ تعریف ۷-۳ صدق نمی‌کند. البته این امر بطور ضمنی به معنای نامعین بودن زبان $\{ww^R\}$ نمی‌باشد، چون ممکن است یک dpda هم‌ارز وجود داشته باشد. اما از قبل می‌دانیم که این زبان حتماً معین نمی‌باشد. بر این اساس و مثال بعد می‌گوییم که بالعکس اتوماتای متناهی، هیچ هم‌ارزی بین اتوماتای پشته‌ای معین و نامعین وجود ندارد. لازم به ذکر است که برخی زبان‌های مستقل‌ازمتن معین نمی‌باشند.

مثال ۷-۱۱

فرض کنید

$$L_1=\{a^nb^n : n\geq 0\}$$

و

$$L_2=\{a^nb^{2n} : n\geq 0\}.$$

با تغییر ساده‌ای در استدلال مستقل‌ازمتن بودن زبان L_1، می‌توان مستقل‌ازمتن بودن L_2 را نیز اثبات کرد. زبان

$$L=L_1\cup L_2$$

هم مستقل‌ازمتن است. هرچند این نتیجه طی یکی از قضایای فصل بعد اثبات شده است، فعلاً با ذکر دلایلی آنرا توجیه می‌کنیم. فرض کنید که $G_1=(V_1,T,S_1,P_1)$ و $G_2=(V_2,T,S_2,P_2)$ گرامرهای مستقل‌ازمتن باشند، بطوریکه $L_1=L(G_1)$ و $L_2=L(G_2)$. اگر V_1 و V_2 را جدا از هم و $S\notin V_1\cup V_2$ را فرض کنیم، آنگاه با ترکیب این‌دو، گرامر $G=(V_1\cup V_2\cup\{S\},T,S,P)$ بوجود می‌آید که در آن،

$$P=P_1\cup P_2\cup\{S\rightarrow S_1\mid S_2\},$$

منجر به تولید $L_1\cup L_2$ می‌شود. جزئیات استدلال مربوطه در فصل ۸ ارائه شده است. اگر فعلاً آنرا بدون استدلال قبول کنیم، می‌بینیم که L مستقل‌ازمتن است. اما L زبان مستقل‌ازمتن معین نمی‌باشد. این نتیجه منطقی به نظر می‌رسد، چون pda باید یک یا دو b را با هریک از a ها تطابق داده و هرگاه که رشته ورودی عضو L_1 یا L_2 باشد، آنرا بعنوان اولین انتخاب مطرح کند. اطلاعات موجود در ابتدای رشته چنان نیست که بتوان به کمک آن انتخاب خود را قطعی کرد. البته، این نوع استدلال براساس الگوریتم خاص ذهنی خود ما استوار است و به این دلیل، هرچند مبتکرانه به نظر می‌رسد، کمکی به اثبات نمی‌کند. همواره باید مراقب روش‌های کاملاً جدید و متفاوتی باشیم که انتخاب اولیه ما را نقض می‌کند. اما خوشبختانه در این مورد با چنین مشکلی مواجه نیستیم و بنابراین، L واقعاً نامعین است. برای اطمینان ادعای زیر را مطرح و اثبات می‌کنیم. اگر L یک زبان مستقل‌ازمتن معین باشد، آنگاه

$$\hat{L}=L\cup\{a^nb^nc^n : n\geq 0\}$$

یک زبان مستقل‌ازمتن خواهد بود. سپس، با داشتن dpda‌ی به نام M برای L، ضمن ساخت یک npda به نام $\hat{M}$ برای $\hat{L}$ ادعای فوق را اثبات می‌کنیم.

در واقع باید بخشی مشابه واحد کنترل M ایجاد نمود تا انتقالاتی که براساس سمبل ورودی b در آن انجام می‌شوند، با سمبل‌های مشابه با ورودی c جایگزین شوند. ورود به این بخش جدید از واحد کنترل پس از خوانده شدن a^nb^n توسط M امکان‌پذیر خواهد بود. به همین دلیل، واکنش بخش دوم در برابر c^n درست. مشابه نحوه واکنش بخش اول در برابر b^n خواهد بود. بنابراین، اگر فرآیندی a^nb^{2n} را شناسایی کند، $a^nb^nc^n$ را نیز پذیرش می‌کند. شکل ۷-۴ تصویری از ساخت ماشین مربوطه را نمایش می‌دهد. استدلال دقیق به این صورت است.

$M=(Q,\Sigma,\Gamma,\delta,q_0,z,F)$ را با فرض

$$Q=\{q_0,q_1,\ldots,q_n\}$$

درنظر بگیرید.

سپس $\hat{M}=(\hat{Q},\Sigma,\Gamma,\delta\cup\hat{\delta},q_0,z,\hat{F})$ را با فرضیات

$$\hat{Q}=Q\cup\{\hat{q}_0,\hat{q}_1,\ldots,\hat{q}_n\}$$

$$\hat{F}=F\cup\{\hat{q}_i : q_i\in F\},$$

در نظر گرفته و با اضافه کردن

$$\hat{\delta}(q_f,\lambda,s)=\{(\hat{q}_f,s)\},$$

برای همهٔ $q_f\in F$، $s\in\Gamma$ و

$$\hat{\delta}(\hat{q}_i,c,s)=\{(\hat{q}_j,u)\},$$

برای همه

$$\delta(q_i,b,s)=\{(q_j,u)\},$$

$\hat{\delta}$، $q_i\in Q, s\in\Gamma, u\in\Gamma^*$ را از δ بسازید. برای آنکه M بتواند a^nb^n را پذیرش کند، باید با فرض $q_i\in F$ داشته باشیم

$$(q_0,a^nb^n,z)\overset{*}{\vdash}_M(q_i,\lambda,u).$$

به دلیل معین‌بودن M، باید رابطه

$$(q_0,a^nb^{2n},z)\overset{*}{\vdash}_M(q_i,b^n,u),$$

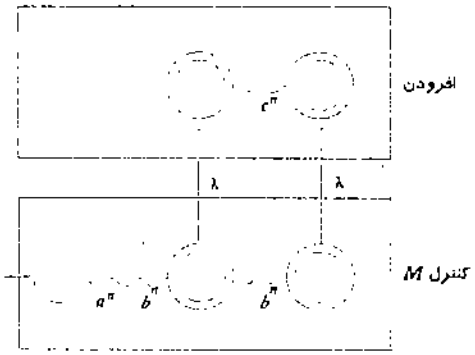
برقرار برده و بنابراین برای پذیرش a^nb^{2n}، باید به ازای برخی از $q_j\in F$، رابطه

$$(q_i,b^n,u)\overset{*}{\vdash}_M(q_j,\lambda,u_1),$$

برقرار باشد. که در اینصورت، براساس ساختار

$$(\hat{q}_i,c^n,u)\overset{*}{\vdash}_M(\hat{q}_j,\lambda,u_1),$$

$a^nb^nc^n$ توسط $\hat{M}$ پذیرش می‌شود. اکنون فقط باید نشان دهیم که هیچکدام از رشته‌ها به غیر از رشته‌های موجود در $\hat{L}$ بوسیله $\hat{M}$ پذیرش نمی‌شوند (نمونه‌های مشابه این مورد در چندین تمرین انتهای این بخش آمده است). در نتیجه $\hat{L}=L(\hat{M})$ بوده و بنابراین $\hat{L}$ مستقل‌ازمتن است. اما در فصل بعد طی مثال ۸-۱ نشان خواهیم داد که $\hat{L}$ مستقل‌ازمتن نیست. بنابراین، فرض اولیه مستقل‌ازمتن و معین‌بودن زبان L حتماً نادرست می‌باشد.



شکل ۷-٤

تمرین‌ها

نشان دهید که $L=\{a^nb^{2n}:n\geq 0\}$ یک زبان مستقل‌ازمتن معین است.

نشان دهید که زبان $L=\{a^nb^m:m\geq n+2\}$ معین است.

۳. آیا زبان $L=\{a^nb^n:n\geq 1\}\cup\{b\}$ معین است؟

٤. آیا زبان $L=\{a^nb^n:n\geq 1\}\cup\{a\}$ معین است؟

۵. نشان دهید که اتومات پشته‌ای مثال ۷-۴ معین نیست، اما زبان این مثال در هر صورت معین است.

۶. برای زبان L تمرین ۱، نشان دهید که L^* یک زبان مستقل‌ازمتن معین است.

۷. با ذکر دلایل کافی توضیح دهید که چرا زبان زیر معین نیست؟

$$L=\{a^nb^mc^k:n=m \text{ یا } m=k\}.$$

۸. آیا زبان $L=\{a^nb^m:n=m \text{ یا } n=m+2\}$ معین است؟

۹. آیا زبان $L=\{wcw^R:w\in\{a,b\}^*\}$ معین است؟

۱۰. هر چند زبان تمرین ۹ معین است، یکی از زبان‌های بسیار نزدیک به آن یعنی $L=\{ww^R:w\in\{a,b\}^*\}$، نامعین می‌باشد. استدلالی ارائه دهید.

۱۱. نشان دهید که $L=\{w\in\{a,b\}^*:n_a(w)\neq n_b(w)\}$ یک زبان مستقل‌ازمتن معین است.

۱۲. نشان دهید که $\hat{M}$ در مثال ۷-۱۱، برای $k\neq n$، $a^nb^nc^n$ را پذیرش نمی‌کند.

۱۳. نشان دهید که $\hat{M}$ در مثال ۷-۱۱ هیچیک از رشته‌های غیرموجود در $L(a^*b^*c^*)$ را پذیرش نمی‌کند.

۱٤. نشان دهید که $\hat{M}$ در مثال ۷-۱۱، $a^nb^{2n}c^k$ را با ضابطه $k>0$ پذیرش نمی‌کند. همچنین نشان دهید که $a^nb^mc^k$ فقط در صورتی پذیرش می‌شود که $m=n$ یا $m=2n$ باشد.

۱۵. نشان دهید که تمامی زبان‌های منظم، مستقل‌ازمتن معین هستند.

۱۶. نشان دهید که اگر L_1 مستقل‌ازمتن معین و L_2 منظم باشد، آنگاه زبان $L_1\cup L_2$ مستقل‌ازمتن معین خواهد بود.

۱۷. نشان دهید که براساس شرایط تمرین ۱۶، $L_1\cap L_2$ یک زبان مستقل‌ازمتن معین است.

۱۸. نمونه‌ای از یک زبان مستقل‌ازمتن معین ذکر کنید که معکوس آن معین نباشد.

۷-٤ گرامرهایی برای زبان‌های مستقل‌ازمتن معین*

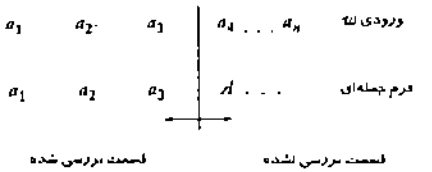
اهمیت زبان‌های مستقل‌ازمتن معین در تجزیه مفید و مؤثر آنها نهفته است. درستی این امر را می‌توان به صورت شهودی و با در نظر گرفتن اتومات پشته‌ای بعنوان ابزار تجزیه تحقیق نمود. بدلیل آنکه در این اتومات امکان بازگشت وجود ندارد، می‌توان یک برنامه کامپیوتری کارا برای آن نوشت. هرچند به دلیل احتمال وجود برخی انتقال‌های λ نمی‌توان آنرا بعنوان یک تجزیه کننده با زمان خطی در نظر گرفت و معرفی کرد، برنامه مذکور دستیابی به هدف و انتخاب مسیر درست را به مراتب ساده‌تر خواهد کرد. اما پیش از همه باید ببینیم از چه گرامرهایی می‌توان برای توصیف زبان‌های مستقل از متن معین استفاده کرد. به همین دلیل، مبحثی را مطرح می‌کنیم که علیرغم عدم جذابیت ظاهری خود، بیش

از هر چیز دیگری در کامپایلرها کاربرد دارد. در اینجا، برخی نتایج مهم این مبحث را معرفی می‌کنیم. خوانندگان علاقه‌مند می‌توانند برای تکمیل اطلاعات خود به منابع و مطالب نوشته شده در مورد کامپایلرها مراجعه کنند.

فرض کنید که بخواهیم به روش تجزیه پشته‌ای، یک اشتقاق چپ‌ترین برای یک رشته خاص ایجاد کنیم. به این منظور، از روش مشروح در شکل ۷-۵ استفاده می‌کنیم: ورودی w را از چپ به راست پویش کرده و درعین حال، فرم جمله‌ای می‌سازیم که پیشوند پایانی آن با پیشوند w تمامی سمبل‌هایی که تاکنون خوانده شده است، تطابق داشته باشد. برای موفقیت کامل در تطابق سمبل‌های متوالی، باید دقیقاً بدانیم که در هر مرحله از کدامیک از قوانین استفاده کنیم. به این ترتیب، علاوه بر جلوگیری از بازگشت، تجزیه‌کننده مفیدتری هم بوجود خواهد آمد. اما پیش از هر چیز باید بدانیم آیا اساساً گرامرهایی وجود دارند که تجزیه کننده مفیدتری را برای ما ایجاد کنند؟ پاسخ این پرسش در گرامرهای کلی مستقل‌ازمتن منفی است. اما اگر فرم گرامر محدود شود، به راحتی می‌توان گرامرهایی با ویژگی مورد نظر بوجود آورد.

ابتدا از s-گرامرهای تعریف ۵-۴ استفاده می‌کنیم. براساس این استدلال در هریک از مراحل تجزیه، دقیقاً می‌دانیم که کدام قانون باید بکار گرفته شود. فرض کنید که $w = w_1w_2$ و فرم جمله‌ای Ax، w_2 را هم اعمال کرده باشیم. برای بدست آوردن سمبل بعدی فرم جمله‌ای که با سمبل بعدی w هم تطابق داده شده باشد، فقط لازم است که به سمبل چپ‌ترین w_2، مثلاً a، نگاه کنیم. اگر هیچ قانون $A \to a$ در این گرامر وجود نداشته باشد، رشته w به این زبان تعلق ندارد. برعکس، اگر چنین قانونی وجود داشته باشد، می‌توان به کار تجزیه ادامه داد. با کمی بررسی متوجه می‌شویم که فقط یک قانون این‌چنین وجود دارد و بنابراین هیچ قدرت انتخابی نخواهیم داشت.

s-گرامرها علیرغم سودمندی خود، در کسب تمامی جنبه‌های نحو زبان‌های برنامه‌سازی چندان موفق نیستند. بنابراین، باید آنها را به گونه‌ای تعمیم دهیم که بدون از دست دادن قابلیت‌های اساسی خود در تجزیه، قدرت خود را هم افزایش دهند. از اینرو یکی از انواع معروف گرامرها به نام LLگرامر را معرفی می‌کنیم. یکی از ویژگی‌های شاخصLL گرامرها آن است که می‌توان با نگاهی به بخش محدود شده ورودی(شامل سمبل خوانده شده بعلاوه تعداد متناهی از سمبل‌ها بعد از آن)، به راحتی قانون مناسب را تعیین کرد. اصطلاح LL اصطلاح استانداردی است که در تمام منابع مربوط به کامپایلرها استفاده می‌شود. بهتر است بدانیم که L اول به معنای پویش از چپ به راست ورودی و L دوم هم به معنای ساخت اشتقاق‌های چپ‌ترین است. تمامی s-گرامرها، LL گرامر محسوب می‌شوند. مفهوم LL گرامر بسیار کلی‌تر از چیزی است که در ظاهر به نظر می‌رسد.



شکل ۷-۵

مثال ۷-۱۲

گرامر

$$S \to aSb \mid ab$$

یک s-گرامر نیست، اما LL گرامر است. برای تعیین قانون قابل استفاده در هر مورد، به دو سمبل متوالی رشته ورودی نگاه می‌کنیم. اگر اولی a و دومی b باشد، باید قانون $S \to ab$ را استفاده کنیم. در غیر اینصورت، از قانون $S \to aSb$ استفاده می‌کنیم.

یک گرامر را در صورتی$LL(k)$ گرامر می‌خوانیم که بتوان، با در اختیار داشتن سمبل جاری خوانده شده و"پیش‌بینی" $k-1$ سمبل بعدی، صرفاً قانون درست را انتخاب کرد. مثال ۷-۱۲ نمونه‌ای از یک $LL(2)$ گرامر بود.

مثال ۷-۱۳

گرامر

$$S \to SS \mid aSb \mid ab$$

بستار مثبت زبان مثال ۷-۱۲ می‌باشد. همانطور که در مثال ۵-۴ نیز گفتیم، این زبان نمونه‌ای از ساختارهای پرانتزی تودرتوی درست، می‌باشد. گرامر فوق برای هیچ مقداری از k ، یک گرامر $LL(k)$ نیست.

برای بررسی دلیل این ادعا، به اشتقاق رشته‌های با طول بزرگتر از دو نگاه می‌کنیم. در شروع کار، دو قانون ممکن $S \to SS$ و $S \to aSb$ را داریم. با بررسی سمبل خوانده شده نمی‌توان درستی یا نادرستی هیچکدام از این قوانین را تعیین کرد. اینک براساس روش پیش‌بینی که در بالا گفته شد، دو سمبل اول یعنی aa را در نظر می‌گیریم. آیا می‌توان در همین قدم اول تصمیم درستی گرفت؟ پاسخ هنوز هم منفی است، چون این دو سمبل ممکن است پیشوندی از رشته‌هایی مثل $aabb$ یا $aabbab$ باشند. در حالت اول، کار از $S \to aSb$ و در حالت دوم، از $S \to SS$ شروع می‌شود. بنابراین گرامر، هرچه که باشد، $LL(2)$نخواهد بود. در ادامه مشاهده می‌کنیم که هر تعداد سمبل را هم که پیش‌بینی کنیم، باز برخی موارد بدون حل باقی می‌مانند.

از این مشاهده نمی‌توان نتیجه گرفت که زبان معین نبوده یا اینکه هیچ LL گرامری به ازای آن وجود ندارد. باید با تجزیه و تحلیل دلیل شکست گرامر اول، یک LL گرامر برای این زبان ارائه کرد. اما نمی‌توان دقیقاً تعیین کرد که تا رسیدن به پایان رشته، الگوی اصلی "a^nb^n" چند مرتبه دیگر تکرار خواهد شد. بازنویسی گرامر راه‌حل مناسبی برای این مشکل است. گرامر

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

یک گرامر LL و تقریباً هم ارز با گرامر اصلی است.

برای تحقیق درستی این امر، اشتقاق چپ‌ترین $w = abab$ را در نظر بگیرید. آنگاه

$$S \Rightarrow aSbS \Rightarrow abS \Rightarrow abaSbS \Rightarrow ababS \Rightarrow abab.$$

مشاهده می‌کنیم که هرگز انتخابی نداریم. اگر سمبل ورودی بررسی شده a باشد، باید از $S \to aSbS$ استفاده کنیم. اما اگر سمبل b باشد، در انتهای رشته قرار داریم و بنابراین باید از $S \to \lambda$ استفاده شود. اما مسأله هنوز بطور کامل حل نشده، چون گرامر جدید می‌تواند رشته تهی تولید کند. این مشکل را با معرفی متغیر شروع جدید S_0 و یک قانون جدید حل می‌کنیم تا به این ترتیب، امکان تولید رشته تهی هم از بین برود. بنابراین، نتیجه نهایی

$$S_0 \to aSbS,$$
$$S \to aSbS \mid \lambda$$

یک LL گرامر و هم ارز با گرامر اصلی است.

هرچند توصیف غیرعملی قبلی داده شده از LL گرامرها، برای درک این مثال‌های ساده کفایت می‌کند، برای کسب نتایج دقیق‌تر، باید تعریف دقیق‌تری هم از این گرامر داشته باشیم. به همین دلیل، بحث خود را با تعریف جامع‌تری از LL گرامرها به پایان می‌رسانیم.

تعریف ۷-۵

فرض کنید $G = (V, T, S, P)$ یک گرامر مستقل‌ازمتن باشد. اگر به ازای تمام زوج‌های اشتقاق‌های چپ‌ترین

$$S \overset{*}{\Rightarrow} w_1 Ax_1 \Rightarrow w_1 y_1 x_1 \overset{*}{\Rightarrow} w_1 w_2,$$
$$S \overset{*}{\Rightarrow} w_1 Ax_2 \Rightarrow w_1 y_2 x_2 \overset{*}{\Rightarrow} w_1 w_3,$$

با فرض $w_1, w_2, w_3 \in T^*$، تساوی k تا سمبل چپ‌ترین w_2 و w_3 به طور ضمنی به معنای $y_1 = y_2$ باشد، آنگاه G را یک $LL(k)$ گرامر می‌خوانیم. (اگر $|w_2|$ یا $|w_3|$ کوچکتر از k باشد، آنگاه k با کوچکترین آنها جایگزین می‌شود).

این تعریف در واقع توضیحی است بر آنچه پیشتر گفته بودیم: چنانچه در هر یک از مراحل اشتقاق چپ‌ترین $(w_1 Ax)$، از k سمبل بعدی ورودی اطلاع داشته باشیم، صرفاً مرحله بعدی اشتقاق قابل تشخیص است (تساوی $y_1 = y_2$ هم تأییدی بر این امر است).

مبحث LL گرامرها یکی از مباحث مهم در مطالعه کامپایلرها محسوب می‌شود. بطوریکه برخی از زبان‌های برنامه‌سازی بوسیله LL گرامرها قابل تعریف بوده و بسیاری از کامپایلرها هم بر اساس

تجزیه‌کننده‌های LL نوشته می‌شوند. اما جامعیت LL گرامرها به حدی نیست که بتوان از آنها برای بحث در مورد تمامی زبانهای مستقل‌ازمتن معین استفاده کرد. در نتیجه، به گرامرهای معین جامع‌تر دیگری هم نیاز داریم. بعنوان یکی از مهم‌ترین این گرامرها می‌توان به گرامرهای موسوم به LR اشاره کرد که در عین ایجاد امکان تجزیه، در ساخت درخت‌های اشتقاق از پایین به بالا هم قابل استفاده هستند. در منابع مربوط به کامپایلرها (از جمله Hunter, ۱۹۸۱) و یا کتاب‌هایی که صرفاً به موضوع روش‌های تجزیه در زبان‌های رسمی پرداخته‌اند (از جمله Aho and Ullman ,1972) ، مطالب متنوعی می‌توان راجع به این گرامرها پیدا کرد.

تمرین‌ها

۱. نشان دهید که گرامر دوم مثال ۷-۱۳ یک LL گرامر بوده و بعلاوه، متناظر با گرامر اصلی است.

۲. نشان دهید که گرامر مربوط به زبان $L = \{w \in \{a,b\}^* : n_a(w) = n_b(w)\}$ که در مثال ۱-۱۳ داده شده، LL گرامر نیست.

۳. یک LL گرامر برای زبان تمرین ۲ پیدا کنید.

۴. یک LL گرامر برای زبان $L(a^*ba) \cup L(abbb^*)$ بسازید.

۵. نشان دهید که تمامی LL گرامرها غیرمبهم هستند.

۶. نشان دهید که اگر G یک $LL(k)$ گرامر باشد، آنگاه $L(G)$ یک زبان مستقل‌ازمتن معین است.

۷. نشان دهید که زبان‌های مستقل‌ازمتن معین، هیچگاه ذاتاً مبهم نیستند.

۸. فرض کنید G یک گرامر مستقل‌ازمتن به در فرم نرمال گریباخ باشد. الگوریتمی را توصیف کنید که به ازای هر k مفروض، تعیین کند که آیا G گرامر $LL(k)$ هست یا نه.

۹. با این فرض که $\Sigma = \{a,b,c\}$، LL گرامرهایی برای زبان‌های زیر ارائه دهید:

الف) $L = \{a^n b^m c^{n+m} : n \geq 0, m \geq 0\}$.

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

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

د) $L = \{w : n_a(w) < n_b(w)\}$.

ه) $L = \{w : n_a(w) + n_b(w) \neq n_c(w)\}$.

فصل ۸

خواص زبان‌های مستقل ازمتن

در طبقه‌بندی زبان‌های صوری، خانواده زبان‌های مستقل‌ازمتن در وسط قرارگرفته است؛ از یک طرف، زبان‌های مستقل‌ازمتن، خانواده زبان‌های مهم اما محدودی از قبیل زبان‌های مستقل‌ازمتن معین و منظم را شامل می‌شود. از طرف دیگر، خانواده زبان‌های گسترده‌تری وجود دارند که زبان‌های مستقل‌ازمتن، خانوادهٔ خاصی از آن محسوب می‌شوند. برای مطالعه ارتباط بین خانواده‌های زبان‌ها، درک تفاوت‌ها و شباهت‌های آنها، خواص خانواده‌های مختلف را مطالعه می‌کنیم. در این فصل، همانند فصل ۴، بسته بودن تحت انواع عملگرها، الگوریتم‌های مربوط به تصمیم‌گیری ویژگی‌های اعضای یک خانواده و نتایج مهمی از قبیل لم‌های تزریق را بررسی کرده و از آنها به عنوان ابزاری جهت درک روابط بین خانواده‌های مختلف و همچنین دسته‌بندی زبان‌های خاص در یک گروه استفاده می‌کنیم.

۸-۱ دو لم تزریق

از لم تزریق که در قضیه ۴-۸ مطرح شد، می‌توان بعنوان ابزاری مفید و مناسب جهت اثبات نامنظم بودن برخی زبان‌ها استفاده کرد. در مورد دیگر خانواده‌های زبان‌ها، لم تزریق مشابهی تعریف و ارائه شده است. در اینجا راجع به دو لم بحث خواهیم کرد، که یکی از آنها برای تمامی زبان‌های مستقل-ازمتن و دیگری برای زبان‌های مستقل‌ازمتن محدود شده‌ای به نام خطی کاربرد دارد.

لم تزریق برای زبان‌های مستقل‌ازمتن

قضیه ۸-۱

فرض کنید L یک زبان مستقل‌ازمتن نامتناهی باشد. آنگاه عدد صحیح مثبت m وجود دارد، بطوریکه

هر $w \in L$ با فرض $|w| \geq m$ را می‌توان به صورت

$$w = uvxyz, \tag{۸-۱}$$

با شرایط

$$|vxy| \leq m, \tag{۸-۲}$$

و

$$|vy| \geq 1, \tag{۸-۳}$$

چنان تجزیه کرد که به ازای هر $i = 0,1,2,\ldots$،

$$uv^i xy^i z \in L. \tag{۸-۴}$$

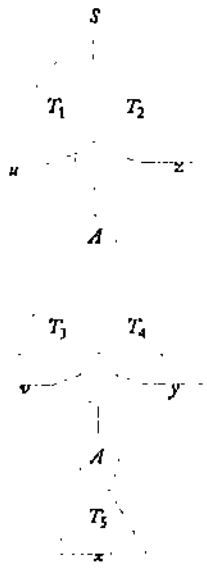
این نتیجه بعنوان لم تزریق برای زبان‌های مستقل‌ازمتن خوانده می‌شود.

اثبات: زبان $L - \{\lambda\}$ را در نظر گرفته و فرض کنید که به ازای آن، یک گرامر G فاقد هرگونه قانون-واحد یا قانون λ داشته باشیم. بدلیل آنکه طول رشته‌های واقع در سمت راست تمامی قوانین محدود است، مثلاً k، طول هر اشتقاق $w \in L$ باید حداقل $|w|/k$ باشد. بنابراین، به دلیل نامتناهی بودن L، اشتقاق‌هایی با طول اختیاری و درخت‌های اشتقاق نظیر با ارتفاع اختیاری وجود خواهند داشت.

حال یکی از این درخت‌های اشتقاق مرتفع و مسیری با طول مفروض از ریشه به یکی از برگها را در نظر بگیرید. براساس طرح کلی شکل ۸-۱، به دلیل متناهی بودن تعداد متغیرها در G، یکی از متغیرهای واقع در این مسیر باید تکرار شود. متناظر با درخت اشتقاق شکل ۸-۱، اشتقاق

$$S \overset{*}{\Rightarrow} uAz \overset{*}{\Rightarrow} uvAyz \overset{*}{\Rightarrow} uvxyz,$$

را داریم که در آن، u، v، x، y و z همگی رشته‌هایی از پایانی‌ها هستند. براساس رابطه فوق داریم $A \overset{*}{\Rightarrow} x$ و $A \overset{*}{\Rightarrow} vAy$، بنابراین تمامی رشته‌های $uv^i xy^i z$، $i = 0,1,2,\ldots$ بوسیله این گرامر قابل تولید بوده و لذا، در L قرار دارند. بعلاوه، می‌توان چنین فرض کرد که در اشتقاق‌های $A \overset{*}{\Rightarrow} vAy$ و $A \overset{*}{\Rightarrow} x$، هیچکدام از متغیرها تکرار نمی‌شوند. برای بررسی این مورد، به طرح کلی درخت اشتقاق شکل ۸-۱ نگاه کنید. در زیر درخت T_5 هیچ متغیر تکراری وجود ندارد؛ (در غیر اینصورت، این استدلال فقط در مورد آن متغیر تکراری قابل بررسی بود). به همین ترتیب، می‌توان فرض کرد که در زیر درخت‌های T_3 و T_4، هیچ متغیری تکرار نمی‌شود. بنابراین، طول رشته‌های v، x و y فقط به قوانین گرامر بستگی داشته و می‌توان آنها را مستقل از w و با فرض (۸-۲) کرانه‌دار در نظر گرفت. در نهایت، چون هیچ قانون واحد و قانون λ در گرامر وجود ندارد، رشته‌های v و y نمی‌توانند هر دو تهی باشند و بنابراین، (۸-۳) بدست می‌آید.



شکل ۸-۱

این نتیجه، استدلال را کامل می‌کند و روابط (۸-۱) تا (۸-۴) برقرار است. ■

از این لم تزریق که کارا می‌باشد، می‌توان برای اثبات عدم تعلق یک زبان مفروض به خانواده زبان‌های مستقل‌ازمتن استفاده نمود. بطورکلی کاربرد و استفاده از آن مشابه لم‌های تزریقی است که به صورت نقیض استفاده می‌شوند تا برای اثبات عدم تعلق یک زبان به یک خانواده خاص استفاده شوند. در قضیه ۴-۸ گفتیم که استدلال درست را می‌توان مانند بازی در برابر حریفی باهوش تجسم کرد، اما وجود قوانین فوق کار را تا حدی برای ما مشکل می‌کند. مثلاً در زبان‌های منظم، زیررشته xy با طول کرانه‌دار شده به m، از انتهای سمت چپ w آغاز می‌شود. بنابراین، زیررشته y که باید تزریق شود، در بین m سمبل آغازین w قرار دارد. در زبان‌های مستقل‌ازمتن، فقط یک کران روی $|vxy|$ داریم. زیررشته u که پیش از vxy قرار می‌گیرد، طول دلخواهی دارد. هرچند این ویژگی آزادی بیشتری به حریف می‌بخشد، این موارد، استدلال‌هایی را که براساس قضیه ۸-۱ انجام می‌شود، کمی پیچیده‌تر می‌کند.

مثال ۸-۱

نشان دهید که زبان

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

مستقل‌ازمتن نیست.

وقتی حریف m را انتخاب می‌کند، ما نیز رشته $a^m b^m c^m$ از L را انتخاب می‌کنیم. حال حریف

گزینه‌های مختلفی پیش‌رو دارد: اگر vxy را انتخاب کند که فقط حاوی a باشد، آنگاه رشته تزریق شده حتماً در L وجود نخواهد داشت. اما اگر صورت تجزیه‌ای را انتخاب کند که v و y آن دارای تعداد برابری a و b باشند، آنگاه رشته تزریق شده $a^k b^k c^m$ با ضابطه $k \neq m$ تولید می‌شود که این رشته بازهم در L وجود ندارد. در واقع، تنها گزینه‌ای که حریف برای شکست ما در بازی در اختیار دارد، انتخاب vxy به گونه‌ای است که vy دارای تعداد برابری a، b و c باشد. اما وجود محدودیت (۸-۱) این‌کار را غیرممکن می‌کند. بنابراین، L مستقل‌ازمتن نیست.

چنانچه در مورد زبان $L=\{a^n b^n\}$ از همین استدلال استفاده کنیم، به دلیل مستقل‌ازمتن بودن زبان، حتماً شکست خواهیم خورد. با انتخاب هر یک از رشته‌های عضو L، مثلاً $w=a^m b^m$، حریف ممکن است $v=a^k$ و $y=b^k$ را انتخاب کند. در اینصورت، هر i را که انتخاب کنیم، رشته تزریق شده حاصل از w، حتماً در L وجود خواهد داشت. البته به خاطر داشته باشید که این امر اثباتی بر مستقل از متن بودن L محسوب نمی‌شود؛ بلکه فقط می‌گوییم که نتوانسته‌ایم هیچ نتیجه‌ای از لم تزریق بگیریم. برای اثبات مستقل‌ازمتن بودن L باید از استدلال دیگری، مانند ساخت یک گرامرمستقل‌ازمتن، استفاده کرد.

بدلیل آنکه این استدلال، ادعای مطرح شده در مثال ۷-۱۱ را تأیید می‌کند، می‌توان از آن بعنوان ابزاری برای رفع خلاء موجود در آن مثال هم استفاده کرد. زبان

$$\hat{L}=\{a^n b^n\}\cup\{a^n b^{2n}\}\cup\{a^n b^n c^n\}$$

مستقل‌ازمتن نیست. رشته $a^m b^m c^m$ در $\hat{L}$ وجود دارد، در حالیکه نتیجه تزریق شده اینچنین نیست.

■

مثال ۸-۳

زبان زیر را در نظر بگیرید.

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

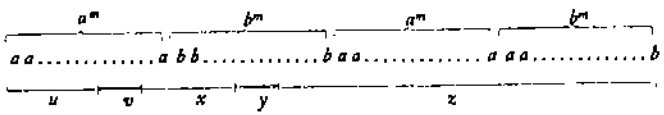
هرچند این زبان در ظاهر بسیار مشابه زبان مستقل‌ازمتن مثال ۵-۱ به نظرمی‌رسد، در واقع مستقل‌ازمتن نیست. رشته

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

را در نظر بگیرید. گرچه در این مورد حریف می‌تواند از روش‌های مختلفی برای انتخاب vxy استفاده کند، ما هم یک ضدحمله عالی برای هر یک از آنها داریم. بعنوان مثال، برای انتخاب فرم موجود در شکل ۸-۲ می‌توان با استفاده از $i=0$، رشته‌ای به فرم

$$a^k b^l a^m b^m, k<m \text{ یا } j<m$$

ایجاد کرد که در L وجود نداشته باشد. متناسب با هر انتخاب دیگر حریف، می‌توان استدلال‌هایی مشابه مطرح کرد. بنابراین، نتیجه می‌گیریم که زبان L مستقل‌ازمتن نیست.



شکل ۸-۲

■

مثال ۸-۴

نشان دهید که زبان

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

مستقل‌ازمتن نیست.

با توجه به اینکه حریف m را انتخاب می‌کند، ما $w=a^{m!}$ را انتخاب می‌کنیم. مشخص است که صورت تجزیه شده، هر چه که باشد، به فرم $v=a^k, y=a^l$ خواهد بود. پس، $w_0=uxz$ طولی معادل $m!-(k+l)$ دارد. این رشته فقط در صورتی در L قرار دارد که به ازای j مفروض،

$$m!-(k+l)=j!$$

اما چون با ضابطه $k+l\leq m$،

$$m-(k+l)>(m-1)!$$

و این امر امکان‌پذیر نیست. در نتیجه، زبان فوق مستقل‌ازمتن نیست.

■

مثال ۸-۵

نشان دهید که زبان

$$L=\{a^n b^j: n=j^2\}.$$

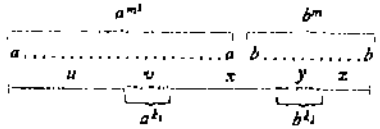
مستقل‌ازمتن نیست.

با درنظرگرفتن m تعریفی از قضیه ۸-۱، ما هم رشته $a^{m^2} b^m$ را انتخاب می‌کنیم. حال حریف گزینه‌های متعددی دارد. تنها موردی که باید به آن دقت کرد، در شکل ۸-۳ نمایش داده شده است. با i مرتبه تزریق رشته، رشته جدیدی با $m^2+(i-1)k_1$ تعداد a و $m+(i-1)k_2$ تعداد b تولید می‌شود. چنانچه حریف $k_1\neq 0$ و $k_2\neq 0$ را انتخاب کند، ما هم می‌توانیم $i=0$ را انتخاب کنیم. بدلیل آنکه

$$\begin{aligned}(m-k_2)^2 &\leq (m-1)^2\\ &=m^2-2m+1\\ &<m^2-k_1,\end{aligned}$$

نتیجه مورد نظر در L وجود ندارد. همچنین اگر حریف $k_1=0, k_2\neq 0$ یا $k_1\neq 0, k_2=0$ را انتخاب

کند، آنگاه مجدداً براساس فرض $i=0$، رشته تزریق شده در L وجود نخواهد داشت. از اینرو می‌توان نتیجه گرفت که زبان L مستقل‌ازمتن نیست.



شکل ۸-۳

□

یک لم تزریق برای زبان‌های خطی

قبلاً در مورد تفاوت بین گرامرهای مستقل‌ازمتن خطی و غیرخطی صحبت کردیم. در اینجا، راجع به همین تمایزها در زبان‌های مختلف بحث خواهیم کرد.

تعریف ۸-۱

زبان مستقل‌ازمتن L در صورتی خطی خوانده می‌شود که گرامر مستقل‌ازمتن خطی G وجود داشته باشد، بطوریکه $L=L(G)$ باشد.

می‌دانیم که تمامی زبان‌های خطی، مستقل‌ازمتن هستند، اما هنوز نمی‌توان با قطعیت راجع به درستی حالت عکس هم حرفی زد.

مثال ۸-۱

زبان $L=\{a^nb^n : n\geq 0\}$ یک زبان خطی است. یکی از گرامرهای خطی این زبان در مثال ۱-۱۱ آمده است. گرامر ارائه شده در مثال ۱-۱۳ برای زبان $L=\{w\in\{a,b\}^* : n_a(w)=n_b(w)\}$ خطی نبوده و بنابراین، این زبان لزوماً خطی نیست.

□

البته، تنها براساس خطی‌نبودن یک گرامر خاص، نمی‌توان گفت که زبان تولید شده بوسیله آن خطی نیست. برای اثبات خطی نبودن یک زبان، باید عدم وجود گرامر خطی متناظر با آن را اثبات کنیم. اینکار را می‌توان به روش معمول انجام داد؛ یعنی تعاریفی برای ساخت گرامر برای زبان‌های خطی ایجاد کرده و سپس نشان داد که زبان‌های مستقل‌ازمتن مفروض فاقد یکی از این ویژگی‌های لازم هستند.

قضیه ۸-۲

فرض کنید L یک زبان خطی نامتناهی باشد. آنگاه عدد صحیح و مثبت m وجود خواهد داشت،

بطوریکه هر $w\in L$ با ضابطه $|w|\geq m$ را می‌توان به صورت $w=uvxyz$ با ضابطه‌های

$$|uvyz|\leq m, \qquad (۸-۵)$$

$$|vy|\geq 1, \qquad (۸-۶)$$

به صورتی تجزیه کرد که به ازای تمام $i=0,1,2,\ldots$،

$$uv^ixy^iz\in L. \qquad (۸-۷)$$

توجه داشته باشید به دلیل آنکه به جای (۸-۲) از (۸-۵) استفاده کرده‌ایم، نتایج این قضیه با نتایج قضیه ۸-۱ تفاوت دارد! یعنی برای آنکه رشته‌های v و y تزریق شوند، باید به ترتیب در بین m سمبل انتهای چپ و راست w قرار داشته باشند. طول رشته میانی x اختیاری است.

اثبات: به دلیل خطی بودن این زبان، یک گرامر خطی G به ازای آن وجود خواهد داشت. در این استدلال، باید G را فاقد هرگونه قانون‌واحد و قانون λ درنظرگرفت. با بررسی اثبات‌های قضایای ۶-۳ و ۶-۴ مشخص می‌شود که حذف قوانین‌واحد و قوانین λ خدشه‌ای به خطی‌بودن گرامر وارد نمی‌کند. بنابراین می‌توان تصور کرد که G ویژگی لازم را دارا می‌باشد.

حال اشتقاق

$$S\Rightarrow uAz\Rightarrow uvAyz\Rightarrow uvxyz=w$$

از رشته $w\in L(G)$ را در نظر بگیرید:

فرض کنید که به ازای هر $w\in L(G)$، یک متغیر A وجود دارد، بطوریکه

۱. در اشتقاق جزئی $S\Rightarrow uAz$ هیچ متغیری تکرار نمی‌شود،

۲. در اشتقاق جزئی $S\Rightarrow uAz\Rightarrow uvAyz$، به غیر از A، هیچ کدام از متغیرها تکرار نمی‌شوند،

۳. تکرار A باید طی m مرحله اول انجام شود و m به گرامر بستگی دارد نه به w.

در صورت برقرار بودن شرایط فوق، آنگاه طول‌های u، y، v و z باید مستقل از w کرانه‌دار شوند. این مطلب درستی ضمنی (۸-۵)، (۸-۶) و (۸-۷) را اثبات می‌کند.

برای تکمیل استدلال، مجدداً باید نشان دهیم که شرایط فوق در مورد تمامی گرامرهای خطی برقرار است. این‌کار با بررسی دنباله وقوع متغیرها به راحتی قابل انجام است. فعلاً از ارائه جزئیات صرف‌نظر کرده و آنها را بعنوان تمرین ۱۶ در انتهای همین بخش ذکر می‌کنیم. ■

مثال ۸-۲

زبان

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

خطی نیست.

برای اثبات، فرض کنید که این زبان خطی بوده و قضیه ۸-۲ را برای رشته

$$w = a^m b^{2m} a^m$$

بکار بریم. نامساوی (۸-۷) نشان می‌دهد که در این حالت، رشته‌های u، v و z باید تماماً از a تشکیل شوند. اگر این رشته را یک مرتبه تزریق کنیم، $a^{m-k}b^{2m}a^{m+l}$ یا فرض $k \geq 1$ یا $l \geq 1$ بدست می‌آید که در L وجود ندارد. این نتیجه با قضیه ۸-۲ در تضاد می‌باشد و بنابراین، زبان فوق خطی نیست.

■

مثال اخیر پاسخی به پرسش کلی مطرح شده در مورد رابطه بین خانواده زبان‌های خطی و مستقل-ازمتن است. با این مثال می‌توان گفت که خانواده زبان‌های خطی یکی از زیرمجموعه‌های مناسب خانواده زبان‌های مستقل‌ازمتن هستند.

تمرین‌ها

۱. نشان دهید که زبان

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

مستقل‌ازمتن نیست.

۲. نشان دهید که زبان $\{n$ یک عدد اول است $: L = \{a^n$ مستقل‌ازمتن نیست.

۳. نشان دهید که $L = \{ww^R w : w \in \{a,b\}^*\}$ یک زبان مستقل‌ازمتن نیست. ⊖

۴. نشان دهید که زبان $L = \{w \in \{a,b,c\}^* : n_a^2(w) + n_b^2(w) = n_c^2(w)\}$ مستقل‌ازمتن نیست.

۵. آیا زبان $L = \{a^n b^m : n = 2^m\}$ مستقل‌ازمتن است؟

۶. نشان دهید که زبان $L = \{a^{n^2} : n \geq 0\}$ مستقل‌ازمتن نیست.

۷. نشان دهید که زبان‌های زیر روی $\Sigma = \{a,b,c\}$ مستقل‌ازمتن نیستند.

الف) $L = \{a^n b^j : n \leq j^2\}$. ⊖

ب) $L = \{a^n b^j : n \geq (j-1)^3\}$.

ج) $L = \{a^n b^j c^k : k = jn\}$.

د) $L = \{a^n b^j c^k : k > n, k > j\}$.

ه) $L = \{a^n b^j c^k : n < j, n \leq k \leq j\}$.

و) $L = \{w : n_a(w) < n_b(w) < n_c(w)\}$. ⊖

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

ح) $L = \{w \in \{a,b,c\}^* : n_a(w) + n_b(w) = 2n_c(w), n_a(w) = n_b(w)\}$.

ط) $L = \{a^n b^m :$ n و m هر دو اول هستند$\}$

ی) $L = \{a^n b^m :$ n اول است یا m اول است$\}$

ک) $L = \{a^n b^m :$ n اول است و m اول نیست$\}$

۸. تعیین کنید کدامیک از زبان‌های زیر مستقل‌ازمتن هستند.

الف) $L = \{a^n ww^R a^n : n \geq 0, w \in \{a,b\}^*\}$.

ب) $L = \{a^n b^j a^n b^j : n \geq 0, j \geq 0\}$. ⊖

ج) $L = \{a^n b^j a^j b^n : n \geq 0, j \geq 0\}$.

د) $L = \{a^n b^j a^k b^l : n + j \leq k + l\}$.

ه) $L = \{a^n b^j a^k b^l : n \leq k, j \leq l\}$.

و) $L = \{a^n b^n c^j : n \leq j\}$.

ز) $L = \{w \in \{a,b,c\}^* : n_a(w) = n_b(w) = 2n_c(w)\}$.

۹. در قضیه ۸-۱، یک کرانه برای m برحسب خواص گرامر G پیدا کنید.

۱۰. مستقل‌ازمتن بودن یا نبودن زبان زیر را بررسی کنیم.

$$L = \{w_1 c w_2 : w_1, w_2 \in \{a,b\}^*, w_1 \neq w_2\}.$$ ⊖

۱۱. نشان دهید که زبان $L = \{a^n b^n a^m b^m : n \geq 0, m \geq 0\}$ مستقل‌ازمتن و غیرخطی است.

۱۲. نشان دهید که زبان زیر خطی نیست.

$$L = \{w : n_a(w) \geq n_b(w)\}.$$ ⊖

۱۳. نشان دهید که زبان $L = \{w \in \{a,b,c\}^* : n_a(w) + n_b(w) = n_c(w)\}$ مستقل‌ازمتن و غیرخطی است.

۱۴. تعیین کنید آیا زبان $L = \{a^n b^j : j \leq n \leq 2j-1\}$ خطی است یا خیر.

۱۵. تعیین کنید آیا زبان مثال ۵-۱۲ خطی است یا خیر.

۱۶. فرض کنید G یک گرامر خطی دارای k متغیر باشد. نشان دهید که در هر دنباله از متغیرها، متغیر A باید به صورتی تکرار شود که

الف) اولین وقوع A حتماً در موقعیت $p \leq k$ باشد،

ب) تکرار A باید حداکثر پیش از $q \leq k+1$ رخ دهد و

ج) هیچ متغیر تکراری دیگری نباید بین موقعیت‌های p و q وجود داشته باشد.

۱۷. ادعای مطرح شده در قضیه ۸-۲ را ثابت کنید :

به ازای هر زبان خطی(فاقد λ) یک گرامر خطی فاقد قانون λ و قانون واحد وجود دارد.

۱۸. مجموعه تمام رشته‌های a/b را در نظر بگیرید که در آن، a و b اعداد صحیح اعشاری مثبت با ضابطه $a < b$ باشند. در این‌صورت این مجموعه رشته‌های مفروض همهٔ کسرهای ممکن دهدهی را نمایش می‌دهد. تعیین کنید آیا این زبان مستقل‌ازمتن هست یا خیر؟

• – نشان دهید که مکمل زبان تمرین ۶ مستقل‌ازمتن نیست.

آیا زبان $L = \{a^{nm} : n \text{ و } m \text{ اعداد اول هستند}\}$ مستقل‌ازمتن است؟

۸-۳ ویژگی‌های بستاری و الگوریتم‌های تصمیم‌گیری برای زبان‌های مستقل‌ازمتن

در فصل ۴، ضمن مطالعه بسته بودن تحت برخی عملگرها و الگوریتم‌ها، راجع به خواص خانواده زبان‌های منظم تصمیم‌گیری کردیم. تمامی سؤالات مطرح شده در آن فصل به راحتی قابل پاسخگویی بودند. اما وقتی همین سؤالات را درباره زبان‌های مستقل‌ازمتن مطرح می‌کنیم، کار کمی مشکل‌تر می‌شود. چون اولاً، خواص بسته بودن که در زبان‌های منظم برقرار است، لزوماً همگی برای در زبان‌های مستقل‌ازمتن برقرار نمی‌باشد. حتی اگر هم چنین باشد، استدلال‌های لازم برای اثبات درستی آنها غالباً بسیار پیچیده است. ثانیاً، برای بسیاری از پرسش‌های مربوط به زبان‌های مستقل‌ازمتن که از نظر شهودی مهم و ساده به نظر می‌رسند، نمی‌توان پاسخ مشخصی ارائه کرد. این گفته ممکن است در ابتدا تعجب‌برانگیز به نظر برسد و به همین دلیل، در ادامه آنرا شرح و بسط خواهیم داد. در این بخش، فقط به نمونه‌ای از برخی خواص مهم اشاره می‌کنیم.

بسته بودن زبان‌های مستقل‌ازمتن

قضیه ۸-۳

خانواده زبان‌های مستقل‌ازمتن تحت اجتماع، الحاق و بستار ستاره‌ای بسته است.

اثبات: فرض کنید L_1 و L_2 دو زبان مستقل‌ازمتن باشند که به ترتیب، بوسیله گرامرهای مستقل‌ازمتن $G_1 = (V_1, T_1, S_1, P_1)$ و $G_2 = (V_2, T_2, S_2, P_2)$ تولید می‌شوند. می‌توان، ضمن حفظ کلیت موضوع، می‌توان مجموعه‌های V_1 و V_2 را جدا از هم در نظر گرفت.

اینک فرض کنید که زبان $L(G_3)$ بوسیله گرامر

$$G_3 = (V_1 \cup V_2 \cup \{S_3\}, T_1 \cup T_2, S_3, P_3),$$

تولید می‌شود و S_3 متغیری است که در $V_1 \cup V_2$ وجود ندارد. قوانین G_3 همگی از قوانین G_1 و G_2 به همراه یک قانون شروع انتخابی جدید تشکیل شده، تا به این ترتیب، بتوان از دو گرامر مختلف استفاده کرد. به بیان دقیق‌تر،

$$P_3 = P_1 \cup P_2 \cup \{S_3 \to S_1 \mid S_2\}.$$

مشخص است که G_3 یک گرامر مستقل‌ازمتن بوده و بنابراین $L(G_3)$ هم یک زبان مستقل‌ازمتن است. اما مشاهده می‌کنیم که

$$L(G_3) = L_1 \cup L_2. \tag{۸-۸}$$

بعنوان مثال، فرض کنید که $w \in L_1$ باشد. آنگاه

$$S_3 \Rightarrow S_1 \Rightarrow w$$

یکی از اشتقاق‌های ممکن گرامر G_3 است. می‌توان راجع به $w \in L_2$ هم استدلال مشابهی را مطرح کرد. همچنین، اگر $w \in L(G_3)$ باشد، آنگاه یا

$$S_3 \Rightarrow S_1 \tag{۸-۹}$$

یا

$$S_3 \Rightarrow S_2 \tag{۸-۱۰}$$

اولین مرحله از اشتقاق خواهند بود. فرض کنید از (۸-۹) استفاده شود. بدلیل آنکه فرم‌های جمله‌ای اشتقاق شده از S_1 متغیرهایی در V_1 داشته و V_1 و V_2 جدا از هم هستند، قوانین اشتقاق

$$S_1 \Rightarrow w$$

فقط در P_1 قرار دارد. از اینرو w حتماً در L_1 خواهد بود. اما اگر ابتدا از (۸-۱۰) استفاده شود، آنگاه w حتماً در L_2 قرار دارد و در نتیجه، $L(G_3)$ اجتماع L_1 و L_2 است.

سپس گرامر،

$$G_4 = (V_1 \cup V_2 \cup \{S_4\}, T_1 \cup T_2, S_4, P_4)$$

را در نظر بگیرید. در اینجا نیز S_4 یک متغیر جدید بوده و

$$P_4 = P_1 \cup P_2 \cup \{S_4 \to S_1 S_2\}.$$

آنگاه، به راحتی بدست می‌آید که

$$L(G_4) = L(G_1)L(G_2).$$

در نهایت، $L(G_5)$ با فرض

$$G_5 = (V_1 \cup \{S_5\}, T_1, S_5, P_5),$$

را در نظر بگیرید که در آن، S_5 یک متغیر جدید است و

$$P_5 = P_1 \cup \{S_5 \to S_1 S_5 \mid \lambda\}.$$

پس

$$L(G_5) = L(G_1)^*.$$

بنابراین نشان دادیم که خانواده زبان‌های مستقل‌ازمتن تحت اجتماع، الحاق و بستار ستاره‌ای، بسته است. ■

قضیه ۸-٤

خانواده زبان‌های مستقل‌ازمتن تحت اشتراک و مکمل‌گیری بسته نیست.

اثبات: دو زبان

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

و

$$L_2 = \{a^n b^m c^m : n \geq 0, m \geq 0\}$$

را در نظر بگیرید. روش‌های مختلفی برای اثبات مستقل‌ازمتن بودن L_1 و L_2 وجود دارد. بطورنمونه، یکی از گرامرهای L_1

$$S \to S_1 S_2,$$
$$S_1 \to aS_1 b \mid \lambda,$$
$$S_2 \to cS_2 \mid \lambda.$$

است.

همچنین، می‌توان گفت که L_1 الحاق دو زبان مستقل‌ازمتن بوده و بنابراین، براساس قضیه ۸-۳ مستقل‌ازمتن است. اما

$$L_1 \cap L_2 = \{a^n b^n c^n : n \geq 0\},$$

که قبلاً مستقل‌ازمتن نبودن آنرا اثبات کردیم. بنابراین، خانواده زبان‌های مستقل‌ازمتن تحت اشتراک بسته نیست.

برای اثبات بخش دوم قضیه، از قضیه ۸-۳ و معرفی مجموعه

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

استفاده می‌کنیم. اگر خانواده زبان‌های مستقل‌ازمتن تحت مکمل‌گیری بسته باشد، آنگاه سمت راست عبارت فوق، برای تمامی زبان‌های مستقل‌ازمتن L_1 و L_2، یک زبان مستقل‌ازمتن خواهد بود. اما این مطلب با آنچه اخیراً اثبات کردیم - اشتراک دو زبان مستقل از متن لزوماً مستقل‌ازمتن نیست- در تناقض است. در نتیجه، خانواده زبان‌های مستقل‌ازمتن تحت مکمل‌گیری بسته نیست. ■

هرچند زبان بدست آمده از اشتراک دو زبان مستقل‌ازمتن، ممکن است مستقل‌ازمتن نباشد، ولی در صورتی که یکی از این زبان‌ها منظم باشد، می‌توان از بسته بودن اشتراک آنها سخن گفت.

قضیه ۸-۵

فرض کنید L_1 یک زبان مستقل‌ازمتن و L_2 یک زبان منظم باشد. آنگاه $L_1 \cap L_2$ مستقل‌ازمتن است.

اثبات: $M_1 = (Q, \Sigma, \Gamma, \delta_1, q_0, z, F_1)$ را npda ای فرض کنید که L_1 را پذیرش می‌کند و

$M_2 = (P, \Sigma, \delta_2, p_0, F_2)$ dfa ای است که L_2 را می‌پذیرد. اتومات پشته‌ای $\hat{M} = (\hat{Q}, \Sigma, \Gamma, \hat{\delta}, \hat{q}_0, z, \hat{F})$ را می‌سازیم که حرکت موازی M_1 و M_2 را شبیه‌سازی می‌کند؛ یعنی وقتی سمبلی از رشته ورودی خوانده می‌شود، $\hat{M}$ بطور همزمان M_1 و M_2 را به حرکت در می‌آورد. به این منظور فرض می‌کنیم که

$$\hat{Q} = Q \times P,$$
$$\hat{q}_0 = (q_0, p_0),$$
$$\hat{F} = F_1 \times F_2,$$

و $\hat{\delta}$ را به صورتی تعریف می‌کنیم که

$$((q_k, p_l), x) \in \hat{\delta}((q_i, p_j), a, b)$$

اگر و تنها اگر

$$(q_k, x) \in \delta_1(q_i, a, b)$$

و

$$\delta_2(p_j, a) = p_l.$$

همچنین لازم است که اگر $a = \lambda$ باشد، آنگاه $p_j = p_l$ درنظرگرفته می‌شود. به بیان دیگر، حالت‌های $\hat{M}$ با زوج‌های (q_i, p_j) نامگذاری می‌شوند. این زوج‌ها بیانگر حالت‌هایی هستند که ممکن است M_1 و M_2، پس از خواندن یک رشته ورودی به در آن قرار بگیرند. به راحتی می‌توان به کمک استدلال استقراء نشان داد که

$$((q_0, p_0), w, z) \vdash^*_{\hat{M}} ((q_r, p_s), \lambda, x),$$

با فرض $q_r \in F_1$ و $p_s \in F_2$، اگر و تنها اگر

$$(q_0, w, z) \vdash^*_{M_1} (q_r, \lambda, x),$$

و

$$\delta^*(p_0, w) = p_s.$$

بنابراین، یک رشته بوسیله $\hat{M}$ پذیرش می‌شود اگر و تنها اگر بوسیله M_1 و M_2 پذیرش شود؛ یعنی اگر آن رشته در $L(M_1) \cap L(M_2) = L_1 \cap L_2$ وجود داشته باشد. ■

خاصیت مورد اشاره در این قضیه به نام بسته بودن تحت اشتراک منظم خوانده می‌شود. بنابر نتیجه قضیه، می‌گوییم که خانواده زبان‌های مستقل‌ازمتن تحت اشتراک منظم بسته است. از این خاصیت بسته بودن می‌توان گاهی اوقات در ساده‌کردن استدلال‌ها برای برخی زبان‌ها استفاده کرد.

مثال ۸-۵

نشان دهید که زبان

$$L=\{a^n b^n : n\geq 0, n\neq 100\}$$

مستقل‌ازمتن است.

این ادعا را می‌توان با ساخت یک pda یا یک گرامر مستقل‌ازمتن برای زبان فوق اثبات کرد. با استفاده از قضیه ۸-۵ این استدلال راحت‌تر می‌شود.

فرض کنید

$$L_1=\{a^{100}b^{100}\}.$$

آنگاه، چون L_1 متناهی است، منظم می‌باشد. همچنین، به راحتی می‌توان نشان داد که

$$L=\{a^n b^n : n\geq 0\}\cap \overline{L_1}.$$

بنابراین، براساس بسته بودن زبان‌های منظم تحت مکمل‌گیری و بسته بودن زبان‌های مستقل‌ازمتن تحت اشتراک منظم، نتیجه مورد نظر بدست می‌آید.

■

مثال ۸-۶

نشان دهید که زبان

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

مستقل از متن نیست.

هر چند می‌توان در این‌مورد از لم تزریق استفاده کرد، استفاده از بسته بودن تحت اشتراک منظم استدلال را بسیار کوتاه‌تر می‌کند. فرض کنید که L مستقل‌ازمتن باشد. آنگاه

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

هم مستقل‌ازمتن خواهد بود. اما چون این‌گونه نیست، نتیجه می‌گیریم که L هم مستقل‌ازمتن نیست.

■

خواص بسته بودن زبان‌ها نقش مهمی در نظریه زبان‌های صوری ایفا می‌کنند. بعلاوه، به کمک آنها می‌توان خواص بسته بودن بسیار بیشتری را برای زبان‌های مستقل‌ازمتن اثبات کرد. برخی از این نتایج مهم در قالب تمرین در انتهای همین بخش آمده است.

برخی خواص تصمیم‌پذیر زبان‌های مستقل‌ازمتن

با کنار هم قرار دادن قضایای ۵-۲ و ۶-۶ وجود یک الگوریتم عضویت برای زبان‌های مستقل‌ازمتن را اثبات کردیم. البته این الگوریتم یکی از خواص اساسی هر خانواده زبانی است که در کاربرد عملی

سودمند است. می‌توان برخی ویژگی‌های ساده‌تر را هم برای زبان‌های مستقل‌ازمتن تعیین کرد. اما فعلاً فرض می‌کنیم که زبان بوسیله گرامر خود پذیرفته می‌شود.

قضیه ۸-۶

با داشتن گرامر مستقل‌ازمتن $G=(V,T,S,P)$، یک الگوریتم برای تصمیم‌گیری در مورد تهی بودن یا نبودن $L(G)$ وجود دارد.

اثبات: برای سهولت کار فرض کنید که $\lambda\notin L(G)$. درغیر اینصورت، باید تغییرات جزئی در استدلال ایجاد کرد. ما از الگوریتم مربوط به حذف متغیرها و قوانین غیرمفید استفاده می‌کنیم. اگر به این نتیجه برسیم که S غیرمفید است، $L(G)$ حداقل دارای یک عضو می‌باشد. ■

قضیه ۸-۷

با داشتن گرامر مستقل‌ازمتن $G=(V,T,S,P)$، الگوریتمی برای تعیین نامتناهی بودن یا نبودن $L(G)$ وجود دارد.

اثبات: فرض می‌کنیم که در G هیچ قانون λ، قانون واحد و متغییر غیرمفیدی وجود ندارد. تصور کنید که در این گرامر یک متغیر تکراری وجود دارد، بطوریکه یک $A\in V$ و یک اشتقاق

$$A\overset{+}{\Rightarrow} xAy$$

برای آن داریم. بدلیل آنکه G را فاقد هر قانون واحد و قانون λ فرض کردیم، x و y نمی‌توانند به صورت همزمان تهی باشند. از آنجایی‌که A نه یک متغیر میرا و نه یک متغیر غیرمفید است، بنابراین داریم

$$S\overset{*}{\Rightarrow} uAv\overset{*}{\Rightarrow} w$$

و

$$A\overset{*}{\Rightarrow} z,$$

که در آن u,v,z در T^* قرار دارند. اما در اینصورت

$$S\overset{*}{\Rightarrow} uAv\overset{*}{\Rightarrow} ux^nAy^nv\overset{*}{\Rightarrow} ux^nzy^nv$$

برای تمامی n ها امکان‌پذیر بوده و بنابراین $L(G)$ نامتناهی است.

حال اگر هیچ‌کدام از متغیرها تکرار نشوند، آنگاه طول تمامی اشتقاق‌ها بوسیله $|V|$ کرانه‌دار خواهد شد. در این صورت، $L(G)$ متناهی است.

بنابراین، برای ارائه الگوریتمی برای تعیین متناهی بودن یا نبودن $L(G)$، فقط باید ببینیم که آیا در این گرامر یک متغیر تکراری وجود دارد یا خیر. این‌کار را می‌توان به راحتی و با ترسیم گراف

وابستگی برای متغیرها انجام داد. توجه کنید، در صورتی یال (A,B) وجود دارد که قانون متناظر

$$A \to xBy$$

برای آن وجود داشته باشد.

پس، تمامی متغیرهایی که در حلقه قرار داشته باشند، متغیرهای تکراری هستند. در نتیجه، گرامر فوق دارای متغیر تکرای خواهد بود اگر و تنها اگر گراف وابستگی آن یک حلقه داشته باشد.

از آنجایی‌که اکنون یک الگوریتم برای تصمیم‌گیری در مورد وجود متغیر تکراری در گرامر داریم، از این الگوریتم می‌توان برای تعیین نامتناهی بودن یا نبودن $L(G)$ استفاده کرد. □

نکته جالب اینکه، دیگر خواص ساده زبان‌های مستقل‌ازمتن به همین راحتی قابل بحث و بررسی نیست. مثلاً به کمک هیچ الگوریتمی نمی‌توان تعیین کرد که آیا دو گرامر مستقل‌ازمتن مفروض یک زبان یکسان را تولید می‌کنند یا خیر و بنابراین، جوابی برای قضیه ۴-۷ وجود ندارد. در حال حاضر، نمی‌توان تعریف فنی مناسب و دقیقی برای عبارت "هیچ الگوریتمی وجود ندارد" ارائه کرد، اما معنای شهودی این جمله مشخص است. در ادامه بحث دوباره در مورد این نکته مهم توضیح خواهیم داد.

تمرین‌ها

۱. آیا مکمل زبان مثال ۸-۸ مستقل‌ازمتن است؟ ⊖

۲. زبان L_4 در قضیه ۸-۴ را در نظر بگیرید. نشان دهید که این زبان خطی است.

۳. نشان دهید که خانواده زبان‌های مستقل‌ازمتن تحت هم‌ریختی بسته است

۴. نشان دهید که خانواده زبان‌های خطی تحت هم‌ریختی بسته است.

۵. نشان دهید که خانواده زبان‌های مستقل‌ازمتن تحت معکوس بسته است. ⊖

۶. کدامیک از خانواده زبان‌هایی که تا اینجا بحث کردیم، تحت معکوس بسته نیستند؟

۷. نشان دهید که خانواده زبان‌های مستقل‌ازمتن در حالت کلی، تحت تفاضل بسته نیست، اما تحت تفاضل منظم بسته است؛ یعنی اگر L_1 مستقل‌ازمتن و L_2 منظم باشد، آنگاه $L_1 - L_2$ مستقل‌ازمتن است.

۸. نشان دهید که خانواده زبان‌های مستقل‌ازمتن معین تحت تفاضل منظم بسته است.

۹. نشان دهید که خانواده زبان‌های خطی تحت اجتماع بسته است، اما تحت الحاق بسته نیست. ⊖

۱۰. نشان دهید که خانواده زبان‌های خطی تحت اشتراک بسته نیست.

۱۱. نشان دهید که خانواده زبان‌های مستقل‌ازمتن معین تحت اجتماع و اشتراک بسته نیست.

۱۲. مثالی از یک زبان مستقل از متن ارائه دهید که مکمل آن مستقل‌ازمتن نباشد.

۱۳. • نشان دهید که اگر L_1 خطی و L_2 منظم باشد، آنگاه L_1L_2 یک زبان خطی است. ⊖

۱۴. نشان دهید که خانواده زبان‌های مستقل‌ازمتن غیرمبهم، تحت اجتماع بسته نیست.

۱۵. نشان دهید که خانواده زبان‌های مستقل‌ازمتن غیرمبهم، تحت اشتراک بسته نیست. ⊖

۱۶. L را یک زبان مستقل‌ازمتن معین فرض کرده و زبان جدید $L_1 = \{w : aw \in L, a \in \Sigma\}$ را تعریف می‌کنیم. آیا لزوماً L_1 یک زبان مستقل‌ازمتن معین است؟

۱۷. نشان دهید که زبان $\{a^n b^n : n \geq 0\}$ یکی از ضرایب ۵ نیست L مستقل‌ازمتن است.

۱۸. نشان دهید که زبان زیر مستقل‌ازمتن است:

$$L = \{w \in \{a,b\}^* : n_a(w) = n_b(w)\}.$$ و w دارای زیر رشتهٔ aab نمی‌باشد.

۱۹. آیا خانواده زبان‌های مستقل‌ازمتن معین تحت هم‌ریختی بسته است؟

۲۰. جزئیات استدلال استقرایی قضیه ۸-۵ را کامل کنید.

۲۱. الگوریتمی ارائه دهید که بتوان به کمک آن، به ازای هر گرامر مستقل‌ازمتن G، درستی یا نادرستی $\lambda \in L(G)$ را تعیین کرد. ⊖

۲۲. نشان دهید می‌توان بوسیله الگوریتمی تعیین کرد که آیا در زبان تولید شده بوسیله یک گرامر مستقل‌ازمتن، رشته‌هایی با طول کمتر از عدد مفروض n وجود دارد.

۲۳. فرض کنید که L_1 یک زبان مستقل‌ازمتن و L_2 منظم باشد. نشان دهید که می‌توان بوسیله یک الگوریتمی وجود عضو مشترک در L_1 و L_2 را تعیین کرد.

فصل

ماشین‌های تورینگ

در فصل‌ها و مباحث گذشته، با برخی از ایده‌های اساسی از قبیل مفاهیم زبان‌های منظم و مستقل‌ازمتن و همچنین ارتباط آنها با اتوماتای متناهی و پذیرنده‌های پشته‌ای آشنا شدیم. در جریان مطالعه خود پی بردیم که زبان‌های منظم یکی از زیرمجموعه‌های محض زبان‌های مستقل‌ازمتن بوده و به همین دلیل، اتوماتای پشته‌ای قدرتمندتر از اتوماتای متناهی هستند. همچنین بطور گذرا مشاهده کردیم که زبان‌های مستقل‌ازمتن، هرچند اساس مطالعه زبان‌های برنامه‌سازی را تشکیل می‌دهند، گستردگی و جامعیت چندانی ندارند. این نقطه ضعف را در فصل قبل به کمک مثال‌هایی به طور کامل شرح داده و نتیجه گرفتیم که برخی زبان‌های ساده از قبیل $\{a^nb^nc^n\}$ و $\{ww\}$ مستقل‌ازمتن نیستند. بنابراین باید حوزه‌های مطالعاتی خود را گسترش داده، با شجاعت پا را از جزیره زبان‌های مستقل‌ازمتن بیرون بگذاریم و از روبرو شدن با دنیاهای جدید ترس به خود راه ندهیم. امیدوار باشیم که بتوانیم خانواده زبان‌های جدیدی را کشف کنیم که این مثال‌ها را هم شامل شود. به این منظور، باید تصویر کلی را که قبلاً از اتومات در ذهن خود داشتیم، مجدداً بررسی و کمی اصلاح کنیم. با مقایسه اتومات متناهی با اتومات پشته‌ای درمی‌یابیم که امکان ذخیره‌سازی موقت، وجه تمایز این دو محسوب می‌شود. بطوریکه با حذف حافظه، یک اتومات متناهی ایجاد شده و اگر از پشته بعنوان حافظهٔ استفاده کنیم، اتومات پشته‌ای خود را قدرتمندتر کرده‌ایم. براساس این مشاهده می‌توان انتظار داشت که با افزایش قابلیت‌های حافظهٔ اتومات، خانواده زبان‌های قدرتمندتری را هم کشف کنیم. بعنوان مثال، شاید بتوان در طرح کلی شکل ۱-۳، از دو پشته، سه پشته، یک صف یا ابزار ذخیره‌سازی دیگری استفاده کرد. آیا با تغییر ابزار ذخیره‌سازی، نوع جدیدی از اتومات و در نتیجه خانواده زبان جدیدتری بوجود می‌آید؟ اتخاذ این روش، سؤالات متعدد و در عین حال نه چندان جالبی را مطرح می‌کند. بهتر و آموزنده‌تر

است که با طرح سؤالات کلی‌تر، مفهوم اتومات را در حدامکان بسط و گسترش دهیم. مثلاً ببینیم قدرتمندترین اتوماتا کدام هستند و چه محدودیت‌های محاسباتی دارند؟ به این ترتیب، مفهوم اساسی به نام **ماشین‌های تورینگ** و در نتیجه، تعریف دقیقی از ایده محاسبه مکانیکی یا الگوریتمی مطرح می‌شود.

مطالعه خود را با ارائه تعریفی صوری از ماشین‌های تورینگ آغاز کرده و سپس به بررسی مواردی می‌پردازیم که در حین انجام برخی برنامه‌های ساده رخ می‌دهد. در ادامه، استدلال می‌کنیم که هرچند مکانیزم ماشین‌های تورینگ هنوز به اندازه کافی رشد پیدا نکرده و در واقع در مراحل ابتدایی خود بسر می‌برد، مفهوم آن به قدری گسترده است که فرآیندهای بسیار پیچیده را هم تحت پوشش می‌گیرد. بحث در این مورد ما را به سمت معرفی **تز تورینگ** هدایت می‌کند. براساس این نظریه، تمامی فرآیندهای محاسباتی، حتی آنها که بوسیله کامپیوترهای پیشرفته امروزی انجام می‌شوند، بوسیله ماشین‌های تورینگ هم قابل انجام هستند.

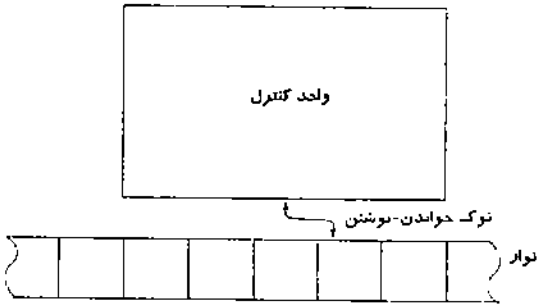
۹-۱ ماشین تورینگ استاندارد

با آنکه انواع مختلفی از اتوماتا با ابزار ذخیره‌سازی پیچیده و پیشرفته وجود دارند، حافظهٔ ذخیره‌سازی ماشین‌های تورینگ در عمل بسیار ساده است. این حافظه را می‌توان به صورت آرایه‌ای منفرد و تک‌بعدی از سلول‌هایی تصور کرد که هر یک قادر به نگهداری فقط یک سمبل هستند. این آرایه به صورت نامحدود در هر دو طرف راست و چپ گسترش پیدا کرده و به این دلیل، حجم نامحدودی از اطلاعات را در خود ذخیره می‌کند. همچنین، اطلاعات موجود در آرایه با هر ترتیبی، قابل خواندن و تغییر هستند. این دست ابزارهای ذخیره‌سازی، به دلیل شباهت ذاتی با نوارهای مغناطیسی در کامپیوترهای قدیمی، اصطلاحاً **نوار** خوانده می‌شوند.

تعریف ماشین تورینگ

یک ماشین تورینگ در واقع اتوماتی است که در آن، از نوار بعنوان حافظهٔ ذخیره‌سازی موقت استفاده شده باشد. این نوار به سلول‌هایی تقسیم شده و هر سلول قادر به نگهداری فقط یک سمبل است. به این نوار یک **هد خواندن- نوشتن** متصل است که می‌تواند به سمت راست یا چپ نوار حرکت کرده و در هر حرکت فقط یک سمبل را بخواند و بنویسد. برای تعریف تدریجی مدل کلی که از فصل اول در مورد اتومات در ذهن دارید، اتوماتی که فعلاً بعنوان ماشین تورینگ مطرح می‌کنیم فاقد فایل ورودی و هرنوع مکانیزم خروجی خاصی می‌باشد. هر نوع ورودی یا خروجی لازم بر روی خود نوار ماشین تولید خواهد شد. بعداً درک خواهیم کرد که انجام این اصلاحات در مدل کلی بخش ۱-۲ تغییرات چندانی، در آن ایجاد نخواهد کرد. حتی می‌توان بدون کوچکترین تأثیر بر نتایجی که بعداً خواهیم گرفت، فایل ورودی و یک نوع خروجی خاص را هم در این مدل گنجاند. اما چون این کار تعریف اتومات حاصل را تا حدودی مشکل می‌کند، فعلاً از انجام آن صرفنظر می‌کنیم.

تصویر شکل ۹-۱ تصویر شهودی از یک ماشین تورینگ را ارائه می‌کند. ایده بالا در قالب تعریف ۹-۱ مدون شده است.



شکل ۹-۱

تعریف ۹-۱

ماشین تورینگ M با

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

تعریف می‌شود که در آن،

- Q مجموعه حالات داخلی است،
- Σ الفبای ورودی است،
- Γ مجموعه متناهی از سمبل‌هایی به نام الفبای نوار است،
- δ تابع انتقال است،
- $\square \in \Gamma$ سمبل ویژه‌ای به نام خالی است،
- $q_0 \in Q$ حالت شروع است،
- $F \subseteq Q$ مجموعه حالت‌های پایانی است.

در تعریف ماشین‌های تورینگ، فرض می‌کنیم که $\Sigma \subseteq \Gamma - \{\square\}$؛ به این معنا که الفبای ورودی زیرمجموعه‌ای از الفبای نوار به استثنای فضای خالی است. به دلایلی که بعداً خواهیم گفت، □ها بعنوان سمبل ورودی استفاده نمی‌شوند. تابع انتقال δ به صورت زیر تعریف می‌شود.

$$\delta : Q \times \Gamma \to Q \times \Gamma \times \{L, R\}.$$

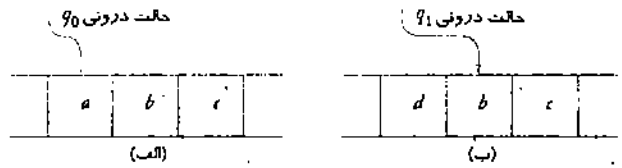
بطورکلی، δ یک تابع جزئی روی $Q \times \Gamma$ بوده و با تفسیر آن می‌توان به نحوه عملکرد ماشین تورینگ مورد نظر پی برد. استدلال‌های δ شامل حالت جاری واحد کنترل و سمبلی است که در حال حاضر از روی نوار خوانده می‌شود. در نتیجه، یک حالت جدید برای واحد کنترل، یک سمبل جدید

برای نوار بعنوان جایگزین سمبل قبلی و یک عمل حرکتی L یا R ایجاد می‌شود. براساس نشانهٔ حرکت می‌توان تعیین کرد که هد خواندن- نوشتن، پس از نوشتن سمبل جدید بر روی نوار، به اندازه یک سلول به کدامیک از طرفین، چپ یا راست حرکت می‌کند.

مثال ۹-۱

شکل ۹-۲ وضعیت را قبل و بعد از حرکت انجام شده بوسیله انتقال زیر نشان می‌دهد:

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



شکل ۹-۲ وضعیت (الف) قبل از حرکت و (ب) پس از حرکت

ماشین‌های تورینگ را می‌توان مدل نسبتاً ساده شده‌ای از یک کامپیوتر تصور کرد. آنها مجهز به یک واحد پردازشگر با حافظه متناهی بوده و در نوار خود، یک حافظهٔ ذخیره‌سازی ثانویه با ظرفیت نامحدود دارند. این کامپیوترها قادر به انجام دستورالعمل‌های بسیار محدودی هستند: به محض برخورد با یک سمبل بر روی نوار خود، با بررسی نتیجه، در مورد اقدام لازم تصمیم‌گیری می‌کنند. ماشین فقط قادر به بازنویسی سمبل جاری، تغییر حالت کنترل و جابجایی هد خواندن- نوشتن می‌باشد. شاید تصور کنید که این مجموعه کوچک از دستورالعمل‌ها برای انجام کارهای پیچیده به هیچ وجه کافی و قدرتمند نیستند. اما واقعیت برخلاف این است و ماشین‌های تورینگ، در مجموع، ابزار بسیار قدرتمندی محسوب می‌شوند. تابع انتقال δ وظیفه تعریف نحوه عملکرد این کامپیوتر را که غالباً "برنامه" ماشین خوانده می‌شود، برعهده دارد.

اتومات با در اختیار داشتن برخی اطلاعات روی نوار خود، از حالت شروع راه‌اندازی می‌شود. سپس با دنباله‌ای از مراحل که تحت کنترل تابع انتقال δ انجام می‌شود، به حرکت ادامه می‌دهد. در جریان این فرآیند، محتویات هر یک از سلول‌های روی نوار را می‌توان بارها وارسی و در صورت لزوم، اقدام به تغییر آن نمود. در نهایت، کل فرآیند متوقف شده و در اثر آن، ماشین تورینگ در حالت توقف قرار می‌گیرد. زمانی ماشین تورینگ توقف می‌کند که به یک پیکربندی برسد که هیچ δ ی برای آن تعریف نشده باشد. توجه داشته باشید که δ یک تابع جزئی است. در واقع، فرض می‌کنیم که برای هیچ کدام از حالت‌های پایانی، هیچ انتقالی تعریف نمی‌شود. بنابراین ماشین تورینگ با ورود به حالت پایانی، متوقف می‌شود.

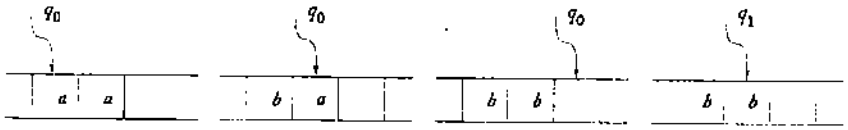
مثال ۹-۲

ماشین تورینگی را با فرضیات زیر در نظر بگیرید:

$$Q=\{q_0,q_1\},$$
$$\Sigma=\{a,b\},$$
$$\Gamma=\{a,b,\square\},$$
$$F=\{q_1\},$$
$$\delta(q_0,a)=(q_0,b,R),$$
$$\delta(q_0,b)=(q_0,b,R),$$
$$\delta(q_0,\square)=(q_1,\square,L).$$

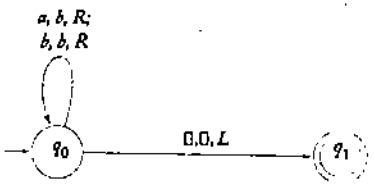
ماشین تورینگ فوق از حالت q_0 آغاز به کار می‌کند. در اینصورت، سمبل a، زیر هد خواندن- نوشتن قرار گرفته و بنابراین، می‌توان از قانون انتقال $\delta(q_0,a)=(q_0,b,R)$ برای جابجایی آن استفاده کرد. هد خواندن- نوشتن یک b را جایگزین a کرده و سپس، روی نوار به سمت راست حرکت می‌کند. ماشین در حالت q_0 باقی خواهد ماند. هریک از a های بعدی هم با یک b جایگزین می‌شوند، اما هیچکدام از b ها تغییری نخواهند کرد. ماشین پس از برخورد با اولین سلول خالی، به اندازه یک سلول به سمت چپ حرکت کرده و در حالت پایانی q_1 توقف خواهد کرد.

شکل ۹-۳ از چپ به راست مراحل مختلف فرآیند ایجاد یک پیکربندی شروع ساده را نمایش می‌دهد.



شکل ۹-۳

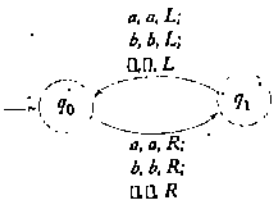
در اینجا هم می‌توان از گراف‌های انتقال برای نمایش ماشین‌های تورینگ استفاده کرد. با این تفاوت که، برای برچسب‌دار کردن یال‌های گراف از سه مؤلفه سمبل جاری نوار، سمبلی که جایگزین آن می‌شود و جهت حرکت هد خواندن- نوشتن، استفاده می‌شود. ماشین تورینگ مثال ۹-۲ بوسیله گراف انتقال شکل ۹-۴ نمایش داده شده است.



شکل ۹-۴

مثال ۹-۳

برای بررسی وقایع ماشین تورینگ شکل ۹-۵، یک حالت خاص را بررسی می‌کنیم. فرض کنید که نوار در ابتدای به کار ماشین، دارای $ab\ldots$ بوده و هد خواندن- نوشتن روی a قرار داشته باشد. ماشین a را می‌خواند، اما آنرا تغییر نمی‌دهد. سپس به حالت q_1 رفته و هد خواندن- نوشتن به سمت راست حرکت می‌کند. بنابراین اکنون بر روی b قرار دارد. این سمبل هم خوانده شده و بدون تغییر رها می‌شود. ماشین به حالت q_0 بازگشته و هد خواندن- نوشتن به سمت چپ حرکت می‌کند. حال دقیقاً به حالت اولیه ماشین برمی‌گردیم و توالی حرکات دوباره آغاز می‌شود. بر این اساس می‌توان گفت که ماشین با هر اطلاعات اولیه‌ای روی نوار خود، برای همیشه به کار ادامه خواهد داد، گرچه هد خواندن- نوشتن به تناوب به راست و چپ حرکت می‌کند، اما هیچ تغییری در نوار بوجود نمی‌آورد. این مثال نمونه‌ای از ماشین‌های تورینگی است که هیچگاه متوقف نمی‌شوند. برای نامگذاری این وضعیت، از همان اصطلاح آشنا در مبحث برنامه‌سازی استفاده کرده و می‌گوییم که ماشین تورینگ در یک حلقه نامتناهی قرار دارد.



شکل ۹-۵

■

به دلیل تنوع و تعدد تعریف ماشین‌های تورینگ، بهتر است که با خلاصه کردن خصوصیات اصلی مدل پیشنهادی خود، آنرا تحت عنوان **ماشین تورینگ استاندارد** مطرح کنیم:

۱. ماشین تورینگ، مجهز به نواری است که از دو طرف نامحدود است و به این دلیل، هر تعداد حرکت به سمت چپ و راست مجاز می‌باشد.

۲. ماشین تورینگ معین خوانده می‌شود، چون δ حداکثر یک حرکت را برای هر پیکربندی تعریف می‌کند.

۳. هیچ قابل ورودی خاصی وجود ندارد. فرض می‌کنیم که در هنگام راه‌اندازی روی نوار، اطلاعات خاصی وجود دارد. قسمتی از این محتویات نوار را می‌توان بعنوان ورودی در نظر گرفت. بعلاوه، هیچ وسیلهٔ خروجی خاصی هم وجود ندارد. پس از توقف ماشین، می‌توان قسمتی یا تمامی محتویات نوار را بعنوان خروجی در نظرگرفت.

از این قراردادها اساساً در فصل بعد استفاده خواهیم کرد. به این دلیل در فصل ۱۰، ضمن اشاره به مدل‌های دیگر ماشین‌های تورینگ، در مورد ارتباط آنها با مدل استاندارد خود بحث خواهیم کرد.

یک وجه اشتراک ماشین‌های تورینگ و pda ها آن است که بهترین و مناسب‌ترین روش برای نمایش توالی پیکربندی‌ها در هر دو، استفاده از وضعیت‌های لحظه‌ای می‌باشد. تمامی پیکربندی‌ها بطور کامل براساس حالت جاری واحد کنترل، محتویات نوار و موقعیت هد خواندن- نوشتن تعیین می‌شوند. این منظور، دنباله سمبل‌های را معرفی و استفاده خواهیم کرد که در آن،

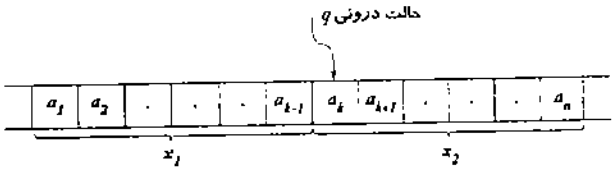
$$x_1 q x_2$$

یا

$$a_1 a_2 \cdots a_{k-1} q a_k a_{k+1} \cdots a_n$$

توصیف پیکربندی‌ها برای ماشینی است که در حالت q قرار داشته و نوار آن مشابه نوار شکل ۹-۴ باشد. سمبل‌های $a_1,\ldots,a_n$ معرف محتویات نوار و q، حالت واحد کنترل است. به صورت قراردادی محل هد خواندن- نوشتن روی سلولی است که سمبل آن بلافاصله پس از q قرار دارد.

در توصیف لحظه‌ای فقط مقدار متناهی اطلاعات در سمت راست و چپ هد خواندن- نوشتن ارائه می‌شود. براساس فرض، تمام بخش‌های تخصیص‌نیافته نوار را باید حاوی نشانهٔ خالی □ در نظر گرفت. این □ها، به دلیل تاثیرنداشتن، به صورت آشکار در پیکربندی نمایش داده نمی‌شوند. اما چنانچه موقعیت آن‌ها در بحث، نقش داشته باشد، می‌توان نشانهٔ خالی را در پیکربندی نمایش داد. بعنوان مثال، پیکربندی $q\square w$ به این معناست که هد خواندن- نوشتن روی سلولی قرار گرفته که بلافاصله در سمت چپ اولین سمبل w قرار داشته و این سلول حاوی یک نشانهٔ خالی □می‌باشد.



شکل ۹-۶

مثال ۹-۴

تصاویرشکل ۹-۳ متناظر با دنباله پیکربندی $q_0aa, bq_0a, bbq_0\square, bq_1b$ هستند.

انتقال از یک پیکربندی به پیکربندی دیگر بوسیله $\vdash$ نمایش داده می‌شود. بنابراین، اگر

$$\delta(q_1, c) = (q_2, e, R),$$

آنگاه حرکت

$$abq_1cd \vdash abeq_2d$$

در صورتی امکان‌پذیر است که حالت داخلی q_1 باشد، نوار حاوی $abcd$ بوده و هد خواندن-نوشتن

روی c قرار داشته باشد. نشانهٔ $\vdash$ معمولاً به معنای تعداد دلخواهی از حرکات است. در استدلال‌ها، از زیرنویس‌هایی از قبیل $M\vdash$ برای تمایز بین ماشین‌های مختلف استفاده می‌شود.

□

عملکرد ماشین تورینگ شکل ۹-۳ را می‌توان با

$$q_0aa \vdash bq_0a \vdash bbq_0\square \vdash bq_1b$$

یا

$$q_0aa \vdash bq_1b$$

نمایش داد.

□

پیش از ادامه بحث، بهتر است مشاهدات خود را به صورتی صوری ارائه کنیم.

تعریف ۹-۳

فرض کنید $M=(Q,\Sigma,\Gamma,\delta,q_0,\square,F)$ یک ماشین تورینگ باشد. آنگاه هر رشته $a_1\dots a_{k-1}q_1a_ka_{k+1}\dots a_n$ با فرض $a_i\in\Gamma$ و $q_1\in Q$ یکی از پیکربندهای M است. حرکت مفروض

$$a_1\dots a_{k-1}q_1a_ka_{k+1}\dots a_n \vdash a_1\dots a_{k-1}bq_2a_{k+1}\dots a_n$$

امکان‌پذیر خواهد بود اگر و تنها اگر

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

حرکت مفروض

$$a_1\dots a_{k-1}q_1a_ka_{k+1}\dots a_n \vdash a_1\dots q_2a_{k-1}ba_{k+1}\dots a_n$$

امکان‌پذیر خواهد بود اگر و تنها اگر

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

اگر برای هر q_j و a که در آن، $\delta(q_j,a)$ تعریف نشده باشد،

$$x_1q_ix_2 \vdash y_1q_jay_2,$$

می‌گوییم M با آغاز از پیکربندی شروع $x_1q_ix_2$، توقف می‌کند. دنباله پیکربندی‌هایی که به یک حالت توقف ختم می‌شوند، محاسبه نامیده می‌شود.

مثال ۹-۳ یک نمونه از مواردی بود که ماشین‌های تورینگ هرگز متوقف نشده و در دام حلقه بی‌نهایت گرفتار می‌شوند که امکان رهایی از آن وجود ندارد. این وضعیت یکی از وضعیت‌های مهم در بحث ماشین‌های تورینگ بوده و به این دلیل، از فرم خاص زیر

$$x_1qx_2 \vdash \infty,$$

برای نمایش آن استفاده می‌کنیم؛ فرم فوق به این معناست که ماشین با راه‌اندازی از پیکربندی شروع x_1qx_2، وارد یک حلقه شده و هرگز توقف نمی‌کند.

ماشین‌های تورینگ در نقش پذیرنده‌های زبان

می‌توان به ترتیب زیر، از ماشین‌های تورینگ بعنوان پذیرنده هم استفاده کرد: رشته مفروض w روی نوار نوشته شده و □ها هم در قسمت‌های استفاده نشده، قرار می‌گیرند. ماشین از حالت شروع q_0 راه‌اندازی شده و هد خواندن- نوشتن روی چپ‌ترین سمبل w قرار می‌گیرد. اگر پس از دنباله‌ای از حرکات، ماشین تورینگ وارد یکی از حالت‌های پایانی و در نتیجه توقف کند، آنگاه w پذیرش می‌شود.

تعریف ۹-۳

فرض کنید $M=(Q,\Sigma,\Gamma,\delta,q_0,\square,F)$ یک ماشین تورینگ باشد. آنگاه زبان

$$L(M)=\{w\in\Sigma^+ : q_f\in F, x_1,x_2\in\Gamma^* \text{ برای برخی از } q_0w \vdash x_1q_fx_2\}$$

بوسیله M پذیرش می‌شود.

براساس این تعریف، ورودی w روی نوار نوشته شده و □ها در هر یک از طرفین آن قرار می‌گیرند. اینک به راحتی می‌توان دلیل غیرمجاز بودن استفاده از □ها بعنوان ورودی را بیان کرد: چون با حذف آنها مطمئن می‌شویم که تمامی ورودی در یک ناحیه مناسب تعریف شده از نوار که بوسیلهٔ □ها در چپ و راست آن احاطه شده است، محدود می‌شود. در صورت نادیده‌گرفتن این قرارداد، محدوده وجود ورودی نامشخص بوده و بنابراین، ماشین نمی‌داند که باید در کجا به دنبال ورودی بگردد. بنابراین، هر تعداد □ هم که توسط ماشین پیدا شود، بازهم این امکان وجود دارد که در جایی روی نوار با یک ورودی □ دیگر مواجه شود.

براساس تعریف ۹-۳، وقتی $w\in L(M)$، نمی‌توان هیچ نتیجه‌ای در مورد هیچ‌کدام از ورودی‌های دیگر مطرح کرد. اما اگر w عضو $L(M)$ نباشد، یکی از دو حالت زیر اتفاق می‌افتد: یا ماشین در یک حالت غیرپایانی متوقف می‌شود و یا به یک حلقه بی‌نهایت وارد شده و هرگز متوقف نمی‌شود. بنابراین، هر رشته‌ای که باعث توقف M نشود، عضو $L(M)$ نمی‌باشد.

برای $\Sigma=\{0,1\}$، ماشین تورینگی طراحی کنید که زبان تعریف شده بوسیله عبارت منظم ^{*00} را پذیرش کند.

این مثال یکی از تمرین‌های ساده در برنامه‌ریزی ماشین‌های تورینگ است. با شروع از انتهای سمت چپ ورودی، تک تک سمبل‌ها را خوانده و به این ترتیب بررسی می‌کنیم: اگر سمبل خوانده شده 0 باشد، همچنان به حرکت خود به سمت راست ادامه می‌دهیم. اگر پیش از روبرو شدن با 0 به یک □ برسیم، به پایان رشته رسیده‌ایم و بنابراین، آنرا پذیرش می‌کنیم. اگر در جایی از رشته با 1 برخورد کردیم، رشته در $L(^*00)$ وجود نخواهد داشت و در یک حالت غیرپایانی متوقف می‌شویم.

برای بررسی محاسبه، کافی است دو حالت داخلی $Q=\{q_0,q_1\}$ و یک حالت پایانی $F=\{q_1\}$ را در نظر بگیریم. تابع انتقال را به صورت

$$\delta(q_0,0)=(q_0,0,R),$$
$$\delta(q_0,\square)=(q_1,\square,R),$$

در نظر می‌گیریم. زمانیکه یک 0 زیر هد خواندن- نوشتن قرار داشته باشد، هد به سمت راست حرکت خواهد کرد. به محض خواندن یک سمبل 1، از آنجایی‌که $\delta(q_0,1)$ تعریف نشده است، ماشین در حالت غیرپایانی q_0 متوقف می‌شود. توجه داشته باشید که اگر ماشین تورینگ در حالت شروع q_0 از روی □ آغاز به کار کند، باز هم در یک حالت پایانی متوقف خواهد شد. هرچند می‌توان این وضعیت را به معنای پذیرش λ تلقی کرد، به دلایل فنی از ذکر رشته تهی در تعریف ۹-۳ صرفنظر کرده‌ایم.

■

با افزایش پیچیدگی زبان‌ها، شناسایی آنها مشکل‌تر می‌شود. بدلیل آنکه در ماشین‌های تورینگ از مجموعه دستورالعمل‌های ابتدایی و ساده‌ای استفاده می‌شود، برنامه‌ریزی برای محاسباتی که بوسیله یک زبان سطح بالاتر به راحتی قابل انجام است، معمولاً در ماشین‌های تورینگ به سختی انجام می‌شود. با این وجود، این مشکل قابل حل بوده و همانطور که در مثال بعد می‌بینیم، به راحتی از نظر مفهومی قابل درک است.

مثال ۹-۹

برای $\Sigma=\{a,b\}$، ماشین تورینگی را طراحی کنید که

$$L=\{a^nb^n : n\geq 1\}$$

را پذیرش کند. از نظر شهودی، مسأله را به صورت زیر حل می‌کنیم. با آغاز از چپ‌ترین a، آنرا بررسی و با سمبلی مثل x جایگزین می‌کنیم. سپس اجازه می‌دهیم که هد خواندن- نوشتن با حرکت به سمت راست، چپ‌ترین b را هم پیدا کند. سپس آنرا با سمبل دیگری مثل y جایگزین می‌کنیم. اینک دوباره به سمت چپ و به سراغ چپ‌ترین a رفته، آنرا با یک x جایگزین می‌کنیم. همچنین به سراغ

چپ‌ترین b رفته و آنرا با y جایگزین می‌کنیم و به همین ترتیب تا به آخر ادامه می‌دهیم. با ادامه این حرکت پاندولی، تک تک a ها را با b نظیر آن تطابق می‌دهیم. اگر هیچ a یا b دیگری باقی نماند، آنگاه رشته حتماً عضو L خواهد بود.

با صرفنظر از جزئیات، به جواب کامل $Q=\{q_0,q_1,q_2,q_3,q_4\}$، $\Sigma=\{a,b\}$، $F=\{q_4\}$ و $\Gamma=\{a,b,x,y,\square\}$ می‌رسیم. انتقالات را می‌توان در چند بخش مطرح کرد. مجموعه

$$\delta(q_0,a)=(q_1,x,R),$$
$$\delta(q_1,a)=(q_1,a,R),$$
$$\delta(q_1,y)=(q_1,y,R),$$
$$\delta(q_1,b)=(q_2,y,L)$$

چپ‌ترین a را با یک x جایگزین کرده و باعث می‌شود که هد خواندن- نوشتن، ضمن حرکت به طرف راست و یافتن اولین b، آنرا با یک y جایگزین کند. پس از نوشتن y، ماشین به حالت q_2 وارد می‌شود؛ به این معنا که یک a دقیقاً با یک b جفت شده است.

مجموعه بعدی انتقالات، ضمن معکوس کردن جهت، امکان برخورد با یک x را فراهم کرده، هد خواندن- نوشتن را دوباره روی چپ‌ترین a قرار داده و کنترل را به حالت شروع بازمی‌گرداند.

$$\delta(q_2,y)=(q_2,y,L),$$
$$\delta(q_2,a)=(q_2,a,L),$$
$$\delta(q_2,x)=(q_0,x,R).$$

اینک به حالت شروع q_0 بازگشته و آماده اقدام در مورد a و b بعدی هستیم.

در پایان دور اول از این قسمت از محاسبه، ماشین باید محاسبه تقریبی

$$q_0aa\cdots abb\cdots b \overset{*}{\mapsto} xq_0a\cdots ayb\cdots b,$$

را نیز انجام داده باشد و بنابراین حتماً یک a با یک b تطابق پیدا کرده است. پس از دور دوم، محاسبه جزئی

$$q_0aa\cdots abb\cdots b \overset{*}{\mapsto} xxq_0\cdots ayy\cdots b,$$

کامل شده و الی آخر؛ به این معنا که فرآیند تطابق به درستی انجام شده است.

در صورتیکه ورودی همان رشته a^nb^n باشد، کار بازنویسی به این ترتیب ادامه پیدا کرده و فقط در صورتی متوقف می‌شود که تمامی a ها پاک شده باشند. هد خواندن- نوشتن در جستجوی چپ‌ترین a به سمت چپ حرکت کرده و ماشین در حالت q_2 قرار می‌گیرد. به محض برخورد با یک x، جهت معکوس می‌شود تا به a برسیم. اما در اینصورت، به جای a، با یک y برخورد خواهیم کرد. در پایان، بررسی نهایی صورت می‌گیرد تا مطمئن شویم که تمامی a ها و b ها بطور کامل جایگذاری شده‌اند

(به این ترتیب، می‌توان ورودی‌هایی را ردیابی کرد که در هر یک از آنها، یک a پس از یک b قرار دارد). این کار به صورت زیر انجام می‌شود:

$$\delta(q_0, y) = (q_3, y, R),$$

$$\delta(q_3, y) = (q_3, y, R),$$

$$\delta(q_3, \square) = (q_1, \square, R).$$

چنانچه رشته‌ای را وارد کنیم که در این زبان وجود نداشته باشد، محاسبه در یک حالت غیرپایانی توقف خواهد کرد. بعنوان مثال، اگر رشته $a^n b^m$ با فرض $n > m$ را در ماشین قرار دهیم، ماشین بلاخره در حالت q_1 با یک $\square$ مواجه شده و چون هیچ انتقالی برای این حالت در نظر گرفته نشده، متوقف خواهد شد. هر ورودی دیگر غیرعضو در این زبان هم منجر به توقف ماشین در یک حالت غیرپایانی می‌شود (مراجعه شود به تمرین ۳ انتهای همین بخش).

بعنوان مثال، ورودی $aabb$ پیکربندی‌های متوالی زیر را ایجاد می‌کند:

$$q_0 aabb \mapsto x q_1 abb \mapsto xaq_1 bb \mapsto xq_2 ayb$$

$$\mapsto q_2 xayb \mapsto xq_0 ayb \mapsto xxq_1 yb$$

$$\mapsto xxyq_1 b \mapsto xxq_2 yy \mapsto xq_2 xyy$$

$$\mapsto xxq_0 yy \mapsto xxyq_3 y \mapsto xxyyq_3 \square$$

$$\mapsto xxyy \square q_4 \square.$$

به اینجا که می‌رسیم، ماشین تورینگ در یک حالت پایانی متوقف شده و بنابراین رشته $aabb$ پذیرش می‌شود.

بهتر است که این برنامه را با چند رشته دیگر عضو L و همچنین با چند رشته غیر عضو L ادامه دهید.

□

مثال ۹–۸

ماشین تورینگی طراحی کنید که

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

را پذیرش کند. آنچه در مثال ۹–۷ مطرح کردیم، در این مورد هم قابل استفاده است. ضمن جایگزینی ترتیبی x، y و z با a، b و c، آنها را با هم تطابق می‌دهیم. در پایان، بررسی می‌کنیم که تمامی سمبل‌های اصلی حتماً بازنویسی شده باشند. بدلیل آنکه این مثال از نظر مفهومی تعمیم یافته ساده‌ای از مثال قبل محسوب می‌شود، لزومی به نوشتن برنامه مربوط به آن وجود ندارد. اثبات این مثال ساده را به

دلیل طولانی بودن به خواننده واگذار می‌کنیم. توجه داشته باشید که حتی اگر $\{a^n b^n\}$ یک زبان مستقل‌ازمتن باشد، ولی $\{a^n b^n c^n\}$ که این‌چنین نیست، می‌توان آنرا هم با ساختاری مشابه در ماشین تورینگ پذیرش کرد.

براساس مثال فوق می‌توان نتیجه گرفت که ماشین‌های تورینگ قادر به شناخت برخی از زبان‌های غیر مستقل‌ازمتن هستند و این خود اولین نشانه قوی‌تر بودن ماشین‌های تورینگ در مقایسه با اتوماتای پشته‌ای است.

تا به حال دلایل کافی برای مطالعه مترجم‌ها وجود نداشته است. همچنین برای بررسی نظریه زبان‌ها هم، مطالعه پذیرنده‌ها کفایت می‌کرد. اما همانطور که خواهیم دید، ماشین‌های تورینگ نه تنها به اندازه موضوع پذیرنده‌های زبان جالب هستند، بلکه مدل انتزاعی و ساده‌ای از تمامی کامپیوترهای رقمی در اختیار ما قرار می‌دهند. اگر هدف و وظیفه اصلی یک کامپیوتر را تبدیل ورودی به خروجی در نظر بگیریم، آنها را در واقع می‌توان نوعی مترجم محسوب کرد. برای موفقیت در مدل‌سازی کامپیوترها بوسیله ماشین‌های تورینگ، باید این جنبه را به دقت مورد مطالعه و بررسی قرار دهیم.

در تمامی محاسبات، ورودی‌ها همان سمبل‌هایی به غیر از $\square$ هستند که در زمان راه‌اندازی روی نوار قرار دارند. خروجی آن چیزی است که در نتیجه محاسبه روی نوار قرار می‌گیرد. بنابراین، می‌توان مترجم M را یک ماشین تورینگ برای پیاده‌سازی تابع f با تعریف

$$\hat{w} = f(w),$$

در نظر گرفت، مشروط به اینکه به ازای حالت پایانی مفروض q_f داشته باشیم:

$$q_0 w \overset{*}{\mapsto}_M q_f \hat{w}.$$

تابع f با دامنه D محاسبه‌پذیر توسط تورینگ یا فقط محاسبه‌پذیر گفته می‌شود اگر ماشین تورینگ مفروض $M = (Q, \Sigma, \Gamma, \delta, q_0, \square, F)$ وجود داشته باشد که برای همهٔ $w \in D$،

$$q_0 w \overset{*}{\mapsto}_M q_f f(w), \quad q_f \in F.$$

در ادامه مشاهده خواهیم کرد که تمامی توابع ریاضی معمولی، بوسیلهٔ تورینگ محاسبه‌پذیر بوده و

میزان پیچیدگی آنها هم کوچکترین تأثیری بر این امر نخواهد داشت. کار را با بررسی برخی عملگرهای ساده از قبیل جمع و مقایسه ریاضی آغاز می‌کنیم.

مثال ۹–۹

با داشتن دو عدد صحیح و مثبت x و y، یک ماشین تورینگ برای محاسبه $x+y$ طراحی کنید.

ابتدا باید قراردادی برای نمایش اعداد صحیح مثبت ایجاد کنیم. برای آسانی در محاسبات، از مجموعه سمبل‌های یکتایی استفاده می‌کنیم که براساس آن، هر عدد صحیح مثبت x بوسیله $w(x)\in\{1\}^+$ نمایش داده می‌شود، بطوریکه

$$|w(x)|=x$$

همچنین باید در مورد نحوه قرارگیری اولیه x و y روی نوار و نحوه نمایش حاصل‌جمع آنها در پایان محاسبه، تصمیم‌گیری کنیم. فرض می‌کنیم که $w(x)$ و $w(y)$ به صورت دنباله سمبل‌های یکتایی از 1ها روی نوار قرار داشته، بوسیله یک 0 از هم جدا شده و هد خواندن- نوشتن روی چپ‌ترین علامت $w(x)$ قرار دارد. پس از محاسبه، $w(x+y)$ روی نوار قرار گرفته و فقط یک 0 بعد از آن مشاهده می‌شود و هد خواندن- نوشتن در انتهای سمت چپ نتیجه قرار خواهد داشت. بنابراین، در واقع می‌خواهیم ماشین تورینگی را برای انجام محاسبه

$$q_0w(x)0w(y)\mapsto q_fw(x+y)0,$$

طراحی کنیم که در آن، q_f حالت پایانی باشد. اینکار تقریباً به راحتی قابل انجام است. فقط کافی است پس از انتقال 0 جداکننده به انتهای راست $w(y)$، با ادغام داده‌های دو رشته، اقدام به جمع کردن آنها کنیم. به این منظور، با فرض $Q=\{q_0,q_1,q_2,q_3,q_4\}$، $F=\{q_4\}$ و

$$\begin{aligned}
&\delta(q_0,1)=(q_0,1,R),\\
&\delta(q_0,0)=(q_1,1,R),\\
&\delta(q_1,1)=(q_1,1,R),\\
&\delta(q_1,\square)=(q_2,\square,L),\\
&\delta(q_2,1)=(q_3,0,L),\\
&\delta(q_3,1)=(q_3,1,L),\\
&\delta(q_3,\square)=(q_4,\square,R),
\end{aligned}$$

اقدام به ساخت $M=(Q,\Sigma,\Gamma,\delta,q_0,\square,F)$ می‌کنیم.

توجه داشته باشید که برای انتقال 0 به سمت راست، موقتاً یک 1 اضافی ایجاد می‌کنیم. به این منظور، ماشین باید در حالت q_1 قرار گیرد. برای حذف 1 در پایان محاسبه، از انتقال $\delta(q_2,1)=(q_3,0,L)$ استفاده می‌کنیم. می‌توان به کمک دنباله‌ای از پیکربندی‌ها برای جمع 11 با 11، به صورت زیر آن را بررسی کرد:

$$\begin{aligned}
q_0\text{11011}&\mapsto \text{1}q_0\text{1011}\mapsto \text{11}q_0\text{011}\\
&\mapsto \text{111}q_1\text{11}\mapsto \text{1111}q_1\text{1}\mapsto \text{11111}q_1\square\\
&\mapsto \text{1111}q_2\text{1}\mapsto \text{111}q_3\text{10}\\
&\mapsto q_3\square\text{11110}\mapsto q_4\text{11110}.
\end{aligned}$$

کار با دنباله سمبل‌های یکتایی 1، هرچند در محاسبات مشکل به نظر می‌رسد، ولی در برنامه‌ریزی ماشین‌های تورینگ بسیار سودمند است. در قیاس با دیگر روش‌های نمایش از قبیل دودویی یا اعشاری، برنامه‌های حاصل از این روش کوتاه‌تر و ساده‌تر هستند.

■

اضافه کردن اعداد، یکی از اصلی‌ترین عملیاتی است که در هر کامپیوتری انجام شده و بعلاوه، در ترکیب دستورالعمل‌های پیچیده‌تر نقش دارد. بطور نمونه برای عملگرهای اساسی دیگر می‌توان به کپی‌کردن رشته‌ها و مقایسه‌های ساده اشاره کرد. اینگونه عملگرها به کمک یک ماشین تورینگ به راحتی قابل انجام است.

مثال ۹–۱۰

ماشین تورینگی طراحی کنید که رشته‌های از 1 را کپی کند. به بیان دقیق‌تر، ماشینی ارائه دهید که برای هر $w\in\{1\}^+$،

$$q_0w\mapsto q_fww,$$

را محاسبه کند.

برای حل مسأله، فرآیند شهودی زیر را پیاده می‌کنیم:

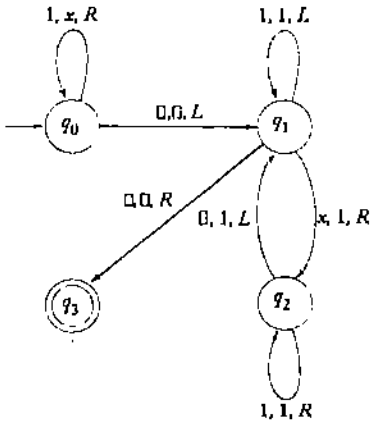
۱. هریک از 1ها را با یک x جایگزین می‌کنیم.

۲. راست‌ترین x را پیدا کرده و آنرا با 1 جایگزین می‌کنیم.

۳. به انتهای سمت راست ناحیه غیر از □ یعنی اولین □ سمت راست فعلی رفته و در آنجا یک 1 ایجاد می‌کنیم.

۴. مراحل ۲ و۳ را آنقدر تکرار می‌کنیم تا هیچ x دیگری وجود نداشته باشد.

جواب سؤال در گراف انتقال شکل ۹-۷ ارائه شده است. شاید نتوان در همان ابتدای کار در مورد صحت پاسخ تصمیم‌گیری کرد. از اینرو، برنامه را با رشته ساده 11 دنبال می‌کنیم. محاسبه مربوطه به صورت زیر خواهد بود:



شکل ۹-۷

$$\begin{aligned} q_0 11 &\vdash x q_0 1 \vdash x x q_0 \square \vdash x q_1 x \\ &\vdash x 1 q_2 \square \vdash x q_1 11 \vdash q_1 x 11 \\ &\vdash 1 q_2 11 \vdash 11 q_2 1 \vdash 111 q_2 \square \\ &\vdash 11 q_1 11 \vdash 1 q_1 111 \\ &\vdash q_1 1111 \vdash q_1 \square 1111 \vdash q_3 1111. \end{aligned}$$

■

مثال ۹-۱۱

x و y را دو عدد صحیح مثبت در نظر بگیرید که بصورت دنباله سمبل‌های یکتایی 1ها نمایش داده شده‌اند. ماشین تورینگی بسازید که اگر $x \geq y$، در یک حالت پایانی q_y متوقف شده و اگر $x < y$، در یک حالت غیرپایانی q_n متوقف خواهد شد. به بیان دیگر، ماشین باید محاسبه زیر را انجام دهد.

$$q_0 w(x) 0 w(y) \vdash^* q_y w(x) 0 w(y) \qquad \text{و اگر } x \geq y$$

$$q_0 w(x) 0 w(y) \vdash^* q_n w(x) 0 w(y) \qquad \text{و اگر } x < y$$

برای حل مسأله، می‌توان با کمی اصلاحات از ایده مثال ۹-۷ استفاده کرد. یعنی، به جای تطابق a ها و b ها، هریک از 1های موجود در طرف چپ 0 جداکننده را با 1 واقع در طرف راست تطابق دهیم. در پایان کار تطابق، بسته به اینکه $x > y$ یا $x < y$، یکی از دو حالت

$$xx \cdots 110 xx \cdots x \square$$

یا

$$xx \cdots xx\, 0 xx \cdots x\, 11 \square,$$

را روی نوار خواهیم داشت. در حالت اول، در هنگام تطابق 1 بعدی، در سمت راست فضای کار با □ روبرو خواهیم شد. این امر را می‌توان دلیلی برای رفتن به حالت q_y تلقی کرد. در حالت دوم، حتی پس از جایگزینی تمامی 1های واقع در سمت چپ، بازهم یک 1 در سمت راست وجود خواهد داشت. از این نکته برای رفتن به حالت q_n استفاده می‌کنیم. ادامه برنامه و تکمیل پاسخ به راحتی قابل انجام بوده و به همین دلیل آنرا بعنوان تمرین ذکر کرده‌ایم.

در این مثال با یک نکته مهم مواجه می‌شویم: یک ماشین تورینگ را می‌توان به گونه‌ای برنامه‌ریزی کرد که از مقایسه‌های ریاضی در تصمیم‌گیری‌های خود استفاده کند. این نوع تصمیم‌گیری‌های ساده در زبان ماشین کامپیوترهایی استفاده می‌شود که بسته به نتیجه یک عملیات جبری، دنباله‌های دستورات متفاوتی اجرا می‌شوند.

■

تمرین‌ها

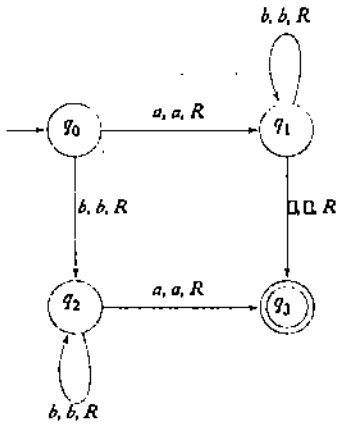
۱. *** – بوسیله یکی از زبان‌های برنامه‌سازی سطح بالا، یک شبیه‌ساز ماشین تورینگ بنویسید. این شبیه‌سازها، برنامه تورینگ را به همراه یک پیکربندی اولیه را به عنوان ورودی دیافت کرد، نتیجه محاسبه را به عنوان خروجی تولید می‌کنند.

۲. یک ماشین تورینگ با حداکثر سه حالت طراحی کنید که زبان $L(a(a+b)^*)$ را پذیرش کند. فرض کنید که $\Sigma = \{a,b\}$ باشد. آیا این‌کار را می‌توان با یک ماشین دوحالتی هم انجام داد؟

۳. ماشین تورینگ مثال ۹-۷ برای ورودی‌های aba و $aaabbbb$، چگونه عمل خواهد کرد؟

۴. آیا ورودی وجود دارد که ماشین تورینگ مثال ۹-۷ را به یک حلقه بی‌نهایت وارد کند؟

۵. یک ماشین تورینگ با گراف انتقال مشابه شکل زیر، چه زبانی را می‌پذیرد؟



۶. اگر در مثال ۹-۱۰، رشته w حاوی سمبلی غیر از 1 باشد، چه اتفاقی می‌افتد؟

۷. ماشین‌های تورینگی را طراحی کنید که زبان‌های زیر را روی الفبای $\{a,b\}$ پذیرش کند:

الف) $L = L(aba^*b)$.

ب) $L = \{w \mid w$ زوج است$\}$.

ج) $L = \{w \mid w$ ایکی از ضرایب ۳ است$\}$.

د) $L = \{a^n b^m : n \geq 1, n \neq m\}$.

ه) $L = \{w : n_a(w) = n_b(w)\}$.

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

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

ح) $L = \{a^n b^{2n} : n \geq 1\}$.

در هر یک از مسائل، δ را با جزئیات کامل بنویسید. سپس، با ارائه مثال‌های آزمایشی صحت پاسخ‌های خود را بررسی کنید.

۸. ماشین تورینگی طراحی کنید که زبان زیر را پذیرش کند.

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

۹. ماشین تورینگی را برای محاسبه تابع

$$f(w) = w^R,$$

بسازید که برای آن، $w \in \{0,1\}^+$ می‌باشد.

۱۰. ماشین تورینگی طراحی کنید که وسط رشته‌ای با طول زوج را پیدا کند. در حالت خاص، اگر $w = a_1 a_2 \ldots a_n a_{n+1} \ldots a_{2n}$ و $a_i \in \Sigma$، ماشین تورینگ باید $\hat{w} = a_1 a_2 \ldots a_n c a_{n+1} \ldots a_{2n}$ را تولید کند که در آن $c \in \Gamma - \Sigma$.

۱۱. ماشین‌های تورینگی را برای محاسبه توابع زیر طراحی کنید که در آن، x و y اعداد صحیح مثبتی هستند که به صورت دنباله‌ای از سمبل‌های 1 نمایش داده می‌شوند.

الف) $f(x) = 3x$.

ب) $f(x,y) = \begin{cases} x-y & x > y \\ 0 & x \leq y \end{cases}$

ج) $f(x,y) = 2x + 3y$.

د) $f(x) \begin{cases} \frac{x}{2} & \text{اگر } x \text{ زوج باشد} \\ \frac{x+1}{2} & \text{اگر } x \text{ فرد باشد} \end{cases}$

ه) $f(x) = x \bmod 5$.

و) $f(x) = \left[\frac{x}{2}\right]$، که در آن $\left[\frac{x}{2}\right]$ به بزرگترین عدد صحیح کوچکتر یا مساوی $\left[\frac{x}{2}\right]$ دلالت می‌کند.

۱۲. یک ماشین تورینگ با فرض $\Gamma = \{0,1,\square\}$ طراحی کنید که وقتی از هر سلول حاوی یک $\square$ یا 1 راه‌اندازی می‌شود، متوقف شود اگر و تنها اگر در مکانی از نوار یک 0 وجود داشته باشد.

۱۳. راه‌حل کامل مثال ۹-۸ را بنویسید.

۱۴. فرض کنید که ورودی 111 در اختیار ماشین تورینگ مثال ۹-۱۰ قرار بگیرد. دنباله پیکربندی‌های مربوطه را بنویسید. اگر نوار این ماشین با 110 راه‌اندازی شود، چه اتفاقی می‌افتد؟

۱۵. با ارائه استدلال‌های کافی، نشان دهید که اگر ماشین تورینگ مثال ۹-۱۰ واقعاً محاسبه مورد نظر را انجام دهد، چه اتفاقی می‌افتد؟

۱۶. تمام جزئیات مثال ۹-۱۱ را کامل کنید.

۱۷. فرض کنید که در مثال ۹-۹ بخواهیم x و y را به صورت دودویی نمایش دهیم. برنامه‌ای برای ماشین تورینگ بنویسید که بتواند محاسبه مورد نظر را در این نمایش انجام دهد.

۱۸. مثال ۹-۹ را با فرض نمایش x و y به صورت اعداد اعشاری حل کنید.

۱۹. شاید متوجه شده باشید که در تمامی مثال‌های این بخش تنها یک حالت پایانی وجود داشت. آیا بطورکلی می‌توان گفت که برای همه ماشین‌های تورینگ، یک ماشین دیگر با تنها یک حالت پایانی وجود دارد که همان زبان را می‌پذیرد؟

۲۰. بوسیله تعریف ۹-۲ می‌توان رشته‌های تهی را از تمامی زبان‌های پذیرش شده توسط یک ماشین تورینگ حذف کرد. این تعریف را به نحوی اصلاح کنید تا بتوان زبان‌های دارای λ را پذیرش کند.

۹-۲ ترکیب ماشین‌های تورینگ برای کارهای پیچیده

تا به اینجا دیدیم که چگونه می‌توان برخی عملیات‌های مهمی را که در تمامی کامپیوترها انجام می‌شوند، بر روی ماشین‌های تورینگ نیز اجرا کرد. بدلیل آنکه این دست عملیات‌های اولیه، اجزاء سازنده دستورات پیچیده‌تر در کامپیوترهای دیجیتالی محسوب می‌شوند، نحوه ادغام این عملیات‌های پایه‌ای را در ماشین‌های تورینگ بررسی خواهیم کرد. برای نمایش نحوه ترکیب ماشین‌های تورینگ، از یکی از روش‌های متعارف در برنامه‌سازی استفاده می‌کنیم. کار را با یک توصیف سطح بالا آغاز کرده و آنرا مکرراً اصلاح می‌کنیم تا برنامه عملاً به زبان موردنظر تبدیل شود. برای توصیف ماشین‌های تورینگ در سطحی بالا، می‌توان از روش‌های مختلفی استفاده کرد. نمودارهای بلوکی و شبه‌کدها دو روش متعارفی هستند که به دفعات در مثال‌های بعدی مورد استفاده قرار خواهند گرفت. در نمودارهای بلوکی، محاسبات در کادرهایی با کارکرد مشخص و جزئیات داخلی نامشخص انجام می‌شود. استفاده از این کاردها، ضمناً به معنای امکان‌پذیر بودن ساخت آنها می‌باشد. بعنوان اولین نمونه، ماشین‌های مثال‌های ۹-۹ و ۹-۱۱ را با هم ترکیب می‌کنیم.

مثال ۹-۱۲

ماشین تورینگی را طراحی کنید که تابع زیر را محاسبه کند.

$$f(x,y)=\begin{cases} x+y & \text{اگر } x \geq y \\ 0 & \text{اگر } x < y \end{cases}$$

برای ساده کردن بحث، x و y را اعداد صحیح مثبت نمایش داده شده بوسیله دنبالهٔ اها فرض می‌کنیم. مقدار صفر بوسیله 0 نمایش داده شده و بقیه نوار □ است.

برای نمایش محاسبه $f(x,y)$ در یک سطح بالا، می‌توان از شکل ۹-۸ استفاده کرد. براساس این شکل، ابتدا با استفاده از یک ماشین مقایسه‌گر مشابه مثال ۹-۱۱، درستی یا نادرستی $x \geq y$ را مشخص می‌کنیم. در صورت درستی مقایسه‌گر، یک پیام شروع برای جمع‌کننده ارسال کرده و سپس $x+y$ را محاسبه می‌کند. در غیر اینصورت، یک برنامه پاک‌کننده آغاز به کار کرده و هریک از اها را به □ تبدیل می‌کند.

در اغلب مباحث بعدی، از این دست بسته‌های که داخل آنها نامشخص است و به صورت سطح بالا تعریف می‌شوند برای نمایش ماشین‌های تورینگ استفاده می‌کنیم. در مقایسه با استفاده از مجموعه‌های بزرگ δهای متناظر، روش اخیر مطمئناً سریعتر و روشن‌تر است. اما پیش از پذیرفتن این روش نمایش سطح بالا، باید آنرا اثبات کنیم. مثلاً منظور دقیق خود را از عبارت "مقایسه‌گر، یک پیام شروع برای جمع‌کننده ارسال می‌کند" بیان کنیم. بدلیل آنکه نمی‌توان از تعریف ۹-۱ برای پاسخگویی به این پرسش استفاده کرد، روش زیر را ارائه می‌کنیم.

برنامه مقایسه‌گر C، مطابق مثال ۹-۱۱، یعنی با استفاده از ماشین تورینگی که حالت‌های آن بوسیله Q مشخص شده باشند، انجام می‌شود. حالت‌های جمع‌کننده هم مطابق مثال ۹-۹، بوسیله A مشخص می‌شوند. در مورد پاک‌کننده E هم، ماشین تورینگی را می‌سازیم که حالت‌های آن با E مشخص شده باشد. محاسباتی که باید بوسیله C انجام شوند شامل

$$q_{C,0}w(x)0w(y) \vdash q_{A,0}w(x)0w(y) \qquad \text{اگر } x \geq y$$

$$q_{C,0}w(x)0w(y) \vdash q_{E,0}w(x)0w(y) \qquad \text{اگر } x < y$$

می‌باشد. اگر $q_{A,0}$ و $q_{E,0}$ را به ترتیب، بعنوان حالتهای شروع A و E در نظر بگیریم، مشاهده می‌کنیم که C با A یا E آغاز می‌شود.

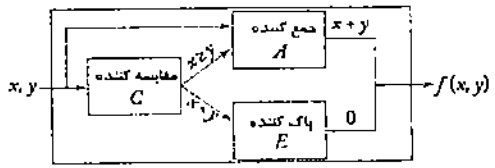
محاسباتی که بوسیله جمع‌کننده A انجام می‌شوند

$$q_{A,0}w(x)0w(y) \vdash q_{A,f}w(x+y)0,$$

و محاسباتی که بوسیله پاک‌کننده E انجام می‌شود،

$$q_{E,0}w(x)0w(y) \vdash q_{E,f}0,$$

خواهد بود. در نتیجه، ماشین تورینگی بوجود می‌آید که عمل C و A و E را مطابق شکل ۹-۸، با هم ترکیب می‌کند.



شکل ۹-۸

■

شبه‌کد، یکی دیگر از روش‌های مفید و سطح بالا برای برنامه‌ریزی ماشین‌های تورینگ است. در برنامه‌نویسی کامپیوتری، شبه‌کد روشی برای شرح کلی یک مساسبه با استفاده از عبارات توصیفی معنادار است. هرچند از این توصیف‌ها نمی‌توان در کامپیوتر استفاده کرد، ولی بصورت فرضی می‌توان در صورت لزوم، آنرا به یک زبان مناسب ترجمه کرد. کلان‌دستور نمونه ساده‌ای از شبه‌کد است که امکان ارائه سریعتر و تک‌عبارتی برای دنباله‌ای از جملات سطح پایین‌تر را فراهم می‌کند. ابتدا کلان‌دستور را براساس زبان سطح پایین‌تر تعریف می‌کنیم. سپس با این فرض که به جای هر یک از کلان‌دستورها از کد سطح پایین مربوطه استفاده کرده‌ایم، کلان‌برنامه را در قالب یک برنامه مطرح می‌کنیم. از این روش به طور گسترده در برنامه‌نویسی ماشین‌های تورینگ استفاده شده است.

مثال ۹-۱۳

کلان‌دستور

$$\textit{if a then } q_j \text{ else } q_k,$$

با تفسیر زیر را در نظر بگیرید. اگر ماشین تورینگ یک سمبل a را بخواند، آنگاه باید بدون تغییر محتوای نوار یا جابجایی هد خواندن- نوشتن و بدون توجه به حالت جاری خود، به حالت q_j برود. اما اگر سمبل خوانده شده a نباشد، ماشین بدون کوچکترین تغییری به حالت q_k خواهد رفت.

برای پیاده‌سازی این کلان‌دستور، ماشین تورینگ باید چند مرحله را پشت‌سر بگذارد.

$$\delta(q_i,a)=(q_{j0},a,R) \text{ ،} q_i \in Q \text{ برای همه}$$

$$\delta(q_i,b)=(q_{k0},b,R) \text{ ،} b \in \Gamma-\{a\} \text{ و همهٔ } q_i \in Q \text{ برای همه}$$

$$\delta(q_{j0},c)=(q_j,c,L) \text{ ،} c \in \Gamma \text{ برای همه}$$

$$\delta(q_{k0},c)=(q_k,c,L) \text{ ،} c \in \Gamma \text{ برای همه}$$

اما توجه داشته باشید که در ماشین‌های تورینگ استاندارد، محل هد خواندن- نوشتن در هر

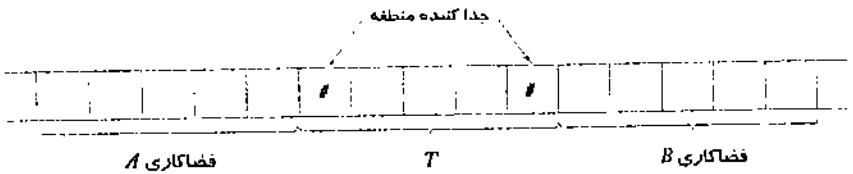
حرکت تغییر می‌کند. در بالا، با معرفی حالت‌های جدید q_{10} و q_{20} از مشکلات ناشی از این امر جلوگیری کردیم. در کلان‌دستورها خواهان تغییر حالت هستیم، اما محل قرارگیری هد خواندن-نوشتن را به هیچ وجه تغییر نمی‌دهیم. هرچند اجازه می‌دهیم که هد به سمت راست حرکت کند، ماشین را در حالت q_{10} یا q_{20} قرار می‌دهیم. بنابراین، باید پیش از ورود به حالت مطلوب q_i یا q_i، یک حرکت به چپ انجام دهیم.

■

حال یک قدم به جلو گذاشته و به جای کلان‌دستور از زیربرنامه استفاده می‌کنیم. معمولاً، کلان‌دستورها در محل وقوع خود با یک کد واقعی جایگزین می‌شوند؛ در حالیکه زیربرنامه قطعه کدی است که مکرراً در صورت نیاز فراخوانی می‌شود. هرچند زیربرنامه‌ها اساس و شالوده زبان‌های برنامه‌سازی سطح بالا را تشکیل می‌دهند، اما می‌توان از آنها در ماشین‌های تورینگ هم استفاده کرد. در نمونه زیر، نشان می‌دهیم که چگونه می‌توان ضمن استفاده از یک ماشین تورینگ بعنوان زیربرنامه، به دفعات آنرا توسط ماشین تورینگ دیگر فراخوانی کرد. اما موفقیت در این‌کار مستلزم ایجاد یک ویژگی جدید است: باید امکان ذخیره‌سازی اطلاعات بر روی پیکربندی ماشین فراخوان فراهم شود تا بتوان در بازگشت از زیربرنامه، مجدداً پیکربندی را بدست آورد. بعنوان مثال، ماشین A در حالت q_i ماشین B را فراخوانی می‌کند. می‌خواهیم پس از پایان کار بر روی B، در حالیکه هد خواندن- نوشتن (که ممکن است در حین عملیات B جابجا شده باشد) در محل اصلی خود قرار دارد، برنامه A را از حالت q_i از سر بگیریم. در دیگر موارد، A ممکن است B را از حالت q_i فراخوانی کند که در این صورت، کنترل باید به حالت مذکور بازگردد. برای رفع مشکل انتقال کنترل، باید امکان انتقال اطلاعات از A به B و بالعکس وجود داشته باشد، باید بتوانیم پس از بازپس‌گیری کنترل از B توسط A، A را مجدداً پیکربندی کنیم و در ضمن، مطمئن شویم که محاسبات موقتاً به تعویق افتاده A، تحت تأثیر اجرای B قرار نمی‌گیرند. به این منظور، مطابق شکل ۹-۹، نوار را به چندین منطقه تقسیم می‌کنیم.

پیش از فراخوانی B توسط A، آن اطلاعات مورد نیاز B (از قبیل حالت جاری A و شناسه‌های مورد نیاز B) را در منطقه مفروض T نوار می‌نویسد. سپس A با یک انتقال به حالت شروع B، کنترل را به B پاس می‌دهد. پس از انتقال، B با استفاده از T ورودی خود را پیدا می‌کند. فضای کار B جدا از T و آن نیز متقابلاً جدا از A بوده و به این دلیل هیچ‌گاه با هم تداخل نمی‌کنند. پس از پایان کار، B نتایج مربوطه را به منطقه T بازمی‌گرداند و A در آنجا به دنبال B خواهد گشت. به این ترتیب، هر دو برنامه می‌توانند به روش لازم با هم تقابل کنند. توجه داشته باشید که این تحولات بسیار شبیه به آن-چیزی است که در کامپیوترهای واقعی در هنگام فراخوانی یک زیربرنامه انجام می‌شود.

حال با روش برنامه‌ریزی ماشین‌های تورینگ بوسیله شبه‌کدها آشنا شدید. البته پیش از آن باید (حداقل به صورت نظری) از نحوه ترجمه شبه‌کد به برنامه واقعی ماشین تورینگ اطلاع داشته باشیم.



شکل ۹-۹

مثال ۹-۱۴

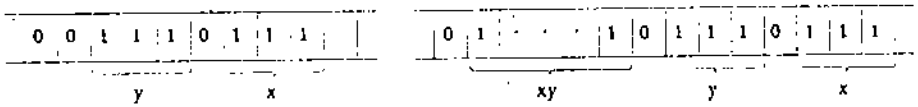
ماشین تورینگی طراحی کنید که دو عدد صحیح مثبت تعریف شده توسط دنباله سمبل‌های ۱ها، در یکدیگر ضرب کند.

برای ساخت ماشین ضرب کنندهٔ مورد نظر، نکاتی را که در مسائل قبل در مورد جمع‌کننده و کپی‌کردن گفتیم، با هم ترکیب می‌کنیم. فرض می‌کنیم که محتویات شروع و پایانی نوار مطابق شکل ۹-۱۰ باشند. بنابراین، فرآیند ضرب‌کردن را می‌توان بعنوان کپی‌کردن مکرر y مضروب، به ازای هر ۱، در مضروب‌فیه x، در نظرگرفت. به این ترتیب، رشته y به هر تعداد لازم به حاصل ضرب جزئی محاسبه شده، اضافه می‌شود. شبه‌کد زیر مراحل اصلی این فرآیند را نشان می‌دهد:

۱. مراحل زیر را آنقدر تکرار کنید که هیچ 1 دیگری در x به چشم نخورد.
یک 1 در x پیدا کرده و آنرا با سمبل a جایگزین کنید.
چپ‌ترین 0 را با $0y$ جایگزین کنید.

۲. تمام a ها را با 1 جایگزین کنید.

هرچند در بالا کلیتی از نحوه ایجاد شبه‌کد ارائه شده است، مراحل کار به قدری ساده است که بدون شک هر کسی قادر به انجام آن است.



شکل ۹-۱۰

■

به لطف ماهیت توصیفی این مثال‌ها، مشاهده می‌کنید که می‌توان همین ماشین‌های تورینگ ساده را، به روش‌های مختلف با هم ترکیب کرده و تورینگ‌های بسیار قدرتمندی ایجاد نمود. مثال‌های ما آنقدر کلی و تفصیلی نبودند که بتوان از آنها برای اثبات چیزی استفاده کرد، اما اکنون دیگر می‌توان با اطمینان کامل اظهار کرد که ماشین‌های تورینگ قادر به انجام کارهای بسیار پیچیده هستند.

تمرین‌ها

۱. جواب کامل مثال ۹-۱۴ را بنویسید.

۲. یک قرارداد برای نمایش اعداد صحیح مثبت و منفی توسط دنباله سمبل‌های 1، ارائه کنید. به کمک این قرارداد، طرح کلی برای ساخت یک کاهنده جهت محاسبه $x-y$ ارائه دهید.

۳. با استفاده از جمع‌کننده‌ها، تفریق‌کننده‌ها، مقایسه‌گرها، کپی‌کننده‌ها یا ضرب‌کننده‌ها، نمودارهای بلوکی برای ماشین‌های تورینگ جهت محاسبه توابع زیر به ازای تمام اعداد صحیح مثبت n رسم کنید.

الف) $f(n)=n(n+1)$. ⊖

ب) $f(n)=n^3$.

ج) $f(n)=2^n$.

د) $f(n)=n!$.

ه) $f(n)=n^{n!}$.

۴. با استفاده از یک نمودار بلوکی، پیاده‌سازی را برای تابع مفروض f طراحی کنید که برای تمام $w_1,w_2,w_3\in\{1\}^*$ به صورت

$$f(w_1,w_2,w_3)=i,$$

تعریف می‌شود. در این رابطه i چنان است که اگر هیچ دو w طول یکسان نداشته باشند $i=\max(|w_1|,|w_2|,|w_3|)$، در غیر اینصورت $i=0$.

۵. توصیف "سطح بالایی" برای ماشین‌های تورینگی پیدا کنید که زبان‌های زیر را روی $\{a,b\}$ پذیرش کند. در هر مسأله، مجموعه‌ای از کلان‌دستورهای مناسب را تعریف کنید که به راحتی قابل پیاده‌سازی باشند. سپس از آنها برای حل موارد زیر استفاده کنید:

الف) $L=\{ww^R\}$.

ب) $L=\{w_1w_2: w_1\neq w_2,\ |w_1|=|w_2|\}$.

ج) مکمل زبان قسمت (الف). ⊖

د) $L=\{a^nb^m: m=n^2, n\geq 1\}$.

ه) $L=\{a^n:$ n یک عدد اول است $\}$.

۶. روشی برای نمایش اعداد گویا روی ماشین‌های تورینگ پیشنهاد کنید. سپس روشی برای اضافه و کم‌کردن این اعداد طراحی کنید.

۷. روش کلی ساخت ماشین تورینگی را ارائه دهید که بتواند عملیات جمع و ضرب کردن اعداد صحیح مثبت x و y را در که با نشانه‌گذاری معمول دهدهی تعریف شده‌اند، انجام دهد.

۸. پیاده‌سازی برای کلان‌دستور

$$searchright(a,q_i,q_j),$$

به این معنی پیدا کنید که، ماشین برای پیدا کردن اولین وقوع a، سمت راست موقعیت فعلی خود را روی نوار جستجو می‌کند. اگر پیش از یک □ به یک a برخورد کنیم، ماشین باید به حالت q_i و در غیر اینصورت به حالت q_j برود. ⊖

۹. با استفاده از کلان‌دستور تمرین قبل، ماشین تورینگی روی $\Sigma=\{a,b\}$ طراحی کنید که زبان $L(ab^*ab^*a)$ را پذیرش می‌کند.

۱۰. با استفاده از کلان‌دستور *searchright* در تمرین ۸، برنامه‌ای برای یک ماشین تورینگ بنویسید که سمبل بلافاصله واقع در سمت چپ چپ‌ترین علامت a را با یک □ جایگزین کند. چنانچه در ورودی هیچ aای وجود نداشته باشد، سمت راست‌ترین سمبل غیر □ را با یک b جایگزین کند.

۹-۳ فرضیه تورینگ

در مبحث قبل دیدیم که ماشین‌های تورینگ از اجزاء ساده‌ای تشکیل شده‌اند. بعلاوه، با یکی از نقاط ضعف عملکرد با این اتوماتای سطح پایین آشنا شدیم: هرچند ترجمه یک نمودار بلوکی یا شبه‌کد به برنامه ماشین تورینگ نظیر آن، به هوش یا قدرت تخیل چندان زیادی نیاز ندارد، انجام عملی این کار زمانبر، خطاپذیر و نه چندان آموزنده است. مجموعه دستورالعمل‌های یک ماشین تورینگ آنقدر محدود است که ارائه هرگونه استدلال، راه‌حل یا اثباتی برای یک مسأله نه چندان ساده هم بسیار خسته‌کننده و ملال‌آور به نظر می‌رسد.

این در حالی است که می‌خواهیم ادعا کنیم ماشین‌های تورینگ علاوه بر انجام عملیات ساده‌ای که برنامه‌های مفصلی برای آنها ایجاد کرده‌ایم، قادر به انجام فرآیندهای پیچیده‌تری نیز هستند که بوسیله نمودارهای بلوکی یا شبه‌کدها توصیف می‌شوند. در اثبات این ادعاها، باید برنامه‌های مربوط به آنها را بطور کامل ارائه کنیم. اما چون انجام این کار ناخوشایند و حتی گمراه‌کننده است، باید حتی‌الامکان از آن اجتناب شود. پس بهتر است بجای صرف وقت، برای نوشتن کدهای طولانی و سطح پایین، روشی برای بحث دقیق و روشن در مورد ماشین‌های تورینگ ارائه کنیم. اما چون تاکنون هیچ اجماع نظری در مورد تعریف این ماشین‌ها وجود نداشته، بهتر است حداقل در مورد یک تعریف به توافق قابل قبولی برسیم. برای بررسی طریقه دستیابی به این توافق، یک موضوع تقریباً فلسفی را مطرح می‌کنیم.

از برخی مثال‌های بخش قبل می‌توان به نتایج تقریباً ساده‌ای دست یافت. اولین نتیجه، این که ماشین‌های تورینگ قدرتمندتر از اتوماتای پشته‌ای به نظر می‌رسند(برای اثبات به تمرین ۲ انتهای همین بخش مراجعه کنید). در مثال ۹-۸ طرح کلی ساخت ماشین تورینگ را برای زبانی ارائه کردیم که مستقل‌ازمتن نبود و درنتیجه، هیچ اتومات پشته‌ای هم به ازای آن وجود نداشت. در مثال‌های ۹-۹، ۹-۱۰ و ۹-۱۱ مشاهده کردیم که ماشین‌های تورینگ قادر به انجام برخی عملیات ساده ریاضی، دستکاری در رشته و انجام برخی مقایسه‌های ساده هستند. همچنین در مباحث گذشته دیدیم که می‌توان عملیات‌های ساده را با هم ترکیب و از آنها برای حل مسائل پیچیده‌تر کمک گرفت، چگونه چندین ماشین تورینگ می‌توانند با هم ترکیب شوند و چگونه گاهی اوقات از یک برنامه بعنوان

زیربرنامهٔ برنامه دیگر می‌توان استفاده کرد. بدلیل آنکه می‌توان از این روش‌ها برای ایجاد عملیات بسیار پیچیده استفاده کرد، شاید بتوان انتظار داشت که یک ماشین تورینگ را هم از نظر قدرت تا حدودی به یک کامپیوتر نزدیک کرد.

به فرض، این ادعا را مطرح کنیم که ماشین‌های تورینگ، از بعضی جهات، به اندازه یک کامپیوتر رقمی امروزی قدرتمند هستند. چگونه می‌توان این فرضیه را تأیید یا رد کرد؟ یک روش برای تأیید این است که دنباله‌ای از مسائل تدریجاً مشکل را در نظر گرفته و آنها را با یک ماشین تورینگ حل می‌کنیم. همچنین می‌توان مجموعه دستورالعمل‌های زبان ماشین یک کامپیوتر خاص را در نظر گرفته و ماشین تورینگی را طراحی کرد که قادر به انجام تمامی دستورالعمل‌های داخل این مجموعه باشد. هرچند انجام این کار خارج از حوصله ما و کتاب ما می‌باشد، اما اگر فرضیه بالا درست باشد، باید در نهایت جواب درستی برای آن وجود داشته باشد. هرچند هر موفقیتی در این زمینه می‌تواند عقیده ما را در مورد درستی فرضیه محکم‌تر کند، باز هم هیچ چیزی را اثبات نمی‌کند. مشکل واقعی ناشی از آن است که نه دقیقاً معنای عبارت یک "کامپیوتر رقمی امروزی" را می‌دانیم و نه روشی برای ارائه یک تعریف دقیق داریم.

روش دیگر برای حل مسأله بالا این است که ابتدا روالی را مطرح و یک برنامه کامپیوتری برای آن نوشت. سپس نشان داد که هیچ ماشین تورینگی برای آن وجود ندارد. در صورت اثبات، می‌توان آنرا بعنوان پایه و دلیلی برای رد فرضیه بالا ارائه کرد. اما واقعیت این است که تا به حال هیچ کس قادر به ارائه چنین مثال نقضی نبوده است. بنابراین، این شکست‌های متوالی را باید دلیل ضمنی برای غیرقابل نقض‌کردن آن فرضیه تلقی کرد. تمامی شواهد تا به اینجا حاکی از آن بوده که ماشین‌های تورینگ در کل به اندازه کامپیوترها قدرتمند هستند.

استدلال‌هایی از این دست باعث شد تا A.M.Turing و برخی افراد دیگر، در اوایل دهه ۳۰، ایده معروفی به نام **نظریه تورینگ** را مطرح کنند. براساس این فرضیه، هر محاسبه‌ای که بوسیله ابزار مکانیکی قابل انجام باشد، بوسیله یک ماشین تورینگ هم انجام‌پذیر خواهد بود.

این گفته یک اظهار نظر کلی بوده و بنابراین باید معنای نظریه تورینگ را به خاطر داشته باشیم. این‌کار با طرح ادعا و اثبات آن امکان‌پذیر نیست، بلکه باید عبارت "بوسیله ابزار مکانیکی" را دقیقاً تعریف کنیم. به این منظور، باید یک مدل انتزاعی دیگر و تقریباً مشابه مدل‌های قبلی در اختیار داشته باشیم. بهتر است نظریه تورینگ را، بیشتر تعریفی از مؤلفه‌های یک محاسبه مکانیکی تلقی کرد: یک محاسبه مکانیکی خواهد بود اگر و تنها اگر بوسیله یک ماشین تورینگ مفروض قابل انجام باشد.

اگر این ایده را پذیرفته و ماشین‌های تورینگ را صرفاً نوعی تعریف در نظر بگیریم، جامع بودن یا نبودن تعریف بالا مورد تردید قرار می‌گیرد. آیا این تعریف آنقدر گسترده است که بتواند هر آنچه را که اکنون بوسیله کامپیوترها انجام می‌دهیم (احتمالاً در آینده انجام خواهیم داد) در خود جای دهد؟ گرچه نمی‌توان بطورصریح به این سؤال جواب "مثبت" داد، اما شواهد زیادی در تأیید آن وجود دارد. برخی استدلال‌ها در تأیید تز تورینگ بعنوان تعریف یک محاسبه مکانیکی عبارتند از:

۱. هر عملی را که بتوان توسط هر نوع کامپیوترهای رقمی امروزی انجام داد، بوسیله یک ماشین تورینگ نیز، قابل انجام است.
۲. تا به حال هیچ‌کس قادر نبوده مسأله‌ای را طرح و آنرا بوسیله الگوریتم‌های شهودی حل کند، اما یک برنامه ماشین تورینگ به ازای آن نوشته نشده باشد.
۳. هرچند مدل‌های جایگزینی برای محاسبه مکانیکی پیشنهاد شده، هیچ‌کدام از آنها قدرتمندتر از مدل ماشین تورینگ نبوده و نیستند.

این استدلال‌ها ضمنی بوده و نمی‌توان تز تورینگ را براساس آنها اثبات کرد. تز تورینگ علیرغم توجیه‌پذیر بودن، هنوز در حد یک فرضیه باقی مانده است. اما اگر تز تورینگ را صرفاً یک تعریف بدانیم، از یک نکته غفلت کرده‌ایم. از یک نظر، تز تورینگ درست همان نقشی را در علوم کامپیوتر بازی می‌کند که قوانین اساسی فیزیک و شیمی بر عهده دارند. بعنوان مثال، فیزیک کلاسیک تا حد زیادی براساس قوانین حرکت نیوتن استوار است. با این وجود، این به اصطلاح قوانین نیوتن هیچ الزام منطقی بوجود نمی‌آورند؛ بلکه بیشتر، مدل‌های موجهی برای تشریح دنیای فیزیک محسوب می‌شوند. اما چون نتایج حاصل از آنها با تجربه و مشاهدات ما همخوانی دارد، آنها را تا به امروز پذیرفته‌ایم. هرچند نمی‌توان درستی این قوانین را اثبات کرد، با این وجود امکان نامعتبر بودن آنها را هم نباید نادیده گرفت. اگر یکی از نتایج تجربی ما با نتیجه‌ای که براساس این قوانین می‌گیریم در تضاد باشد، می‌توان اعتبار آنها را مورد تردید قرار داد. از طرف دیگر، ناکامی‌های مکرر در نامعتبر کردن یک قانون، اطمینان ما از صحت آنها را هم افزایش می‌دهد. همین امر در مورد تز تورینگ صدق می‌کند و بنابراین می‌توان آنها را یکی از قوانین اساسی علوم کامپیوتر دانست. نتایجی که از این تز می‌گیریم، با دانسته‌های ما در مورد کامپیوترهای واقعی همخوانی دارد و تا به حال، تمام تلاش‌ها برای نامعتبر جلوه دادن آن با شکست مواجه شده‌اند. همواره این امکان وجود دارد که فرد دیگری در گوشه‌ای از دنیا تعریف دیگری را مطرح کند که هرچند در نظریه ماشین‌های تورینگ مدنظر قرار نگرفته، اما در محدوده عقاید شهودی محاسبه مکانیکی قرار دارد. در این صورت، برخی از مباحث بعدی ما به طورکلی تغییر و اصلاح خواهد شد. اما چون تا به حال چنین اتفاقی رخ نداده و احتمال وقوع آن هم در آینده بسیار ناچیز است، تا به امروز مجبوریم تعریف بالا را بپذیریم.

حال که تز تورینگ را پذیرفتیم، می‌توان تعریف دقیقی از یک الگوریتم ارائه کرد.

تعریف ۹-۵

یک **الگوریتم** برای تابع $f : D \to R$، ماشین تورینگ M است که با هر ورودی داده شده $d \in D$ روی نوار خود، و در نهایت با پاسخ درست $f(d) \in R$ بر روی نوار خود، متوقف می‌شود.

به‌خصوص، می‌توان الزام کرد که به ازای هر $d \in D$،

$$q_0 d \overset{*}{\mapsto}_M q_f f(d), \quad q_f \in F.$$

به رسمیت شناختن وجود الگوریتم در برنامه ماشین‌های تورینگ به ما امکان می‌دهد که ادعاهایی از قبیل "یک الگوریتم وجود دارد که ..." یا " هیچ الگوریتمی وجود ندارد که ..."را به دقت اثبات و با اطمینان بپذیریم. با این وجود، ساخت صریح یک الگوریتم برای مسائل حتی نه چندان مشکل هم کار بسیار سخت و طاقت‌فرسایی است. برای رفع این نقص، می‌توان دست به دامان تز تورینگ شد و ادعا کرد که هر آنچه را که بوسیله یک کامپیوتر قابل انجام است، بوسیله یک ماشین تورینگ هم انجام‌پذیر می‌باشد. در نتیجه، می‌توان در تعریف ۵-۹، به جای عبارت "ماشین تورینگ" از "برنامه C" استفاده کرد. این‌کار تا حد زیادی باعث تسهیل ارائه الگوریتم می‌شود. در واقع، همانطور که قبلاً انجام دادیم، یک قدم به جلو برداشته و توصیف‌های شفاهی یا نمودارهای بلوکی را بعنوان الگوریتم می‌پذیریم این‌کار به این دلیل قابل انجام است که همواره می‌توان از این توصیف‌ها و نمودارها برای نوشتن برنامه ماشین تورینگ استفاده کرد. هرچند این کار تا حد زیادی باعث تسهیل بحث می‌شود، ولی ما را به شدت در معرض انتقادات قرار می‌دهد. چون عبارت "برنامه C " کاملاً مشخص و مفهوم است، اما عبارت نامشخص "توصیف شفاهی دقیق" ممکن است ما را به سمت پرتگاه ادعای وجود الگوریتم‌های غیرممکن سوق دهد. با این وجود، این خطر در مقایسه با مزیت تسهیل بحث و روشن کردن معنای شهودی آن ناچیز بوده و با پذیرش آن می‌توان توصیف‌های دقیقی را برای برخی فرآیندهای پیچیده‌تر ارائه کرد. اگر شما خواننده عزیز در مورد اعتبار و صحت هریک از این ادعاها شک دارید، می‌توانید با نوشتن یک برنامه مناسب به یکی از زبانهای برنامه‌سازی به چشم خود صحت آنرا مشاهده و بطور منطقی آنرا تأیید کنید.

تمرین‌ها

۱. ** – مجموعه دستورالعمل‌های زبان ماشین یک کامپیوتر دلخواه را در نظر بگیرید. چگونه می‌توان مجموعه دستورالعمل‌های مختلف این کامپیوتر را بوسیله یک ماشین تورینگ اجرا کرد.

۲. در بحث بالا، در جایی اشاره کردیم که ماشین‌های تورینگ قدرتمندتر از اتوماتای پشته‌ای به نظر می‌رسند. بدلیل آنکه همواره می‌توان نوار یک ماشین تورینگ را به صورتی تغییر داد که مانند یک پشته عمل کند، عملاً هم می‌توان ادعا کرد که ماشین تورینگ قدرتمندتر از اتومات پشته‌ای است. در این استدلال از چه نکته مهمی غفلت کرده‌ایم؟

۳. ** – در منابع مختلف، مقالات جالبی در مورد ماشین‌های تورینگ وجود دارد. بعنوان یکی از بهترین نمونه‌ها می‌توان به مقاله‌ای تحت عنوان "ماشین‌های تورینگ" در *Scientific American* چاپ May سال۱۹۸۴ به قلم J.E.Hopcroft اشاره کرد. در این مقاله، ایده‌هایی که در اینجا مطرح کردیم به همراه برخی شواهد تاریخی در مورد فعالیت‌های تورینگ و دیگر فعالان این حرفه به چشم می‌خورد. یک نسخه از این مقاله تهیه کنید و پس از خواندن، آنرا به اختصار شرح دهید.

فصل ۱۰

مدل‌های دیگر ماشین‌های تورینگ

تعریفی که در فصل قبل از ماشین‌های تورینگ استاندارد ارائه کردیم تنها تعریف ممکن نبود و تعریف‌های دیگری هم وجود دارند که می‌توانند به همان اندازه گویا و مفید باشند. بعلاوه، قدرت ماشین‌های تورینگ ارتباط چندانی با ساختار مورد استفاده در آنها ندارد. در این فصل، با نگاهی به گونه‌های مختلف ماشین‌های توزینگ نشان می‌دهیم که ماشین‌های تورینگ استاندارد، از نقطه نظری که تعریف می‌کنیم، معادل با مدل‌های پیچیده‌تر هستند.

در صورت پذیرش تز تورینگ، انتظار می‌رود که پیچیده کردن ماشین‌های تورینگ استاندارد از طریق تجهیز آنها به ابزار ذخیره‌سازی پیچیده‌تر، تأثیری بر قدرت اتومات نداشته باشد. هر نوع محاسبه‌ای که با این تنظیمات جدید قابل انجام باشد، مدلی از محاسبه مکانیکی محسوب شده و از اینرو، بوسیله یک مدل استاندارد هم قابل انجام است. با این وجود، بهتر است مدل‌های پیچیده‌تر را هم مطالعه کنیم. حداقل مزیت این کار آن است که به عینه شاهد اثبات یکی از نتایج قدرت ماشین تورینگ خواهیم بود و بنابراین، اعتماد و اطمینان ما نسبت به تز تورینگ افزایش می‌یابد. می‌توان با ایجاد تغییراتی در تعریف ۹-۱، مدل‌های متنوعی از ماشین‌های تورینگ را ایجاد و مورد مطالعه قرار داد. بعنوان مثال، ماشین‌های تورینگ با بیش از یک نوار یا چند نواری را در نظر می‌گیریم که در ابعاد مختلف، گسترش و توسعه پیدا می‌کنند. البته انواع ماشین‌های تورینگ مورد بحث در اینجا از بین پرکاربردترین و معروف‌ترین آنها انتخاب شده است.

سپس نگاهی به ماشین‌های تورینگ نامعین انداخته و نشان می‌دهیم که این ماشین‌ها قدرتمندتر از انواع معین خود نیستند. البته این امر ممکن است عجیب و غیرمنتظره به نظر برسد، چون تز تورینگ فقط در مورد محاسبات مکانیکی صدق کرده و نکات ضمنی ناشی از نامعین بودن را شامل نمی‌شود. مشکل دیگری که بوسیله تز تورینگ به راحتی قابل حل نیست، هنگامی بوجود می‌آید که یک ماشین

برنامه‌های مختلفی را در زمان‌های مختلف اجرا می‌کند. این امر منجر به ارائه تز ماشین‌های تورینگ با "قابلیت برنامه‌ریزی مجدد" یا "عمومی" می‌شود.

در پایان، نگاهی به اتوماتای کراندار خطی خواهیم داشت. این ماشین‌های تورینگ علیرغم برخورداری از یک نوار نامتناهی، فقط به صورت محدود قادر به استفاده از نوار می‌باشند.

۱۰-۱ اعمال تغییرات جزئی در تعریف ماشین تورینگ

ابتدا برخی تغییرات نسبتاً جزئی را در تعریف ۹-۱ اعمال کرده و تأثیر یا عدم تأثیر آنها را در مفهوم کلی بررسی می‌کنیم. با هر تغییر تعریف، نوع جدیدی از اتوماتا را معرفی کرده و این سؤال را مطرح می‌کنیم که آیا اتوماتای جدید واقعاً با اتوماتای قبلی تفاوت دارد یا خیر. اما پیش از آن باید منظور دقیق خود را از تفاوت اساسی بین دسته‌های مختلف اتوماتا بیان کنیم. هرچند ممکن است تفاوت‌های مشخصی بین تعریف دسته‌های اتوماتا وجود داشته باشد، این تفاوت‌ها بطور معمول هیچ پیامد قابل ملاحظه‌ای به همراه ندارند. قبلاً نمونه‌ای از این مورد را در اتوماتای متناهی معین و نامعین مشاهده کردیم. هرچند این دو اتومات تعاریف کاملاً متفاوتی دارند، اما به دلیل اینکه هر دو متعلق به خانواده زبان‌های منظم هستند، هم‌ارز محسوب می‌شوند. بر این اساس، می‌توان هم‌ارزی یا غیر هم‌ارزی کلی بین دسته‌های مختلف اتوماتا را به صورت زیر تعریف کرد.

هم‌ارزی بین دسته‌های مختلف اتوماتا

هرگاه هم‌ارزی بین دو اتومات یا دو دسته اتومات را تعریف می‌کنیم، باید دقیقاً منظور خود را از این هم‌ارزی بیان کنیم. در ادامه فصل، با اقتباس از روال تعریف هم‌ارزی در nfa ها و dfa ها، خاصیت را براساس امکان یا عدم امکان پذیرش زبان‌ها مطرح می‌کنیم.

تعریف ۱۰-۱

دو اتومات را در صورتی هم‌ارز می‌خوانیم که هر دو یک زبان یکسان را پذیرش کنند. دو دسته اتوماتای C_1 و C_2 را در نظر بگیرید. اگر برای هر اتومات M_1 در C_1، اتومات M_2 در C_2 وجود داشته باشد بطوریکه

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

می‌گوییم که قدرت C_2 حداقل به اندازه C_1 است. اگر عکس این رابطه هم درست باشد و به ازای هر M_2 موجود در C_2، یک M_1 در C_1 وجود داشته باشد بطوریکه $L(M_1) = L(M_2)$ باشد، آنگاه C_1 و C_2 را هم ارز می‌خوانیم.

روش‌های مختلفی برای تعریف هم‌ارزی بین اتوماتا وجود دارد. با استفاده از قضیه ۲-۲ هم‌ارزی

را در dfa ها و nfa ها تعریف کردیم. اما برای اثبات هم‌ارزی در ماشین‌های تورینگ غالباً از روش مهمی به نام شبیه‌سازی استفاده می‌شود.

فرض کنید که M یک اتومات باشد. می‌گوییم که اتومات دیگر $\hat{M}$ قادر به شبیه‌سازی یکی از محاسبات M است اگر $\hat{M}$ بتواند به ترتیب زیر یکی از محاسبات M را تقلید کند. فرض کنید $d_0, d_1, \ldots$ دنباله‌ای از پیکربندی‌های محاسبه M باشد، یعنی،

$$d_0 \vdash_M d_1 \vdash_M \cdots \vdash_M d_n \cdots.$$

بنابراین، $\hat{M}$ با انجام

$$\hat{d}_0 \vdash_{\hat{M}} \hat{d}_1 \vdash_{\hat{M}} \cdots \vdash_{\hat{M}} \hat{d}_n \cdots,$$

این محاسبه را شبیه‌سازی می‌کند. لازم به ذکر است که $\hat{d}_0, \hat{d}_1, \ldots$ پیکربندی‌هایی هستند که هر یک از آن‌ها یک پیکربندی منحصربفردی از M را معرفی می‌کند. به بیان دیگر، با در اختیار داشتن پیکربندی شروع نظیر M و به شرط اطلاع از محاسبه انجام شده بوسیله آن، می‌توان محاسبات انجام شده بوسیله $\hat{M}$ را دقیقاً تعیین کرد.

توجه داشته باشید که در شبیه‌سازی، فقط یکی از حرکات $d_i \vdash_M d_{i+1}$ از M، ممکن است در چندین حرکت $\hat{M}$ دخالت داشته باشد. گاهی اوقات، پیکربندی‌های میانی در $\hat{d}_i \vdash_{\hat{M}} \hat{d}_{i+1}$ مشابه هیچ‌کدام از پیکربندی‌های M نیستند؛ اما اگر بتوان پیکربندی‌های مهم $\hat{M}$ را تعیین کرد، این امر کوچکترین تأثیر بدی بر کار ما نخواهد داشت. تا زمانی که بتوان با بررسی محاسبه $\hat{M}$ اقدام به شناسایی محاسبات M کرد، می‌توان از شبیه‌سازی استفاده کرد. اگر $\hat{M}$ تمامی محاسبات M را شبیه‌سازی کند، می‌گوییم که $\hat{M}$ قادر به شبیه‌سازی M می‌باشد. به روشنی می‌توان دریافت که اگر $\hat{M}$ قادر به شبیه‌سازی M باشد، آنگاه می‌توان وقایع را به گونه‌ای تنظیم کرد که اتوماتای M و $\hat{M}$، یک زبان یکسان را پذیرفته و در نتیجه هر دو هم‌ارز می‌شوند. برای اثبات هم‌ارزی بین دو دسته اتوماتا، نشان می‌دهیم که به ازای هر ماشین از دسته اول، یک ماشین در دسته دوم وجود دارد که آنرا شبیه‌سازی می‌کند.

ماشین‌های تورینگ سکون‌دار

در تعریفی که در فصل قبل از ماشین‌های تورینگ استاندارد ارائه کردیم، هد خواندن-نوشتن به یکی از طرفین راست یا چپ حرکت می‌کرد. گاهی اوقات بهتر است که یک انتخاب سوم هم در نظر گرفته شود، یعنی هد پس از بازنویسی محتوای سلول، در جای خود باقی بماند. این ماشین‌های تورینگ اصطلاحاً سکون‌دار را می‌توان با جایگذاری δ به صورت

$$\delta : Q \times \Gamma \to Q \times \Gamma \times \{L, R, S\}$$

در تعریف ۹-۱ ارائه کرد. در تفسیر تابع انتقال فوق می‌گوییم که S هد خواندن-نوشتن را هیچ حرکتی نمی‌دهد. گنجاندن این انتخاب جدید برای حرکت هد، قدرت اتومات را افزایش نخواهد داد.

قضیه ۱۰-۱

دسته ماشین‌های تورینگ سکون‌دار هم‌ارز با دسته ماشین‌های تورینگ استاندارد می‌باشند.

اثبات: بدلیل آنکه ماشین‌های تورینگ سکون‌دار یکی از مدل‌های بسط‌یافته مدل استاندارد محسوب می‌شوند، ناگفته پیداست که تمامی ماشین‌های تورینگ استاندارد را می‌توان بوسیله ماشین‌های تورینگ سکون‌دار شبیه‌سازی کرد.

برای اثبات حالت عکس، $M = (Q, \Sigma, \Gamma, \delta, q_0, \square, F)$ را یک ماشین تورینگ سکون‌دار در نظر بگیرید که باید بوسیله ماشین تورینگ استاندارد $\hat{M} = (\hat{Q}, \Sigma, \Gamma, \hat{\delta}, \hat{q}_0, \square, \hat{F})$ شبیه‌سازی شود. به ازای هر حرکت M ماشین شبیه‌ساز $\hat{M}$ مراحل زیر را انجام می‌دهد. اگر در حرکت M انتخاب سکون دخالت نداشته باشد، ماشین شبیه‌ساز حرکتی کاملاً مشابه حرکتی را انجام می‌دهد که باید شبیه‌سازی شود. اما اگر در حرکت دادن هر خواندن- نوشتن، S وجود داشته باشد، آنگاه $\hat{M}$ دو حرکت انجام می‌دهد: با اولین حرکت علامت را بازنویسی کرده و هد خواندن- نوشتن را به سمت راست حرکت می‌دهد. در حرکت دوم، هد خواندن- نوشتن را به سمت چپ حرکت داده، ولی تغییری در محتویات نوار انجام نمی‌دهد. ماشین شبیه‌ساز را می‌توان با تعریف $\hat{\delta}$ به صورت زیر با استفاده از M ایجاد نمود:

برای هریک از انتقالات

$$\delta(q_i, a) = (q_j, b, L \text{ or } R),$$

رابطه

$$\hat{\delta}(\hat{q}_i, a) = (\hat{q}_j, b, L \text{ or } R)$$

را در $\hat{\delta}$ قرار می‌دهیم.

به ازای هر انتقال از نوع S به صورت

$$\delta(q_i, a) = (q_j, b, S),$$

انتقال‌های متناظر

$$\hat{\delta}(\hat{q}_i, a) = (\hat{q}_{j_1}, b, R),$$

و

$$\hat{\delta}(\hat{q}_{j_1}, c) = (\hat{q}_j, c, L),$$

را به ازای تمام $c \in \Gamma$، در $\hat{\delta}$ قرار می‌دهیم.

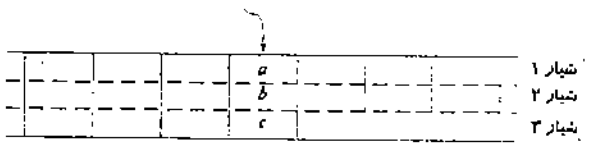
ناگفته پیداست که هر یک از محاسبات M دارای یک محاسبه متناظر در $\hat{M}$ بوده و بنابراین، $\hat{M}$ قادر به شبیه‌سازی M می‌باشد. ■

شبیه‌سازی یکی از روش‌های استاندارد برای اثبات معادل بودن (هم‌ارزی) اتوماتا بوده و مطالبی که در اینجا ارائه می‌کنیم و از جمله قضیه فوق، امکان بحث دقیق در مورد این فرآیند و همچنین اثبات قضایای مربوط به هم‌ارزی را فراهم می‌کند. در مبحث بعد، به دفعات از تعریف‌های جدید استفاده خواهیم کرد، اما همه چیز را به صورت تفصیلی و دقیق شرح نداده و ادامه بسیاری از مطالب را بر عهده خواننده واگذار می‌کنیم. شبیه‌سازی کامل بوسیله ماشین‌های تورینگ غالباً طولانی و خسته‌کننده است. به این منظور، به جای پی‌گیری روند سابق قضیه- اثبات، به ارائه توضیحات اکتفا می‌کنیم. هرچند فقط کلیات شبیه‌سازی‌ها را ارائه می‌کنیم، به راحتی می‌توان با صرف کمی تلاش و دقت اثبات‌های مربوطه را کامل کرد. توصیه می‌شود خوانندگان عزیز برای درک بهتر مطالب، هریک از شبیه‌سازی‌ها را در یک زبان سطح بالاتر یا شبه‌کد در نظر بگیرند.

پیش از معرفی بقیه مدل‌ها، لازم است یک نکته را در مورد ماشین‌های تورینگ استاندارد روشن کنیم. از تعریف ۹-۱ به صورت ضمنی می‌توان دریافت که هر یک از سمبل‌های نوار، ترکیبی از چندین کاراکتر است نه فقط یک کاراکتر. همچنین این نکته را می‌توان با ترسیم نسخه گسترش‌یافته‌ای از شکل ۹-۱ (یعنی شکل ۱۰-۱) که علائم نوار آن سه‌تایی‌های الفبای ساده‌تر دیگری باشند، به راحتی مشاهده کرد.

در این تصویر، هریک از سلول‌های نوار به سه بخش به نام **شیار** تقسیم شده و هر شیار حاوی یکی از اعضای سه‌تایی فوق‌الذکر می‌باشد. از اینرو، این دست اتوماتا را می‌توان ماشین تورینگ **چندشیاری** خواند. اما این تغییر به هیچ وجه تعریف ۱-۹ را گسترش نمی‌دهد، چون فقط لازم است Γ را الفبایی تصور کنیم که هریک از سمبل‌های آن از چندین بخش تشکیل شده است.

با این وجود، در دیگر مدل‌های ماشین تورینگ باید تعریف را تغییر داده و سپس، هم‌ارزی آنها با ماشین‌های استاندارد را اثبات کنیم. در ادامه دو مدل مشابه این مدل را بررسی می‌کنیم که گاهی اوقات بعنوان تعریف استاندارد مورد استفاده قرار می‌گیرند. در ادامه، با برخی از گونه‌های نه چندان متعارف در تمرینات انتهای همین بخش آشنا خواهید شد.



شکل ۱۰-۱

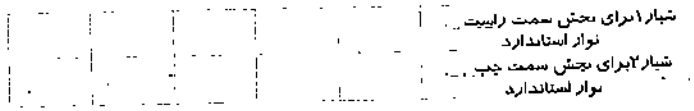
ماشین‌های تورینگ با نوار نیمه‌نامتناهی

بسیاری از صاحبنظران مدل ارائه شده در شکل ۹-۱ را استاندارد نمی‌دانند و از اینرو، از مدل تورینگ تک‌نواری استفاده می‌کنند که فقط از یک طرف نامحدود است. می‌توان این نوار را دارای یک حد چپ در نظر گرفت (شکل ۱۰-۲). ماشین تورینگ مذکور از جهاتی مشابه مدل استاندارد ما بوده و

شکل ۱۰-۲

تنها استثناء این است که وقتی هد خواندن- نوشتن در انتها قرار می‌گیرد، حرکت به چپ به هیچ وجه مجاز نمی‌باشد.

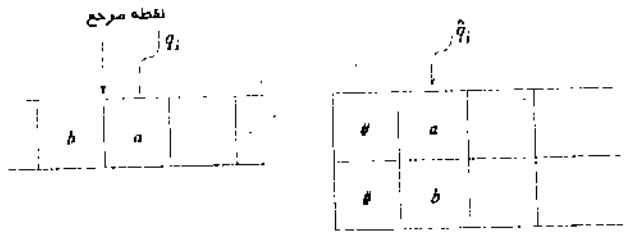
اما این محدودیت هیچ تأثیری بر قدرت ماشین نمی‌گذارد. برای شبیه‌سازی ماشین تورینگ مفروض M با ماشین نوار نیمه نامتناهی $\hat{M}$، از تنظیماتی مشابه شکل ۱۰-۳ استفاده می‌کنیم.



شکل ۱۰-۳

نوار ماشین شبیه‌ساز $\hat{M}$ دو شیار دارد. در شیار بالایی، اطلاعات را در سمت راست یک نقطه مرجع مفروض روی نوار M نگهداری می‌کنیم. این نقطه مرجع را می‌توان مثلاً موقعیت هد خواندن- نوشتن در ابتدای محاسبه درنظر گرفت. شیار پایینی حاوی اطلاعات بخش چپ نوار M، اما درجهت عکس، می‌باشد. برنامه‌ریزی ویژه $\hat{M}$ به گونه‌ای است که فقط تا زمانی قادر به استفاده از اطلاعات روی شیار بالایی می‌باشد که هد خواندن- نوشتن M در سمت راست نقطه مرجع قرار داشته باشد و زمانی به سراغ شیار پایینی می‌رود که M به سمت چپ نوار خود حرکت کند. این وضعیت را می‌توان با تقسیم مجموعه حالت $\hat{M}$ به دو بخش Q_L و Q_U نمایش داد:

از بخش اول هنگام کار روی شیار بالایی و از بخش دیگر در هنگام کار روی شیار پایینی استفاده می‌شود. نشانه‌های ویژه پایان # روی حد چپ نوار قرارمی‌گیرند تا انتقال از یک شیار به شیار دیگر را تسهیل کنند. بعنوان مثال، فرض کنید ماشینی که باید شبیه‌سازی شود و ماشین شبیه‌ساز به ترتیب دارای پیکربندی‌های نشان داده‌شده در شکل ۱۰-۴ باشند و حرکتی که باید شبیه‌سازی شود بوسیله



شکل ۱۰-۴

$$\delta(q_i, a) = (q_j, c, L)$$

تعریف شده است. ماشین شبیه‌ساز ابتدا با انتقال

$$\hat{\delta}(\hat{q}_i, (a,b)) = (\hat{q}_j, (c,b), L),$$

با ضابطه $\hat{q}_i \in Q_U$، حرکت می‌کند. بدلیل آنکه $\hat{q}_i$ به Q_U تعلق دارد، فقط اطلاعات موجود در شیار فوقانی مد نظر قرار می‌گیرد. سپس، ماشین شبیه‌ساز در حالت $\hat{q}_j \in Q_U$ با (#، #) برخورد می‌کند. در مرحله بعد، از انتقال

$$\hat{\delta}(\hat{q}_j, (\#,\#)) = (\hat{p}_j, (\#,\#), R),$$

با ضابطه $\hat{p}_j \in Q_L$ استفاده کرده و انتقال مذکور را در پیکربندی شکل ۱۰-۵ قرار می‌دهد. اینک، ماشین در حالتی از Q_L قرار داشته و روی شیار پایینی عمل می‌کند. مابقی جزئیات شبیه‌سازی به راحتی قابل پیش‌بینی و درک است.

شکل ۱۰-۵

ماشین تورینگ آف‌لاین(off-line)

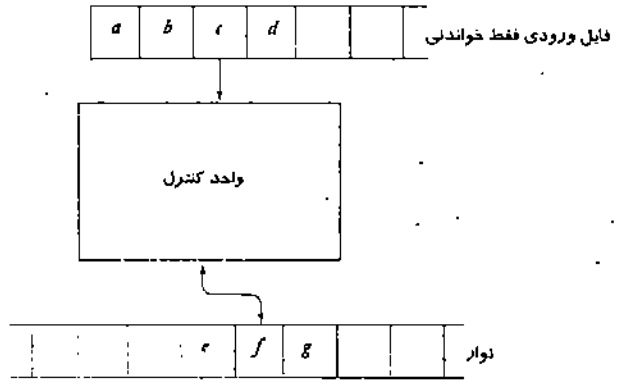
در تعریف کلی اتومات که در فصل یک ارائه شد، از یک فایل ورودی و یک منبع ذخیره‌سازی موقت صحبت کردیم. در تعریف ۹-۱، برای ساده‌کردن بحث، گزینه فایل ورودی را حذف کرده و ادعا کردیم که با این کار هیچ تفاوتی در مفهوم ماشین‌های تورینگ ایجاد نخواهد شد. حال این ادعا را توضیح و بسط خواهیم داد.

اگر فایل ورودی را مجدداً در تصویر مربوط به تعریف قرار دهیم، مدلی به نام **ماشین تورینگ آف‌لاین(off-line)** حاصل خواهد شد. در این نوع ماشین‌ها، تمامی حرکت‌ها توسط حالت درونی، سمبلی که در حال حاضر از فایل ورودی خوانده می‌شود و همچنین آنچه بوسیله هد خواندن- نوشتن مشاهده می‌شود، تصمیم‌گیری می‌شود. یکی از نمایش‌های کلی یک ماشین‌های آف‌لاین را در شکل ۱۰-۶ مشاهده می‌کنید. هرچند به راحتی می‌توان تعریف صوری برای ماشین‌های تورینگ آف‌لاین ارائه داد، این کار را بعنوان تمرین به خواننده واگذار می‌کنم. در ادامه به بیان دلیل هم‌ارزی بین دسته‌های ماشین‌های تورینگ آف‌لاین با دسته ماشین‌های استاندارد می‌پردازیم.

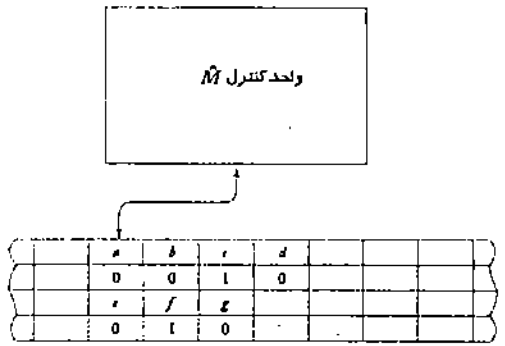
ابتدا می‌توان رفتار ماشین تورینگ استاندارد را بوسیله یک مدل آف‌لاین دلخواه شبیه‌سازی کرد. به

این منظور، فقط لازم است که ماشین شبیه‌ساز، ورودی را از فایل ورودی به نوار کپی کند. مابقی مراحل مشابه ماشین‌های استاندارد معمولی است.

اما حالت عکس، یعنی شبیه‌سازی ماشین آف‌لاین M توسط ماشین استاندارد $\hat{M}$ به کمی توضیح نیاز دارد. یک ماشین استاندارد می‌تواند با استفاده از تنظیم چهار شیاری شکل ۱۰-۷، اقدام به شبیه‌سازی محاسبه در ماشین‌های آف‌لاین کند. در این تصویر، محتویات نوار نمایش داده شده، بیانگر پیکربندی خاصی از شکل ۱۰-۶ می‌باشد. هریک از چهار شیار $\hat{M}$ نقش خاصی را در شبیه‌سازی بر عهده دارند. در اولین شیار، ورودی قرار می‌گیرد، دومین شیار موقعیت خواندن ورودی را علامت‌گذاری می‌کند، شیار سوم بیانگر نوار M بوده و شیار چهارم موقعیت هد خواندن- نوشتن را نمایش می‌دهد.



شکل ۱۰-۶



شکل ۱۰-۷

شبیه‌سازی هر یک از حرکات M مستلزم چندین حرکت $\hat{M}$ است. با شروع از یک موقعیت استاندارد مفروض، مثل انتهای سمت چپ، و با اطلاعات مربوطه‌ای که بوسیله نشانه‌گرهای مخصوص انتهایی علامت‌گذاری شده‌اند، $\hat{M}$ با جستجو در شیار ۲ اقدام به مکان‌یابی محل خواندن فایل ورودی M می‌کند. سمبل نظیر سلول شیار ۱، با قرار دادن واحد کنترل $\hat{M}$ در یک حالت انتخابی به خاطر سپرده می‌شود. سپس، برای یافتن موقعیت هد خواندن – نوشتن M، شیار ۴ جستجو می‌شود. پس از ذکر ورودی و سمبل روی شیار ۳، می‌توان از فعالیت‌های M اطلاع حاصل کرد. این اطلاعات هم توسط یکی از حالات درونی مناسب $\hat{M}$ به خاطر سپرده می‌شود. سپس، هر چهار شیار نوار $\hat{M}$ اصلاح می‌شود تا حرکت M را منعکس کند. در پایان، هد خواندن- نوشتن $\hat{M}$ به موقعیت استاندارد بازگشته و برای شبیه‌سازی حرکت بعدی آماده می‌شود.

تمرین‌ها

۱. برای ماشین تورینگ مجهز به نوار نیمه‌نامتناهی، تعریف صوری ارائه دهید. سپس اثبات کنید که دسته ماشین‌های تورینگ مجهز به نوار نیمه نامتناهی، هم‌ارز با دسته ماشین‌های تورینگ استاندارد است.

۲. برای یک ماشین تورینگ آف‌لاین تعریف صوری ارائه دهید.

۳. با ارائه استدلال‌های کافی، ثابت کنید که تمامی زبان‌هایی که بوسیله یک ماشین تورینگ آف‌لاین پذیرفته می‌شوند، در یک ماشین استاندارد هم پذیرفته می‌شوند.

۴. ماشین تورینگی را در نظر بگیرید که می‌تواند در هر یک از حرکات، یا سمبل نوار را تغییر دهد و یا هد خواندن- نوشتن را جابجا کند، اما قادر به انجام هر دو کار نیست.

الف) برای این دست ماشین‌ها تعریف صوری ارائه دهید.

ب) نشان دهید که این‌چنین دسته ماشین‌های، هم ارز با دسته ماشین‌های تورینگ استاندارد است.

۵. مدلی از یک ماشین تورینگ در نظر بگیرید که با هر حرکت، هد خواندن- نوشتن بیش از یک سلول به چپ یا راست جابجا شده و فاصله و جهت جابجایی توسط تعریف δ تعیین می‌شود. تعریف مختصری از این گونه اتومات ارائه داده و آنرا توسط یک ماشین تورینگ استاندارد شبیه‌سازی کنید.

۶. ماشین تورینگ پاک‌نشدنی، ماشینی است که قادر به تبدیل سمبل‌های غیرخالی به خالی نمی‌باشد. برای ایجاد این ویژگی، محدودیت زیر را قائل می‌شویم:

اگر

$$\delta(q_i, a) = (q_j, \square, L \text{ or } R),$$

آنگاه a باید حتماً □باشد. نشان دهید که اعمال این محدودیت، هیچ خدشه‌ای به کلیت تعریف وارد نمی‌کند.

۷. ماشین تورینگی را در نظر بگیرید که قادر به نوشتن خالی‌ها نباشد؛ یعنی به ازای تمام

$\delta(q_i,a)=(q_j,b,L\ or\ R)$، b باید در $\Gamma-\{\sqcup\}$ وجود داشته باشد. این ماشین چگونه قادر به شبیه‌سازی یک ماشین تورینگ استاندارد می‌باشد؟

۸. به فرض، این شرط را قائل می‌شویم: یک ماشین تورینگ فقط در حالت‌های پایانی قادر به توقف است؛ یعنی الزام کنیم که $\delta(q,a)$ برای تمام زوج‌های (q,a) با ضابطه $a\in\Gamma$ و $q\notin F$ تعریف شود. آیا این امر قدرت ماشین تورینگ را کاهش می‌دهد؟

۹. به‌فرض این محدودیت را قائل می‌شویم که یک ماشین تورینگ باید همواره سمبلی متفاوت از سمبل خوانده شده را بنویسد، یعنی اگر

$$\delta(q_i,a)=(q_j,b,L\ or\ R),$$

آنگاه، a و b باید متفاوت باشند. آیا بکاربستن این محدودیت قدرت اتومات را کاهش می‌دهد؟

۱۰. نسخه‌ای از یک ماشین تورینگ استاندارد را در نظر بگیرید که انتقالات آن، علاوه بر سلولی که مستقیماً زیر هد خواندن- نوشتن قرار گرفته، به سلول‌های بلافاصله واقع در چپ و راست هد هم بستگی داشته باشد. تعریف صوری از این نوع ماشین ارائه داده و سپس، روش کلی شبیه‌سازی آنرا بوسیله یک ماشین تورینگ استاندارد ارائه کنید.

۱۱. یک ماشین تورینگ با فرآیند تصمیم‌گیری متفاوتی را در نظر بگیرید که انتقالات آن در صورتی انجام می‌شود که سمبل فعلی نوار از میان مجموعه‌های مشخص‌شده انتخاب نشود. بعنوان مثال،

$$\delta(q_i,\{a,b\})=(q_j,c,R)$$

در صورتی باعث حرکت مذکور خواهد شد که سمبل فعلی نوار هیچ‌کدام از سمبل‌های a یا b نباشد. این مفهوم را با تعریف صوری ارائه کرده و نشان دهید که ماشین اصلاح‌شده هم ارز با یک ماشین تورینگ استاندارد است.

ماشین‌های تورینگ با حافظه پیچیده‌تر

ابزار ذخیره‌سازی ماشین‌های تورینگ استاندارد آنقدر ساده است که شاید تصور کنید می‌توان با استفاده از ابزارهای ذخیره‌سازی پیچیده‌تر قدرت آنها را هم افزایش داد. دلایل نادرست بودن این تصور را با ذکر دو مثال ارائه خواهیم داد.

ماشین‌های تورینگ چندنواره

ماشین تورینگ چندنواری، ماشین تورینگی با چندین نوار است که هر نوار، دارای هد خواندن - نوشتن می‌باشد که به‌طور مستقل کنترل می‌شود (شکل ۱۰-۸).

تعریف صوری یک ماشین تورینگ چندنواری بوسیله تعریف ۹-۱ امکانپذیر نبوده و از اینرو باید تابع انتقال آنرا اصلاح کرد. نوعاً، یک ماشین n نواری را با $M=(Q,\Sigma,\Gamma,\delta,q_0,\sqcup,F)$ تعریف می‌کنیم که در آن، Q,Σ,Γ,q_0,F مشابه تعریف ۹-۱ می‌باشند، اما

$$\delta: Q\times\Gamma^n\to Q\times\Gamma^n\times\{L,R\}^n$$

بیانگر رویدادهایی است که در تمامی نوارها رخ می‌دهد. به عنوان مثال، اگر $n=2$، با ضابطه پیکربندی مشابه شکل ۱۰-۸ آنگاه

$$\delta(q_0,a,e)=(q_1,x,y,L,R)$$

به صورت زیر تفسیر می‌شود:

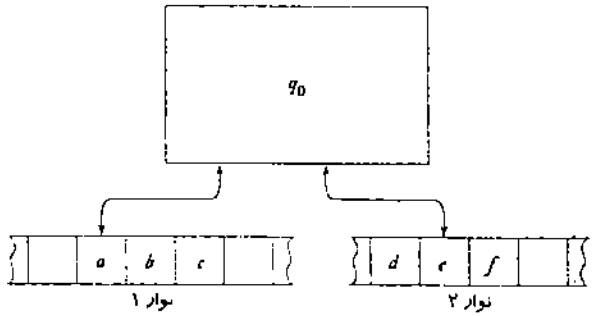
قانون انتقال فقط در صورتی قابل اعمال است که ماشین در حالت q_0 بوده و اولین هد خواندن- نوشتن یک a را و دومین هد هم یک e را ببیند. سپس، سمبل روی اولین نوار با یک x جایگزین شده و هد خواندن- نوشتن آن به سمت چپ حرکت خواهد کرد. در عین حال، سمبل روی نوار دوم به y تغییر یافته و هد خواندن- نوشتن به سمت راست حرکت می‌کند. پس از این کار، واحد کنترل به q_1 تغییر حالت داده و ماشین پیکربندی جدید شکل ۱۰-۹ را به خود می‌گیرد.

برای اثبات هم‌ارزی بین ماشین‌های تورینگ استاندارد و چندنواره، به این صورت بحث می‌کنیم که هر ماشین تورینگ چندنواره مفروض M را می‌توان بوسیله یک ماشین تورینگ استاندارد $\hat{M}$ شبیه‌سازی کرد و برعکس، هر ماشین تورینگ استاندارد را می‌توان بوسیله یک ماشین تورینگ چندنواره شبیه‌سازی کرد. چون همواره می‌توان یک ماشین چندنواره را با نواری راه‌اندازی کرد که قابلیت بیشتری نسبت به دیگری داشته باشد، بخش دوم این ادعا روشن بوده و نیازی به توضیح ندارد. شبیه‌سازی یک ماشین چندنواره بوسیله ماشین یک‌نواری گرچه کمی مشکل‌تر، اما از نظر مفهومی ساده و قابل فهم است.

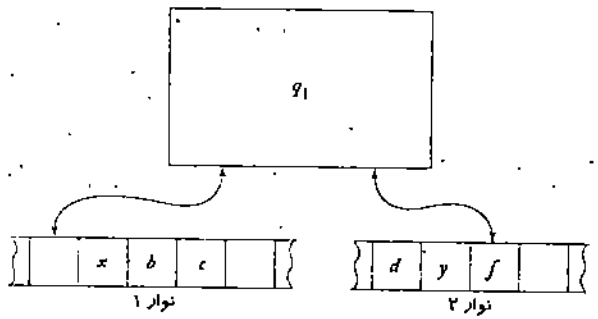
بعنوان مثال، ماشین دونواره‌ای را با پیکربندی مشابه شکل ۱۰-۱۰ در نظر بگیرید. ماشین شبیه‌ساز تک‌نواره، چهار شیارخواهد داشت(شکل ۱۰-۱۱). اولین شیار نمایشگر محتویات نوار۱ ماشین M خواهد بود. در تمامی بخش غیرخالی شیار دوم، به استثنای فقط یک سمبل1 که موقعیت هد خواندن- نوشتن M را نشان می‌دهد، صفر قرار می‌دهیم. شیار ۳ و ۴ نقشی مشابه شیار ۲ ماشین M ایفا می‌کنند. با بررسی شکل ۱۰-۱۱ می‌توان اظهار داشت که به ازای پیکربندی‌های مربوط به $\hat{M}$ (یعنی پیکربندی‌هایی با شکل مذکور)، فقط یک پیکربندی نظیر در M وجود دارد.

نمایش یک ماشین‌های چندنواره بوسیله یک ماشین تک نواره مشابه آن‌چیزی است که قبلاً در مورد شبیه‌سازی ماشین آف‌لاین گفتیم. مراحل عملی شبیه‌سازی هم تا حد زیادی مشابه بوده و تنها تفاوت، مربوط به افزایش تعداد نوارهاست. طرح کلی ارائه شده برای شبیه‌سازی ماشین‌های آف‌لاین با چندین اصلاحات جزئی در این مورد هم قابل اعمال است. بوسیله این طرح می‌توان تابع انتقال $\hat{\delta}$ مربوط به $\hat{M}$ را از تابع انتقال δ مربوط به M بدست آورد. جزئیات روش ساخت علیرغم طولانی بودن، ساده و قابل‌فهم است. مطمئناً، ظاهر طولانی و تفصیلی محاسبات $\hat{M}$ هیچ تأثیری بر نتیجه کار ندارد. بنابراین، هر آنچه که بتوان در مورد M انجام داد، برای $\hat{M}$ هم قابل انجام است.

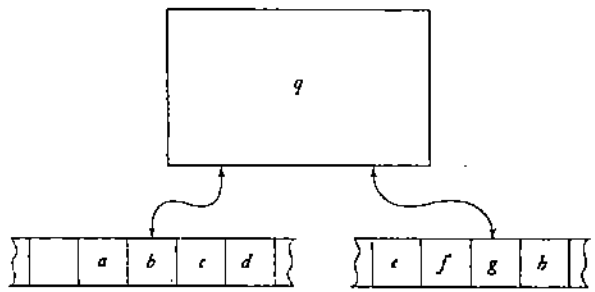
باید یک نکته مهم را به خاطر داشته باشید: وقتی ادعا می‌کنیم که یک ماشین تورینگ چند نواره



شکل ۱۰-۸

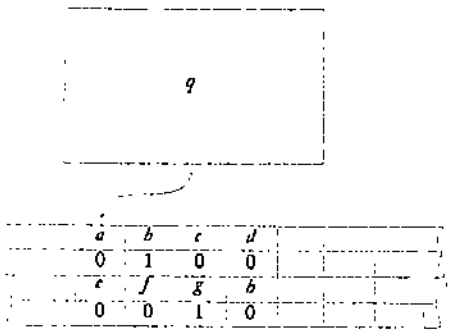


شکل ۱۰-۹



شکل ۱۰-۱۰

قدرتمندتر از یک ماشین استاندارد نیست، در واقع فقط در مورد قابلیت این ماشین‌ها و خصوصاً زبان مورد پذیرش آنها اظهار نظر کرده‌ایم.



شکل ۱۰-۱۱

مثال ۱۰-۱

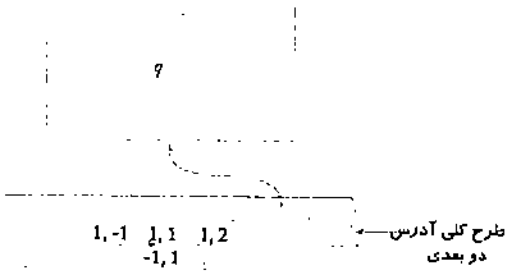
زبان $\{a^n b^n\}$ را در نظر بگیرید. در مثال ۹-۷، روش طولانی و خسته‌کننده‌ای را برای پذیرش این زبان توسط یک ماشین تورینگ یک‌نواره ارائه کردیم. استفاده از یک ماشین دو نواره کار را بسیار ساده‌تر می‌کند. فرض کنید که در آغاز محاسبه، رشته شروع $a^n b^m$ روی نوار ۱ نوشته شده باشد. سپس تمامی a ها را خوانده و آنها را به نوار ۲ کپی می‌کنیم. پس از اتمام a ها، b های روی نوار ۱ را با a های کپی شده روی نوار ۲ تطابق می‌دهیم. به این ترتیب، می‌توان بدون نیاز به جابجایی متوالی هد خواندن-نوشتن به جلو و عقب، تعیین کرد که آیا تعداد a ها و b ها برابر هستند یا خیر.

※

به خاطر داشته باشید که مدل‌های مختلف ماشین‌های تورینگ، فقط از نظر توانایی خود در انجام عملیات‌ها، هم‌ارز خوانده می‌شوند، نه براساس سهولت برنامه‌ریزی یا هر مقیاس دیگری از کارآیی.

ماشین‌های تورینگ چندبعدی

ماشین تورینگ چندبعدی ماشینی است که نوار آن به صورت نامتناهی در بیش از یک بعد گسترش یافته است. نموداری از یک ماشین تورینگ دوبعدی را در شکل ۱۰-۱۲ مشاهده می‌کنید.



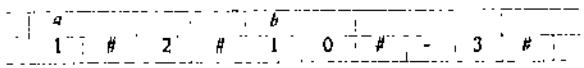
شکل ۱۰-۱۲

در تعریف رسمی یک ماشین تورینگ دوبعدی از تابع انتقال δ به شکل

$$\delta: Q\times\Gamma \to Q\times\Gamma\times\{L,R,U,D\},$$

استفاده می‌شود که در آن، U و D به ترتیب بیانگر حرکت به بالا و پایین هد خواندن- نوشتن هستند. برای شبیه‌سازی این ماشین روی یک ماشین تورینگ استاندارد، می‌توان از مدل دو شیاری مشابه شکل ۱۰-۱۳ کمک گرفت. اولاً، یک مرتب‌سازی یا آدرس را به سلول‌های نوار دوبعدی نسبت می‌دهیم. این کار به روش‌های مختلف، از جمله روش دوبعدی شکل ۱۰-۱۲، قابل انجام است. نوار دو شیاری ماشین شبیه‌ساز، از یک شیار برای ذخیره‌سازی محتویات سلول و از دیگری جهت نگهداری آدرس مربوطه استفاده می‌کند. در طرح‌کلی شکل ۱۰-۱۲، پیکربندی وجود a درسلول (۲و۱) و وجود b در سلول (۳- و۱۰) را مشاهده می‌کنید. شکل ۱۰-۱۳ پیکربندی مربوطه را نمایش می‌دهد. اما یک مشکل وجود دارد: آدرس سلول، ممکن است حاوی اعداد صحیح با طول دلخواه باشد، بنابراین شیار آدرس نمی‌تواند جهت ذخیره‌سازی آدرس‌ها، از یک فیلد با ابعاد ثابت استفاده کند. بلکه باید مطابق آنچه در تصویر ملاحظه می‌کنید، جهت محدوده‌سازی فیلدها، از روش تنظیم متغیر ابعاد فیلد با استفاده از برخی سمبل‌های ویژه استفاده نماید.

حال فرض می‌کنیم که در آغاز شبیه‌سازی هر یک از حرکات، هد خواندن- نوشتن ماشین دو بعدی M و هد خواندن- نوشتن ماشین شبیه‌ساز $\hat{M}$ همواره روی سلول‌های نظیر قرار داشته باشند. برای شبیه‌سازی یک حرکت، ابتدا ماشین شبیه‌ساز $\hat{M}$ آدرس سلولی را که M باید به آن انتقال پیدا کند، محاسبه می‌نماید. این‌کار با استفاده از آدرس‌دهی دوبعدی به راحتی قابل انجام است. پس از محاسبه آدرس، $\hat{M}$ اقدام به یافتن سلول حاوی این آدرس روی شیار ۲ کرده و محتویات سلول را به نحوی تغییر می‌دهد تا حرکت M را شامل شود. اکنون دیگر با معلوم بودن M، به راحتی می‌توان اقدام به ساخت $\hat{M}$ کرد.



شکل ۱۰-۱۳

تمرین‌ها

هدف بخش عمده‌ای از مباحث ما در مورد ماشین‌های تورینگ معتبرسازی تز تورینگ از طریق اثبات شبیه‌سازی وضعیت‌های به ظاهر پیچیده‌تر روی یک ماشین تورینگ استاندارد است. متأسفانه، شبیه‌سازی دقیق، بسیار خسته‌کننده و دارای مفاهیم تکراری است. در تمرینات زیر، شبیه‌سازی‌ها را فقط به اندازه‌ای شرح دهید که از عملکرد درست جزئیات مطمئن شوید.

۱. یک ماشین تورینگ آف‌لاین با چند هد را تعریف کرده و نحوه شبیه‌سازی آنرا بوسیله یک ماشین تورینگ استاندارد شرح دهید.

۲. یک ماشین تورینگ با چند هد را می‌توان یک ماشین تورینگ دارای یک‌نوار و یک واحد کنترل، ولی با چند هد خواندن- نوشتن جداگانه، در نظر گرفت. تعریف صوری از یک ماشین تورینگ با چند هد ارائه کرده و سپس نشان دهید که چگونه می‌توان این دست ماشین‌ها را با یک ماشین تورینگ استاندارد شبیه‌سازی کرد.

۳. تعریف صوری از یک ماشین تورینگ با چند هد- چند نواره ارائه کنید. سپس نشان دهید چگونه می‌توان این ماشین را با یک ماشین تورینگ استاندارد شبیه‌سازی کرد.

۴. تعریف صوری از یک ماشین تورینگ با یک‌نواره و چند واحد کنترل ارائه دهید که هریک دارای فقط یک هد خواندن- نوشتن می‌باشد. نشان دهید چگونه می‌توان این ماشین را با یک ماشین چندنواره شبیه‌سازی کرد.

۵. • - یک **اتومات صفی**، اتوماتی است که حافظه آن صف باشد. فرض کنید که این ماشین، یک ماشین آنلاین باشد، یعنی فاقد هرگونه فایل‌های ورودی بوده و رشته مورد پردازش پیش از شروع محاسبه در صف قرار داده می‌شود. تعریف صوری از این اتومات ارائه داده و سپس، در مورد قدرت آن در قیاس با ماشین‌های تورینگ بحث کنید.

۶. • - نشان دهید که به ازای هر ماشین تورینگ، یک ماشین تورینگ استاندارد حداکثر شش حالتی نظیر وجود دارد.

۷. • - تعداد حالت‌های لازم در تمرین ۶ را حدالامکان کاهش دهید (راهنمایی: کمترین تعداد ممکن سه است).

۸. • - شمارنده، یک پشته با الفبای صرفاً دو سمبلی شامل سمبل شروع پشته و یک سمبل شمارنده است. فقط سمبل شمارنده را می‌توان روی پشته قرار داد یا از آن حذف کرد. **اتومات شمارنده**، اتومات معین است که بعنوان حافظه از یک یا چند شمارنده استفاده می‌کند. نشان دهید که هر ماشین تورینگی را می‌توان با استفاده از یک اتومات شمارنده دارای چهار شمارنده شبیه‌سازی کرد.

۹. نشان دهید که تمامی محاسباتی که بوسیله یک ماشین استاندارد قابل انجام است، توسط یک ماشین چندنواره ایستا و حداکثر دو حالتی هم قابل انجام است.

۱۰. برنامه مفصلی برای محاسبه مثال ۱۰-۱ بنویسید.

۱۰-۳ ماشین‌های تورینگ نامعین

هرچند با بررسی نظریه تورینگ با اطمینان می‌توان گفت که ساختار ویژه نوار تأثیری بر قدرت ماشین تورینگ ندارد، ولی با همین اطمینان نمی‌توان در مورد تأثیر نامعین‌بودن اظهار نظر کرد. بدلیل آنکه نامعین‌بودن امری انتخابی بوده و رنگ و بوی نامعین‌بودن را در خود دارد، از کاربرد نظریه تورینگ صرفنظر می‌کنیم. اگر بخواهیم با اطمینان بگوییم که نامعین‌بودن هیچ تأثیری بر قدرت ماشین تورینگ

ندارد، باید ابتدا این تأثیر را با دید بازتری مورد مطالعه قرار دهیم. مجدداً از شبیه‌سازی کمک گرفته و نشان می‌دهیم که رفتار نامعین را می‌توان به طور معین تحت کنترل و اختیار گرفت.

تعریف ماشین تورینگ نامعین مشابه تعریف اتوماتی است که در تعریف ۹-۱ ارائه شد، با این استثنا که به جای تابع δ آن از

$$\delta : Q \times \Gamma \to 2^{Q \times \Gamma \times \{L,R\}}$$

استفاده می‌شود. مانند موارد قبل، وقتی صحبت از نامعین‌بودن به میان می‌آید، برد δ مجموعه تمام انتقالات ممکنی است که بوسیله ماشین انتخاب می‌شوند.

مثال
یک ماشین تورینگ با انتقالاتی به فرم

$$\delta(q_0, a) = \{(q_1, b, R), (q_2, c, L)\}$$

نامعین است. هر دو حرکت

$$q_0aaa \mapsto bq_1aa$$

و

$$q_0aaa \mapsto q_2\square caa$$

امکان‌پذیر می‌باشد.

□

بدلیل آنکه نقش دقیق نامعین‌بودن در محاسبه توابع هنوز مشخص نیست، اتوماتای نامعین معمولاً بعنوان پذیرنده تلقی می‌شوند. هنگامی می‌گوییم یک ماشین تورینگ نامعین، w را می‌پذیرد که دنباله امکان‌پذیری از حرکات به صورت

$$q_0w \mapsto x_1q_fx_2,$$

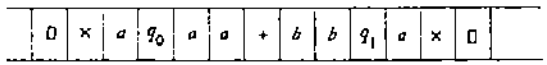
با ضابطه $q_f \in F$ وجود داشته باشد. در یک ماشین نامعین ممکن است حرکاتی وجود داشته باشد که به یک حالت غیرپایانی یا یک حلقه نامتناهی ختم می‌شود. اما، مشابه تمام موارد نامعین دیگر، این گزینه‌ها بلاتأثیر هستند. به بیان دیگر، ما فقط علاقه‌مند و پی‌گیر وجود دنباله‌هایی از حرکات قابل پذیرش هستیم.

برای اثبات اینکه یک ماشین تورینگ نامعین به هیچ وجه قدرتمندتر از نوع معین خود نیست، باید نظیر قطعی آن را، برای نامعین ایجاد کنیم. قبلاً یک مرتبه، تلویحاً به این روش اشاره کردیم. نامعین را

می‌توان یک الگوریتم بازگشت‌به‌عقب معین تلقی کرد. و یک ماشین معین تا زمانی می‌تواند یک ماشین نامعین را شبیه‌سازی کند که بتواند مراحل و اطلاعات نگهداری‌شده برای بازگشت‌به‌عقب را خوب رسیدگی و مدیریت کند. برای بررسی نحوه انجام این کار، نمای دیگری از نامعین را بررسی می‌کنیم که در بسیاری از استدلال‌ها مورد استفاده قرار می‌گیرد:

یک ماشین نامعین را می‌توان ماشینی تلقی کرد که در صورت لزوم، قادر به کپی برداری از خود می‌باشد. در مواقعی که بیش ازیک حرکت امکان‌پذیر باشد، ماشین به هر تعداد لازم کپی تهیه کرده و به هر یک از نسخه‌های کپی‌شده، وظیفه انجام یکی از انتخاب‌های حرکت خود را واگذار می‌کند. این برداشت از نامعین ممکن است نامعین به نظر برسد، چون کپی برداری نامحدود مسلماً از توان کامپیوترهای امروزی خارج است. با این وجود، می‌توان به ترتیب زیر از ایده شبیه‌سازی یاری گرفت.

یک روش برای نمایش شبیه‌سازی، استفاده از ماشین تورینگ استانداردی است که تمامی پیکربندی‌های ممکن از ماشین نامعین را روی نوار خود نگه می‌دارد. توجه داشته باشید که باید برای تفکیک این توصیف‌های لحظه‌ای یک قرارداد تنظیم و براساس آن عمل کرد. شکل ۱۰-۱۴، روشی را برای نمایش دو پیکربندی aq_0aa و bbq_1a ارائه می‌دهد. از نشانه‌های × برای تعیین محدوده دامنه مورد نظر و از نشانه + جهت تفکیک بین دو پیکربندی متوالی استفاده می‌شود. ماشین شبیه‌ساز تمامی پیکربندی‌های فعال را بررسی کرده و بز اساس برنامه ماشین نامعین، اقدام به بروزرسانی آنها می‌کند. ایجاد پیکربندی‌های جدید یا افزایش تعداد پیکربندی‌ها لحظه‌ای باعث جابجایی نشانه‌های × می‌شود. جزئیات این‌کار علیرغم خسته‌کنندگی، به راحتی قابل تصور و فهم است. براساس این شبیه‌سازی، نتیجه می‌گیریم که به ازای هر ماشین تورینگ نامعین ، یک ماشین تورینگ استاندارد معین نظیر وجود دارد.



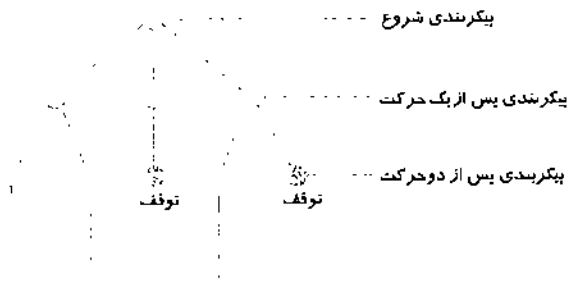
شکل ۱۰-۱۴

قضیه ۱۰-۲

دسته ماشین‌های تورینگ معین و دسته ماشین‌های تورینگ نامعین هم ارز هستند.

اثبات: با استفاده از روش قدم به قدم پیشنهادی فوق، نشان دهید که هر ماشین تورینگ نامعین را می‌توان بوسیله یک ماشین تورینگ معین شبیه‌سازی کرد. ■

چون بعداً به تأثیر نامعین بودن در کاربردهای عملی اشاره خواهیم کرد، بحث در این‌باره را به موقعیت مناسب دیگری موکول می‌کنیم. مانند چند مورد قبل، نامعین بودن را می‌توان بعنوان گزینه‌ای در بین چندین انتخاب در نظر گرفت. این حالت را می توان در قالب یک درخت تصمیم‌گیری مشابه شکل ۱۰-۱۵ به راحتی ارائه کرد.



شکل ۱۰-۱۵

عرض این گونه درخت‌های پیکربندی، بستگی به ضریب شاخه‌سازی، یعنی تعداد انتخاب‌های موجود برای هر حرکت، دارد. اگر k را حداکثر تعداد شاخه‌سازی در نظر بگیریم، آنگاه

$$M = k^n \qquad (۱۰-۱)$$

حداکثر تعداد پیکربندی‌های موجود پس از n حرکت می‌باشد.

حال، برای سهولت مباحث بعدی، تعریف پذیرش زبان را بیشتر توضیح داده و به مسأله عضویت هم اشاره خواهیم کرد.

تعریف ۱۰-۳

هنگامی گفته می‌شود ماشین تورینگ نامعین M زبان مفروض L را *می‌پذیرد* که این ماشین، به ازای تمام $w \in L$، حداقل یکی از پیکربندی‌های ممکن w را بپذیرد. برخی شاخه‌ها ممکن است پیکربندی‌های غیرقابل پذیرش ایجاد کرده و برخی دیگر حتی ماشین را در یک حلقه نامتناهی قرار دهند. اما این امر تأثیری بر پذیرش زبان ندارد.

هنگامی می‌گوییم ماشین تورینگ نامعین M به ازای تمام $w \in \Sigma^*$ برای زبان مفروض L تصمیم‌گیری می‌کند که، مسیری وجود داشته باشد که منجر به پذیرش یا رد زبان شود.

تمرین‌ها

۱. بطور مفصل در مورد شبیه‌سازی یک ماشین تورینگ نامعین بوسیله یک ماشین تورینگ معین توضیح دهید. به طور واضح بیان کنید که چگونه می‌توان ماشین‌های جدید را ایجاد نمود، ماشین‌های فعال را شناسایی کرد و ماشین‌های ساکن‌شده را در بررسی‌های بعدی نادیده گرفت.

۲. نشان دهید که چگونه می‌توان یک ماشین تورینگ نامعین دوبعدی را بوسیله یک ماشین معین شبیه‌سازی کرد.

۳. برنامه‌ای برای یک ماشین تورینگ نامعین بنویسید که زبان

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

را بپذیرد. بار دیگر، ماشین را معین فرض کرده و مجدداً برنامه‌ای برای آن بنویسید.

۴. به طور کلی شرح دهید که چگونه می‌توان برنامه‌ای برای ماشین تورینگی نوشت که زبان زیر را بپذیرد.

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

۵. برنامه ساده‌ای برای یک ماشین تورینگ نامعین بنویسید که زبان زیر را بپذیرد.

$$L = \{xww^R y : x, y, w \in \{a,b\}^+, |x| \geq |y|\}.$$

این مسأله را با یک ماشین تورینگ معین حل کنید.

۶. ماشین تورینگ نامعین را طراحی کنید که زبان زیر را بپذیرد.

$$L = \{a^n : n \text{ یک عدد اول نیست}\}.$$

۷. اتومات دو پشته‌ای یک اتومات pushdown نامعین با دو پشته مستقل است. برای تعریف این دست اتوماتا، تعریف ۷-۱ را به صورت زیر اصلاح می‌کنیم:

$\delta : Q \times (\Sigma \cup \{\lambda\}) \times \Gamma \times \Gamma \to$ زیرمجموعه‌های متناهی از $Q \times \Gamma^* \times \Gamma^*$

تمامی حرکات اتومات بستگی به بالای هر دو پشته داشته و در نتیجه هر حرکت، مقادیر جدیدی بر روی این دو پشته قرارمی دهد می‌کند. نشان دهید که دسته اتوماتای دوپشته‌ای هم ارز با دسته ماشین‌های تورینگ است.

۱۰-۴ ماشین تورینگ عمومی

بحث زیر را در مقابل نظریه تورینگ در نظر بگیرید:

"یک ماشین تورینگ، مطابق آنچه در تعریف ۹-۱ ارائه شد، کامپیوتر خاص منظوره است. اگر δ تعریف شده باشد، ماشین فقط قادر به انجام نوع خاصی از محاسبه است. در حالیکه، کامپیوترهای رقمی ماشین‌های همه‌منظوره هستند و می‌توان آنها را به نحوی برنامه‌ریزی کرد که در زمان‌های مختلف کارهای مختلفی را انجام دهند. در نتیجه، ماشین‌های تورینگ را نمی‌توان هم ارز با کامپیوترهای دیجیتال همه‌منظوره در نظر گرفت."

می‌توان با طراحی یک ماشین تورینگ قابل برنامه‌ریزی، موسوم به **ماشین تورینگ عمومی**، بحث فوق را نقض کرد. ماشین تورینگ عمومی M_u اتوماتی است که با در اختیار داشتن توصیف هر ماشین تورینگ M بعنوان ورودی و رشته w، قادر به شبیه‌سازی محاسبه M روی w می‌باشد. برای ساخت چنین M_u ، ابتدا روش استانداری برای توصیف ماشین‌های تورینگ پیدا می‌کنیم. ضمن حفظ کلیت مسئله، فرض می‌کنیم که

$$Q=\{q_1,q_2,\ldots,q_n\},$$

که در آن، q_1 حالت شروع و q_2 تنها حالت پایانی است؛ همچنین

$$\Gamma=\{a_1,a_2,\ldots,a_m\},$$

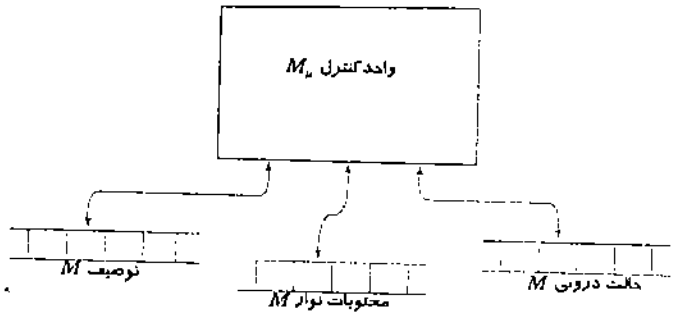
که در آن، a_1 بیانگر خالی می‌باشد. در روش رمزگذاری مورد استفاده ما، q_1 به صورت 1، q_2 به صورت 11 نمایش داده می‌شود و الی آخر. به همین ترتیب، a_1 به صورت 1، a_2 به صورت 11 رمزگذاری می‌شود و الی آخر. از علامت 0 برای تفکیک بین 1ها استفاده می‌کنیم. با در اختیار داشتن حالت شروع و پایان و تعریف خالی‌ها در این قرارداد، تمامی ماشین‌های تورینگ را می‌توان، صرفاً با در اختیار داشتن δ، به طور کامل تعریف کرد. تابع انتقال بر همین اساس رمزگذاری شده و استدلال‌ها و نتیجه در یک دنباله‌ای از پیش تعیین‌شده قرار می‌گیرند. بعنوان مثال، $\delta(q_1,a_2)=(q_2,a_1,L)$ به صورت زیر نمایش داده می‌شود:

$$\cdots 10110110111010\cdots.$$

بر این اساس می‌گوییم که هر ماشین تورینگی را می‌توان توسط رشته‌ای روی $\{0,1\}^*$ به صورت متناهی رمزگذاری کرد و بعلاوه، با در اختیار داشتن هریک از رمزگذاری‌های M می‌توان اقدام به رمزگشایی منحصربفرد آن کرد. برخی از رشته‌ها (مانند رشته 00011) اشاره‌ای به هیچ ماشین تورینگی نداشته و بنابراین می‌توان به جای آنها نقطه‌چین قرار داد تا از بروز هر مشکلی جلوگیری کرده باشیم. مطابق شکل ۱۰-۱۶، الفبای ورودی ماشین تورینگ عمومی مفروض M_u شامل $\{0,1\}$ بوده و ساختاری مشابه یک ماشین چند نواره دارد.

به ازای هر ورودی M و w، نوار ۱ تعریف رمزگذاری‌شده‌ای از M را نگهداری می‌کند. نوار ۲ حاوی محتویات نوار M و نوار ۳ حاوی حالت درونی M می‌باشد. M_u برای تعیین پیکربندی M، ابتدا به محتویات نوارهای ۲ و ۳ نگاه می‌کند. سپس با مراجعه به نوار ۱، عملیات M در این پیکربندی را بررسی می‌کند. در نهایت، نوارهای ۲ و ۳ به نحوی اصلاح می‌شوند که نتیجه حرکت را منعکس کنند.

به دلایل مختلف می‌توان یک ماشین تورینگ عمومی واقعی ایجاد کرد (مراجعه شود به ۱۹۷۸، Qualitz، Dennis، Denning)، اما فرآیند کار به هیچ وجه جالب و آموزنده نیست. از اینرو ترجیح می‌دهیم که به سراغ فرضیه تورینگ برویم. ناگفته پیداست که می‌توان با استفاده از یک زبان برنامه‌سازی اقدام به پیاده‌سازی این نوع ماشین کرد. در واقع، برنامه پیشنهادی در تمرین ۱ بخش ۹-۱ تجسمی از یک ماشین تورینگ عمومی در یک زبان سطح بالاتر بود. بنابراین، انتظار می‌رود که این کار بوسیله یک ماشین تورینگ استاندارد هم قابل انجام باشد. بر این اساس، به‌طور منطقی می‌توان ادعا کرد، ماشین تورینگی وجود دارد که با در اختیار داشتن هر برنامه‌ای، قادر به انجام محاسبات تعیین شده بوسیله آن برنامه بوده و بنابراین، مدل مناسبی از یک کامپیوتر همه‌منظوره می‌باشد.



شکل ۱۰-۱۶

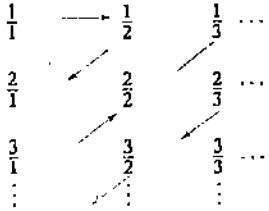
امکان نمایش ماشین‌های تورینگ بوسیله رشته‌های 0 و 1، اثرات جانبی مهمی دارد. اما پیش از بحث در این‌باره، لازم است که برخی نتایج ناشی از نظریه مجموعه‌ها را مرور کنیم.

برخی مجموعه‌ها، متناهی هستند، اما زبان‌ها و مجموعه‌های نامتناهی بسیار زیاد هستند. مجموعه‌های نامتناهی، به دو گروه مجموعه‌های **شمارش‌پذیر** و **شمارش‌پذیر** تقسیم می‌شوند. مجموعه‌ای شمارش‌پذیر خوانده می‌شود که اعضای آن در تناظر یک به یک با اعداد صحیح مثبت باشند. یعنی بتوان اعضای مجموعه را در هر ترتیبی، مثل $x_1,x_2,x_3,\ldots$، نوشت، بطوریکه هر عضو مجموعه دارای یک نشانهٔ شاخص متناهی باشد. بعنوان مثال، مجموعه تمام اعداد صحیح مثبت را می‌توان به ترتیب $0,2,4,\ldots$ نوشت. به‌دلیل آنکه هر عدد صحیح مثبت $2n$ در موقعیت $n+1$ رخ می‌دهد، این مجموعه شمارش‌پذیر است. تا به اینجا با هیچ مثال غیرقابل قبولی برخورد نکردیم. اما همواره این‌گونه نیست و برخی مثال‌های پیچیده‌تر، ممکن است برخلاف شهود به نظر برسند. مجموعه تمام خارج‌قسمتهای به فرم p/q را در نظر بگیرید که در آن، p و q اعداد صحیح مثبت هستند. چگونه می‌توان این مجموعه را به‌گونه‌ای نشان داد که شمارش‌پذیر باشد؟ مشخص است که نمی‌توان برای اینکار از دنباله

$$\frac{1}{1},\frac{1}{2},\frac{1}{3},\frac{1}{4},\cdots$$

استفاده کرد، چون در این دنباله هیچ جایی برای $\frac{2}{3}$ وجود ندارد. اما این نقطه ضعف، دلیلی برای شمارش‌ناپذیر محسوب شدن مجموعه فوق نمی‌باشد! برای حل این مسائل می‌توان با پیروی از یکی از روش‌های هوشمندانه جهت شمارش مجموعه، شمارش‌پذیر عملی آنرا هم اثبات کرد. نگاهی به طرح کلی مدلی از یک ۱۰-۱۷ بیندازید. اگر اعضاء را به همان ترتیبی بنویسیم که در اثر دنبال کردن فلش‌ها بوجود می‌آید، خواهیم داشت:

$$\frac{1}{1},\frac{1}{2},\frac{2}{1},\frac{3}{1},\frac{2}{2},\cdots.$$



شکل ۱۰-۱۷

در اینصورت، عضو $\frac{3}{5}$ جایگاه هفتم را به خود اختصاص داده و هر عضو محل ویژه‌ای را در این دنباله در اختیار دارد. بنابراین مجموعه فوق شمارش‌پذیر است.

براساس این مثال می‌گوییم که یک مجموعه در صورتی شمارش‌پذیر است که بتوان اعضای آنرا با اتخاذ هر روشی، با دنباله خاصی نوشت. این روش به نام روال شمارش خوانده می‌شود. بدلیل آنکه روال شمارش مدلی از یک فرآیند مکانیکی است، می‌توان برای تعریف صوری آن از مدل ماشین‌های تورینگ استفاده کرد.

تعریف ۱۰-۴

فرض کنید S مجموعه‌ای از رشته‌ها روی الفبای مفروض Σ باشد. آنگاه روال شمارش به ازای S ماشین تورینگی ایجاد می‌کند که دنباله مراحل

$$q_0\square \to q_1 x_1 \# s_1 \mapsto q_2 x_2 \# s_2 \ldots,$$

را با ضابطه $x_i \in \Gamma^* - \{\#\}$ ، $s_i \in S$ انجام می‌دهد، بطوریکه هر s واقع در S ، طی تعداد متناهی از مراحل تولید می‌شود. از حالت q_1 برای بیان عضویت در S استفاده می‌شود؛ یعنی در هنگام ورود q_1، رشته پس از $\#$ باید در S قرار داشته باشد.

کمی قبل‌تر گفتیم که تمامی مجموعه‌ها شمارش‌پذیر نبوده و مجموعه‌های شمارش‌ناپذیر هم وجود دارند. اما چون به کمک شمارش دنباله مورد نظر را ایجاد کردیم، نتیجه می‌گیریم هر مجموعه‌ای که یک روال شمارش به ازای آن وجود داشته باشد، شمارش‌پذیر می‌باشد.

به بیان دقیق‌تر، روال شمارش را نمی‌توان الگوریتم خواند؛ چون در صورت نامتناهی بودن S ، این روال هرگز به پایان نخواهد رسید. با این وجود، چون روال مذکور نتایج قابل پیش‌بینی و قابل تعریفی بوجود می‌آورد، می‌توان آنرا یک فرآیند معنادار تلقی کرد.

مثال ۱۰-۳

فرض کنید $\Sigma = \{a,b,c\}$ باشد. می‌توان نشان داد که $S = \Sigma^*$ شمارش‌پذیر است اگر بتوان روال شمارشی پیدا کرد که اعضای خود را به هر ترتیبی، مثلاً مطابق ترتیب فرهنگ لغت، تولید کند. با این

وجود، باید پیش از اعمال ترتیب فرهنگ لغت، تغییراتی در آن ایجاد کنیم. در این فرهنگ‌ها ، تمامی کلماتی که با a شروع می‌شوند، پیش از رشته b قرار می‌گیرند. اما اگر تعداد کلمات نامتناهی باشد، هیچگاه نوبت به b نمی‌رسد. در اینصورت، یکی از شروط تعریف ۱۰-۴، مبنی براینکه هر رشته مفروضی پس از تعداد متناهی از مراحل لیست شود، نقض می‌شود.

در عوض، می‌توان با تغییر ترتیب، طول رشته را بعنوان اولین ضابطه در نظر گرفت و سپس، اقدام به مرتب‌سازی الفبایی تمام رشته‌های هم‌طول نمود. این روال شمارشی دنباله زیر را ایجاد می‌کند:

$$a,b,c,aa,ab,ac,ba,bb,bc,ca,cb,cc,aaa,\ldots.$$

این نوع مرتب‌سازی که کاربردهای فراوانی هم دارد، **ترتیب محض** خوانده می‌شود.

■

یکی از نتایج مهم بحث فوق آن است که ماشین‌های توربنگ شمارش‌پذیر هستند.

قضیه ۱۰-۳

مجموعه تمام ماشین‌های تورینگ، هرچند نامتناهی، شمارش‌پذیر هستند.

اثبات: می‌توان تمامی ماشین‌های تورینگ را با استفاده از 0 و 1 رمزگذاری کرد. برای این روش رمزگذاری، روال شمارش زیر را ایجاد می‌کنیم.

۱. رشته بعدی $\{0,1\}^*$ را در ترتیب محض ایجاد کنید.

۲. بررسی کنید که آیا بوسیله رشته تولید شده می‌توان ماشین تورینگی را تعریف کرد یا خیر. در صورت مثبت بودن جواب، رشته را مطابق تعریف ۱۰-۴، روی نوار بنویسید. در غیر اینصورت، رشته را نادیده بگیرید.

۳. مراحل ۱ و ۲ را مجدداً تکرار کنید.

بدلیل آنکه تمام ماشین‌های تورینگ دارای یک توصیف متناهی می‌باشند، در نتیجه هر ماشینی را می‌توان بوسیله این فرآیند تولید کرد. ■

ترتیب به‌خصوص ماشین‌های تورینگ، بستگی به روش رمزگذاری مورد استفاده در آن دارد؛ در صورت تغییر روش رمزگذاری، باید انتظار ترتیب دیگری را داشته‌باشیم. اما چون این کار هیچ پیامدی به همراه ندارد، می‌توان نتیجه گرفت که نحوه مرتب‌سازی فاقد اهمیت بوده و فقط صرف وجود آن مهم است.

تمرین‌ها

۱. الگوریتمی را شرح دهید، که با بررسی هر رشته‌ای در $\{0,1\}^*$، تعیین کند که آیا رشته مورد نظر بیانگر یک ماشین تورینگ رمزگذاری شده است یا خیر.

۲. با استفاده از روش فوق، یک ماشین تورینگ با ضابطه‌های زیر را بطور کامل رمزگذاری کنید.

$$\delta(q_1,a_1)=(q_1,a_1,R),$$
$$\delta(q_1,a_2)=(q_3,a_1,L),$$
$$\delta(q_3,a_1)=(q_2,a_2,L).$$

۳. کلیات برنامه ماشین تورینگ را برای شمارش مجموعه $\{0,1\}^*$ در ترتیب محض ارائه کنید.

۴. شاخص 0^i1^j در تمرین ۳ چیست؟

۵. ماشین تورینگی را طراحی کنید که مجموعه زیر را در ترتیب محض شمارش کند.

$$L=\{a^nb^n : n\geq 1\}.$$

۶. در مثال ۱۰-۳، تابع $f(w)$ را پیدا کنید که به ازای هر w، شاخص مربوطه را در ترتیب محض ارائه دهد.

۷. نشان دهید که مجموعه تمام سه‌تایی‌های (i,j,k)، با ضابطه عدد صحیح مثبت بودن i,j,k، شمارش‌پذیر است.

۸. S_1 و S_2 را دو مجموعه شمارش‌پذیر فرض کنید. آنگاه نشان دهید که $S_1 \cup S_2$ و $S_1 \times S_2$ هم شمارش‌پذیر هستند.

۹. نشان دهید که حاصل‌ضرب دکارتی تعداد متناهی از مجموعه‌های شمارش‌پذیر، شمارش‌پذیر است.

۱۰-۵ اتوماتای کراندار خطی

هرچند نمی‌توان با پیچیده‌کردن ساختار نوار اقدام به افزایش قدرت ماشین‌های تورینگ استاندارد نمود، اما می‌توان با محدودسازی نحوه کاربرد نوار، دست به کاهش آن زد. قبلاً نمونه‌ای از این روش را در اتوماتای پشته‌ای اعمال کردیم. یک اتومات پشته‌ای را می‌توان یک ماشین تورینگ نامعین با نواری دانست که فقط باید بعنوان پشته مورد استفاده قرار گیرد. روش‌های دیگری هم برای محدودسازی کاربرد نوار وجود دارد. بعنوان مثال، فقط از بخش متناهی از نوار بعنوان فضای‌کاری استفاده نمود. می‌توان اثبات کرد که در اینصورت، ماشین تبدیل به یک اتوماتای متناهی خواهد شد (مراجعه شود به تمرین ۳ انتهای همین بخش). اتخاذ این روش جدید جهت محدودسازی کاربرد نوار در عین حال، موجب می‌شود تا ماشین فقط از بخشی از نوار که توسط ورودی اشغال شده، استفاده کند. بنابراین، برای رشته‌های ورودی طولانی فضایی بیشتر از رشته‌های کوتاه نیاز است، و به این ترتیب دسته دیگری از ماشین‌ها به نام اتوماتای کراندار خطی (یا lba) بوجود می‌آید.

اتومات کراندار خطی، مانند یک ماشین تورینگ استاندارد، دارای یک نوار نامحدود می‌باشد؛ اما مقدار قابل استفاده از نوار، تابع ورودی است. در یک حالت خاص، بخش قابل استفاده از نوار را دقیقاً

همان سلول‌هایی در نظر می‌گیریم که بوسیله ورودی اشغال می‌شوند[1]. جهت اجرا، می‌توان ورودی را با دو سمبل ویژه نشانه انتهای سمت چپ [و نشانه انتهای سمت راست] در داخل کروشه نمایش داد. به ازای ورودی w، پیکربندی شروع ماشین تورینگ بوسیله پیکربندی $q_0[w]$ ارائه می‌شود. نشانه‌های پایانی قابل بازنویسی نبوده و هد خواندن- نوشتن را نمی‌توان به سمت چپ [یا سمت راست] حرکت داد. گاهی اوقات می‌گوییم که هد خواندن- نوشتن از روی نشانه‌های انتهایی بیرون "نمی‌پرد".

تعریف ۱۰-۵

یک اتومات کراندار خطی، یک ماشین تورینگ نامعین $M=(Q,\Sigma,\Gamma,\delta,q_0,\square,F)$ مطابق تعریف ۱۰-۲، ولی با این محدودیت است که Σ باید حاوی دو نشانه ویژه [و] باشد، بطوریکه اعضای $\delta(q_i,[\,)$ فقط به فرم $(q_j,[,R)$ و اعضای $\delta(q_i,])$ فقط به فرم $(q_j,],L)$ باشند.

تعریف ۱۰-۶

رشته w بوسیله یک اتومات کراندار خطی پذیرفته می‌شود اگر به ازای برخی $q_f\in F, x_1,x_2\in\Gamma^*$، دنباله‌ای از حرکات

$$q_0[w]\mapsto[x_1q_fx_2]$$

امکان‌پذیر باشد. زبان مورد پذیرش بوسیله این lba مجموعه تمام رشته‌هایی است که به این ترتیب پذیرفته می‌شوند.

توجه داشته باشید که اتومات کراندار خطی در تعریف فوق، نامعین فرض می‌شود. این موضوع چندان کاربردی ندارد، اما در مبحث lba ها ضروری می‌باشد. هرچند lba های معین هم موجود و قابل تعریف هستند، هیچ روشی برای اثبات تناظر آنها با نسخه نامعین خود وجود ندارد. برای درک بهتر موضوع به تمرین ۸ انتهای این بخش مراجعه شود.

مثال ۱۰-۴

زبان

$$L=\{a^nb^nc^n : n\geq 1\}$$

بوسیله یک اتومات کراندار خطی مفروض پذیرفته می‌شود. برای اثبات می‌توان از بحث مثال ۹-۸

[1] در برخی از تعاریف، بخش قابل استفاده نوار ضریبی از طول ورودی است. بعلاوه، این ضریب به زبان بستگی دارد نه به ورودی. هرچند در این کتاب فقط طول دقیق رشته ورودی را مطرح و به عنوان مقیاس در نظر می‌گیریم، ماشین‌های چندشیاری که ورودی آنها فقط روی شیار قرار می‌گیرد، هم موجود و مورد استفاده قرار می‌گیرند.

استفاده کرد. برای انجام محاسبه‌ای که در آنجا گفته شد، فقط از ورودی اصلی استفاده کردیم و از اینرو بوسیله یک اتومات کراندار خطی هم قابل محاسبه است.

■

مثال ۱۰-۵

اتومات کراندار خطی پیدا کنید که زبان زیر را بپذیرد.

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

یک روش برای حل مسأله فوق این است که تعداد a ها را متوالیاً بر $2,3,4,\ldots$ تقسیم کنیم تا جاییکه بتوان یا رشته را پذیرفت و یا آنرا رد کرد. اگر ورودی در L وجود داشته باشد، نهایتاً فقط یک a باقی خواهد ماند؛ در غیر اینصورت، در جایی با یک باقیمانده غیرصفر مواجه خواهیم شد. کلیات راه‌حل را ارائه می‌کنیم تا بکی از پیاده‌سازی‌های دقیق تعریف ۱۰-۵ را به عینه مشاهده کنیم. بدلیل آنکه نوار یک اتومات کراندارخطی ممکن است چندشیاری باشد، از شیارهای اضافی می‌توان بعنوان فضای‌کاری استفاده کرد. در حل این مسأله، از یک نوار دو شیاری استفاده کرده‌ایم. اولین شیار حاوی تعداد a های باقیمانده پس از فرآیند تقسیم بوده و شیار دوم، حاوی تقسیم‌کننده جاری می‌باشد(شکل ۱۰-۱۸). جواب کامل مسأله تقریباً به راحتی بدست می‌آید. با استفاده از تقسیم‌کننده موجود در شیار دوم، تعداد a های واقع روی اولین شیار را تقسیم می‌کنیم. برای اینکار مثلاً بتوان تمامی سمبل‌ها را به استثنای آنهایی که در ضرایب تقسیم‌کننده وجود دارند، حذف نمود. سپس، تقسیم‌کننده را یک مرحله به پیش برده و این کار را آنقدر ادامه می‌دهیم تا یا به یک باقیمانده غیرصفر برسیم و یا فقط یک a باقی بماند.

a هایی که باید بررسی شوند	[	a	a	a	a	a	a	]
تقسیم‌کننده جاری	[	a	a	a			]	

شکل ۱۰- ۱۸

■

به‌دلیل آنکه هیچکدام از دو زبان فوق مستقل‌ازمتن نبودند، می‌توان نتیجه گرفت که اتوماتای کراندار خطی قدرتمندتر از اتوماتای پشته‌ای هستند. برای اثبات این ادعا، بازهم باید نشان دهیم که هر زبان مستقل‌ازمتنی را می‌توان بوسیله یک اتومات کراندار خطی پذیرفت. یک روش برای انجام اینکار در تمرینات ۶ و ۷ انتهای همین بخش ارائه شده است. اما طرح ادعا درباره ارتباط بین ماشین‌های تورینگ و اتوماتای کراندار خطی کمی مشکل‌تر است. به‌دلیل آنکه در مسائلی از قبیل مثال ۱۰-۵ مقدار فضای مجاز، متناسب با طول ورودی است، این دست مسائل را می‌توان بوسیله یک اتومات کراندار خطی هم حل کرد. در واقع، اکثریت زبان‌های ملموس و دارای تعریف مشخص توسط یکی از اتوماتای کراندار خطی مورد پذیرش قرار می‌گیرند. گرچه در برخی منابع اثبات شده که دسته اتوماتای کراندار خطی ضعیف‌تر از دسته ماشین‌های تورینگ نامحدود است، به همین دلیل این کار به راحتی قابل انجام نیست.

تمرین‌ها

۱. جزئیات راه‌حل مثال ۱۰-۵ را ارائه دهید.

۲. پاسخی برای مثال ۱۰-۵ پیدا کنید که نیازی به شیار دوم بعنوان فضای مجاز نداشته باشد.

۳. یک ماشین تورینگ آف‌لاین را در نظر بگیرید که ورودی آن فقط یک مرتبه قابل خواندن بوده، از چپ به راست حرکت می‌کند و قابل بازنویسی نباشد. این ماشین، روی نوار کار خود حداکثر فقط از n سلول اضافی بعنوان فضای کاری استفاده می‌کند و n برای تمام ورودی‌ها ثابت است. نشان دهید که ماشین مذکور متناظر با یک اتومات متناهی است.

۴. اتوماتای کراندار خطی را برای زبان‌های زیر پیدا کنید.

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

ب) $L = \{a^n : n$ یک عدد اول است $\}$.

ج) $L = \{a^n : n$ یک عدد اول نیست $\}$.

د) $L = \{ww : w \in \{a,b\}^+\}$.

ه) $L = \{w^n : w \in \{a,b\}^+, n \geq 1\}$.

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

۵. با این فرض که $\Sigma = \{a,b\}$ باشد، یک lba برای مکمل زبان مثال ۱۰-۵ پیدا کنید.

۶. نشان دهید که به ازای هر زبان مستقل‌ازمتن، یک pda پذیرنده وجود دارد بطوریکه تعداد سمبل‌های موجود در پشته را هیچگاه یک واحد بیشتر از طول رشته ورودی، افزایش نخواهد داد.

۷. با استفاده از مشاهده تمرین بالا نشان دهید که هر زبان مستقل‌ازمتن فاقد λ بوسیله یک اتومات کراندار خطی پذیرفته می‌شود.

۸. برای تعریف یک اتومات کراندار خطی معین، می‌توان از تعریف ۱۰-۵ برای حالت معین، استفاده کرد. پاسخ‌های تمرین ۴ را بررسی کنید. آیا در تمام پاسخ‌ها، اتوماتا کراندار خطی معین هستند؟ در غیر اینصورت، تمامی راه‌حل‌هایی را پیدا کنید که اتوماتای مربوطه کراندار خطی معین باشند.

فصل ۱۱

سلسله مراتب اتوماتا و زبان‌های صور

اینک به بحث اصلی مورد علاقه خود، یعنی مطالعه زبان‌های صوری، باز می‌گردیم. اولین هدف
بررسی زبان‌های مرتبط با ماشین‌های تورینگ و برخی از محدودیت‌های آنهاست. بدلیل آنکه
ماشین‌های تورینگ قادر به انجام انواع محاسبات الگوریتمی می‌باشند، خانواده زبان‌های مرتبط با آن
هم منطقاً باید بسیار گسترده باشند. این خانواده علاوه بر زبان‌های منظم و مستقل‌ازمتن، شامل انو
زبان‌های دیگری است که تا به اینجا مواجه شدیم و خارج از این خانواده‌ها قرار داشتند. سؤال مه
این است که آیا زبانی وجود دارد که توسط هیچ ماشین تورینگی پذیرفته نشود؟ برای پاسخ به ای
پرسش، می‌توان نشان داد که تعداد زبان‌ها، بیشتر از ماشین‌های تورینگ است و بنابراین به ازای برخ
زبان‌ها، هیچ ماشین تورینگی وجود ندارد. مراحل اثبات علیرغم سادگی و کوتاهی، فاقد نکات آموزن
است و از اینرو بینش خوبی از مسأله ایجاد نمی‌کند. به همین دلیل، با ارائه مثال‌های روشنی که عم
باعث شناسایی یکی از این دست زبان‌ها می‌شود، اقدام به تثبیت وجود زبان‌هایی می‌کنیم که بوس
ماشین‌های تورینگ قابل شناسایی نیستند. روش دیگر برای پاسخ به پرسش بالا، تحقیق در مو
ارتباط بین ماشین‌های تورینگ و برخی انواع گرامرها و مرتبط ساختن این گرامرها با گرامرهای منظ
و مستقل‌ازمتن است. به این ترتیب، مفهومی به نام سلسله مراتب گرامرها را معرفی کرده و از ای
رهگذر، روشی برای دسته‌بندی خانواده زبان‌ها ارائه می‌کنیم. برخی نمودارهای نظری مجموعه‌ه
روابط بین خانواده‌های زبان‌های مختلف را به وضوح نمایش می‌دهد.

به بیان دقیق‌تر، بسیاری از استدلال‌های این فصل فقط برای زبان‌های فاقد رشته تهی معتبر و موث
هستند. دلیل قائل شدن محدودیت مذکور آن است که در تعریفی که در مباحث قبلی از ماشین‌ها
تورینگ ارائه کردیم، هیچ ذکری از پذیرش رشته‌های تهی به میان نیامد. برای جلوگیری از تکر

مجدد تعریف یا افزودن یک شرط نقیض به آن، این فرض تلویحی را مطرح می‌کنیم که زبان‌های مورد بحث در این فصل، به جزء در مواردی که به نحوی گفته شود، فاقد λ می‌باشند. البته می‌توان وجود λ را هم در نظر گرفت و تمامی مطالب را با این فرض جدید، مجدداً ارائه کرد؛ اما به دلیل سهولت و غیر لازم نبودن، اینکار را به عهده خواننده واگذار می‌کنیم.

۱۱-۱ زبان‌های بازگشتی و شمارش‌پذیر بازگشتی

ابتدا با چند اصطلاح رایج در مورد زبان‌های مرتبط با ماشین‌های تورینگ آشنا می‌شویم. اما پیش از آن باید بین زبان‌هایی که یک ماشین تورینگ پذیرنده به ازای آن وجود دارد و زبان‌هایی که یک الگوریتم عضویت به ازای آن وجود دارد، تمایز قائل شویم. بدلیل آنکه ماشین‌های تورینگ لزوماً برای ورودی که آنرا نمی‌پذیرند توقیف نمی‌کنند، بنابراین وجود یک ماشین تورینگ پذیرنده بطور ضمنی به معنای وجود الگوریتم عضویت مربوطه نیست.

تعریف ۱۱-۱

زبان مفروض L **شمارش‌پذیر بازگشتی** خوانده می‌شود اگر ماشین تورینگی وجود داشته باشد که آنرا پذیرش کند.

معنای ضمنی تعریف فوق فقط این است که یک ماشین تورینگ M وجود دارد، به طوریکه به ازای هر $w \in L$ و با ضابطه:

$$q_0 w \mapsto_M x_1 q_f x_2.$$

که در آن q_f حالت پایانی می‌باشد. بنابراین، براساس این تعریف نمی‌توان گفت که برای w های غیرموجود در L چه اتفاقی می‌افتد؛ شاید در این صورت ماشین در یکی از حالات غیرپایانی متوقف شده یا اینکه هرگز متوقف نشود و وارد یکی از حلقه‌های نامتناهی گردد. با استفاده از تعریف بعدی، از ماشین می‌خواهیم که وجود یا عدم وجود یک ورودی مفروض را در زبان خود به ما اطلاع دهد.

تعریف ۱۱-۲

زبان مفروض L روی Σ **بازگشتی** خوانده می‌شود اگر ماشین تورینگ M وجود داشته باشد که L را پذیرفته و روی هر w موجود در Σ^* متوقف شود. به بیان دیگر، یک زبان بازگشتی خواهد بود اگر و تنها اگر یک الگوریتم عضویت به ازای آن وجود داشته باشد.

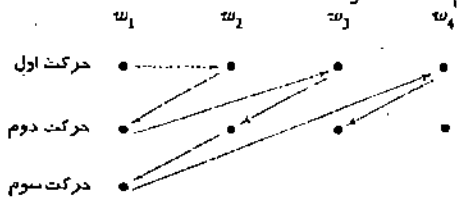
اگر یک زبان بازگشتی باشد، آنگاه به راحتی می‌توان یک روال شمارشی برای آن ایجاد کرد. فرض کنید که M ماشین تورینگی است که عضویت در زبان بازگشتی دلخواه L را تعیین می‌کند. ابتدا ماشین تورینگ دیگری به نام $\hat{M}$ می‌سازیم که تمام رشته‌های موجود در Σ^* را در ترتیب مناسبی از قبیل

$w_1, w_2, \ldots$ ایجاد می‌کند. این رشته‌ها پس از تولید، تبدیل به ورودی‌های M شده و به نحوی اصلاح می‌شوند تا فقط رشته‌هایی را روی نوار خود بنویسند که در L وجود دارند.

اما بررسی وجود یک روال شمارش به ازای هر زبان شمارش‌پذیربازگشتی به این آسانی نیست. نمی‌توان از استدلال بالا بطور کامل و بدون هیچ تغییری استفاده کرد. دلیل آن است که اگر یک w_i در L قرار نداشته باشد، ماشین M، در صورتی که در هنگام راه‌اندازی w_i روی نوار آن قرار داشته باشد، ممکن است هیچگاه متوقف نشده و بنابراین هرگز به نوارهایی از L که در شمارش پس از w_i قرار می‌گیرند، دسترسی پیدا نکند. برای جلوگیری از وقوع این مشکل، محاسبه را به روش دیگری دنبال می‌کنیم: ابتدا بوسیله $\hat{M}$، w_1 را بوجود آورده و اجازه می‌دهیم که M یک حرکت روی آن انجام دهد. سپس بوسیله $\hat{M}$، w_2 را ایجاد کرده و اجازه می‌دهیم که M یک حرکت روی w_2 و حرکت بعدی را روی w_1 انجام دهد. اکنون، w_3 را تولید کرده و یک مرحله حرکت روی w_3 انجام می‌دهیم، مرحله دوم را روی w_2 و مرحله سوم را روی w_1 انجام می‌دهیم و الی آخر. ترتیب مراحل انجام کار در شکل ۱۱-۱ نمایش داده شده است. براساس این شکل، M هرگز وارد حلقه نامتناهی نمی‌شود. بدلیل آنکه هر $w \in L$ بوسیله $\hat{M}$ ایجاد شده و طی چندین مرحله متناهی توسط M پذیرفته می‌شود، تمامی رشته‌های داخل L نهایتاً بوسیله M تولید خواهند شد.

به راحتی می‌توان مشاهده کرد که هر زبانی که یک روال شمارش به ازای آن وجود دارد، شمارش‌پذیر بازگشتی است. ما فقط رشته ورودی داده شده را با رشته‌های متوالی ایجاد شده بوسیله روال شمارش مقایسه می‌کنیم. اگر $w \in L$، نهایتاً به یک همخوانی می‌رسیم و می‌توان اقدام به توقف روال نمود.

با بررسی تعاریف ۱۱-۱ و ۱۱-۲ نمی‌توان اطلاعات چندانی در مورد ماهیت زبان‌های بازگشتی و شمارش‌پذیر بازگشتی بدست آورد. این تعاریف صرفاً عناوینی را به خانواده زبان‌های مرتبط با ماشین‌های تورینگ نسبت می‌دهند، اما هیچ بینشی راجع به ماهیت زبان‌های نماینده این خانواده‌ها ارائه نمی‌کنند. همچنین اطلاعات زیادی راجع به روابط بین زبان‌های مذکور یا ارتباط آنها با خانواده زبان‌هایی که تا به اینجا آشنا شدیم، ارائه نمی‌دهند. این پرسش مطرح می‌شود که " آیا زبان‌هایی هم وجود دارند که علیرغم شمارش‌پذیربازگشتی بودن، بازگشتی نباشند؟" و " آیا زبان‌هایی وجود دارند که به نحوی قابل توصیف بوده، اما شمارش‌پذیربازگشتی نباشند؟" هرچند می‌توان پاسخ‌هایی برای این پرسش‌ها مطرح کرد، نمی‌توان از آنها جهت تولید مثال‌های بسیار روشنی برای اثبات عینی پرسش‌های فوق ، خصوصاً پرسش دوم، استفاده کرد.



شکل ۱۱-۱

به روش‌های مختلف می‌توان اثبات کرد زبان‌هایی وجود دارند که شمارش‌پذیر بازگشتی نیستند. یکی از کوتاه‌ترین این روش‌ها که از نتایج بنیادین و دقیق ریاضی استفاده می‌کند، در زیر ارائه شده است.

اگر S یک مجموعه شمارش‌پذیر نامتناهی باشد. آنگاه مجموعه توانی آن، یعنی 2^S، شمارش‌پذیر نیست.
اثبات: فرض کنید $S=\{s_1,s_2,s_3,\ldots\}$. آنگاه هر عضو مفروض t در 2^S را می‌توان بوسیله توالی از ۰ها و ۱ها نمایش داد؛ به این منظور، ۱ در موقعیت i قرار می‌گیرد اگر و تنها اگر s_i در t وجود داشته باشد. بعنوان مثال، مجموعه $\{s_2,s_3,s_6\}$ با ...01100100 و $\{s_1,s_3,s_5,\ldots\}$ با ...10101 نمایش داده می‌شود. مشخص است که هر یک از اعضای 2^S را می‌توان بوسیله همچین توالی‌هایی نمایش داد و هر چنین توالی هم بیانگر فقط یکی از اعضای 2^S است. فرض کنید که 2^S شمارش‌پذیر باشد؛ آنگاه اعضای آنرا می‌توان به ترتیبی ، مثل $t_1,t_2,\ldots$، نوشته و در جدولی مشابه شکل ۱۱-۲ قرار داد. اعضای روی قطر اصلی این جدول را در نظر گرفته و هر یک از ورودی‌های آنرا مکمل‌گیری کنید؛ یعنی 0 را با 1 جایگزین کنید و برعکس. در مثال شکل ۱۱-۲، با در اختیار داشتن اعضای ...1100، نتیجه ...0011 بدست می‌آید. توالی جدید در امتداد قطر بیانگر یکی از اعضای 2^S، مثلاً t_i به ازای یک i مفروض، است. اما این عضو نمی‌تواند t_1 باشد، چون در امتداد s_1 با t_1 متفاوت است. به همین ترتیب می‌توان اثبات کرد که t_2,t_3 یا هیچکدام از دیگر ورودی‌های روال شمارش هم نمی‌باشد. این تناقض منطقی فقط با نادیده گرفتن فرض شمارش‌پذیربودن 2^S، قابل حل است.

t_1	1	0	0	0	0 ...
t_2	1	1	0	0	0 ...
t_3	1	1	0	1	0 ...
t_4	1	1	0	0	1 ...
⋮					

این نوع استدلال‌ها، به دلیل لزوم دستکاری اعضای روی قطر جدول ، قطری‌کردن نامیده می‌شود. ریاضی‌دان معروفی به نام G. F. Cantor ضمن ابداع این روش، از آن جهت اثبات شمارش‌ناپذیر بودن مجموعه اعداد حقیقی استفاده کرد. می‌توانید با مراجعه به منابع مختلف، استدلال‌های مشابهی را مطالعه کنید. در قضیه ۱۱-۱ از یکی از ساده‌ترین مدل‌های قطری شدن استفاده شده است.

بعنوان یکی از نتایج فوری از قضیه قبل می‌توان اثبات کرد که تعداد ماشین‌های تورینگ کمتر از تعداد زبان‌هاست، بنابراین برخی زبان‌ها شمارش‌پذیربازگشتی نیستند.

قضیه ۱۱-۲

به ازای هر Σغیرتهی، زبان‌هایی وجود دارند که شمارش پذیربازگشتی نیستند.
اثبات: هر زبان زیرمجموعه‌ای از Σ^* است و هر زیرمجموعه این‌چنین، یک زبان است. بنابراین، تعداد مجموعه تمام زبان‌ها دقیقاً 2^{Σ^*} است. به دلیل نامتناهی بودن Σ^* و بر اساس قضیه ۱۱-۱، مجموعه تمام زبان‌های روی Σ شمارش‌پذیر نیست. اما چون می‌توان مجموعه تمام ماشین‌های تورینگ را شمارش کرد، مجموعه تمام زبان‌های شمارش پذیر بازگشتی هم شمارش‌پذیر است. در تمرین ۱۶ انتهای این بخش، به صورت ضمنی اشاره می‌شود که برخی زبان‌ها روی Σ وجود دارند که شمارش‌پذیر بازگشتی نیستند. ■

این اثبات، علیرغم مختصر بودن و سادگی، از بسیاری جهات ناقص و نامناسب است. بعلاوه، چندان آموزنده نبوده و هرچند به طور ضمنی بیانگر وجود زبان‌هایی است که شمارش‌پذیربازگشتی نمی‌باشند، هیچ اطلاعاتی در مورد ماهیت آنها به ما ارائه نمی‌کند. طی مجموعه نتایج زیر، این نتیجه‌گیری را به صورت مفصل‌تر بررسی می‌کنیم.

زبانی که شمارش‌پذیربازگشتی نیست

هر زبانی که بوسیله یک روش الگوریتمی مستقیم قابل توصیف باشد، بوسیله یک ماشین تورینگ هم قابل پذیرش بوده و بنابراین شمارش‌پذیربازگشتی است. بنابراین برای توصیف زبانی که شمارش‌پذیر بازگشتی نباشد، باید از روش غیرمستقیم استفاده کرد. اما اینکار غیرممکن است. پیش از اثبات این ادعا باید مفهوم قطری‌کردن را کمی تغییر دهیم.

قضیه ۱۱-۳

یک زبان شمارش‌پذیر بازگشتی وجود داد که مکمل آن شمارش پذیر بازگشتی نیست.
اثبات: فرض کنید که $\Sigma=\{a\}$ و مجموعه تمام ماشین‌های تورینگ به ازای این الفبای ورودی را در نظر بگیرید. بر اساس قضیه ۱۰-۳، این مجموعه شمارش‌پذیر بوده و بنابراین می‌توان ترتیب مفروض $M_1,M_2,\ldots$ را به اعضای آن نسبت داد. به ازای هر ماشین تورینگ M_i، یک زبان شمارش‌پذیر بازگشتی مرتبط مثل $L(M_i)$ وجود دارد. بالعکس، به ازای هر زبان شمارش‌پذیر بازگشتی روی Σ، ماشین تورینگ مفروضی وجود دارد که آنرا می‌پذیرد.
حال زبان جدید L با تعریف زیر را در نظر می‌گیریم. به ازای هر $i\geq 1$، رشته a^i در L قرار دارد اگر و تنها اگر $a^i\in L(M_i)$. ناگفته پیداست که زبان L به خوبی تعریف شده است، از اینرو عبارت $a^i\in L(M_i)$ و $a^i\in L$ حتماً درست یا نادرست می‌باشد. سپس، مکمل L را در نظر می‌گیریم،

$$\overline{L}=\{a^i : a^i\notin L(M_i)\}, \qquad (1\text{-}11)$$

که آنهم گرچه به خوبی تعریف شده، اما همانطور که نشان خواهیم داد، شمارش‌پذیربازگشتی نمی‌باشد.

برای اثبات از تناقض استفاده کرده و ابتدا فرض می‌کنیم که $\overline{L}$ شمارش‌پذیر بازگشتی است. در اینصورت، باید ماشین تورینگ مفروض M_k وجود داشته باشد بطوریکه

$$\overline{L} = L(M_k). \qquad (۲-۱۱)$$

رشته a^k را در نظر بگیرید. این رشته در $\overline{L}$ قرار دارد یا در L ؟ فرض کنید که $a^k \in \overline{L}$. بر اساس (۱۱-۲)، رابطه اخیر به طور ضمنی به معنای

$$a^k \in L(M_k).$$

است. اما در اینصورت (۱۱-۱) به طور ضمنی اشاره می‌کند که

$$a^k \notin \overline{L}.$$

به همین ترتیب، اگر فرض کنیم که a^k در L قرار داشته باشد، آنگاه $a^k \notin \overline{L}$ و (۱۱-۲) ضمناً به این معناست که

$$a^k \notin L(M_k).$$

اما در اینصورت، از (۱۱-۱) داریم که

$$a^k \in \overline{L}.$$

مجدداً براساس تناقض نتیجه می‌گیریم که فرض شمارش‌پذیر بازگشتی $\overline{L}$، نادرست است.

در تکمیل اثبات این قضیه ، همانطور که گفته شد، باید نشان دهیم که L شمارش‌پذیر بازگشتی است. به همین دلیل، می‌توان از روال شمارش معروف در مورد ماشین‌های تورینگ استفاده کرد: ابتدا با معلوم بودن a^i، با شمارش تعداد a ها، i را پیدا می‌کنیم. سپس با استفاده از روال شمارش در ماشین‌های تورینگ ، M_i را پیدا می‌کنیم. نهایتاً، توصیف آنرا به همراه a^i به یک ماشین تورینگ جهانی به نام M_u ارائه می‌دهیم که عمل M روی a^i را شبیه‌سازی می‌کند. اگر a^i در L قرار داشته باشد، محاسبه انجام شده بوسیله M_u نهایتاً متوقف خواهد شد. این عوامل با هم باعث ایجاد ماشین تورینگی می‌شود که هر $a^i \in L$ را می‌پذیرد. بنابراین، براساس تعریف ۱۱-۱، L شمارش‌پذیر بازگشتی است. ■

اثبات این قضیه از طریق (۱۱-۱) به روشنی یک زبان به خوبی تعریف شده را ارائه می‌دهد که شمارش‌پذیربازگشتی نیست. این بدان معنی نیست که یک تفسیر آسان و شهودی از $\overline{L}$ وجود دارد، چون فقط می‌توان چند عضو انگشت‌شمار و پیش‌پا افتاده این زبان را نمایش داد. با این وجود، $\overline{L}$ به درستی تعریف شده است.

زبانی که شمارش‌پذیربازگشتی است ولی بازگشتی نیست

حال نشان می‌دهیم زبان‌هایی وجود دارند که شمارش‌پذیربازگشتی هستند، اما بازگشتی نیستند. برای اینکار هم باید از یک روش غیرمستقیم استفاده کرد. اما پیش از آن یک نتیجه فرعی را ارائه می‌کنیم.

قضیه ۱۱-۴

اگر زبان L و مکمل $\overline{L}$ آن هر دو شمارش‌پذیربازگشتی باشند، آنگاه هر دو زبان بازگشتی هستند. اگر L بازگشتی باشد، آنگاه $\overline{L}$ هم بازگشتی است و در نتیجه هر دو شمارش‌پذیربازگشتی هستند.

اثبات: اگر L و $\overline{L}$ هر دو شمارش‌پذیربازگشتی باشند، آنگاه ماشین‌های تورینگ M و $\hat{M}$ وجود دارند که به ترتیب، بعنوان روال‌های شمارش به ازای L و $\overline{L}$ عمل می‌کنند. اولین روال باعث تولید $w_1, w_2, \ldots$ در L و دومین روال باعث تولید $\hat{w}_1, \hat{w}_2, \ldots$ در $\overline{L}$ می‌شود. حال فرض کنید که یک $w \in \Sigma^*$ را در اختیار داشته باشیم. ابتدا w_1 را بوسیله M ساخته و آنرا با w مقایسه می‌کنیم. در صورت عدم تطابق، $\hat{w}_1$را بوسیله $\hat{M}$ ایجاد کرده و مجدداً مقایسه می‌کنیم. در صورت عدم تطابق و نیاز به ادامه کار، w_2 را از M و $\hat{w}_2$ را از $\hat{M}$ تولید کرده و الی آخر. از آنجایی که هر $w \in \Sigma^*$ بوسیله M یا $\hat{M}$ تولید می‌شود، نهایتاً به یک همخوانی می‌رسیم. اگر رشته همخوان بوسیله M تولید شده باشد، w به L تعلق داشته و در غیر اینصورت، در $\overline{L}$ قرار دارد. این فرآیند یک الگوریتم عضویت برای L و $\overline{L}$ ایجاد کرده و بنابراین، هر دو زبان مذکور بازگشتی هستند.

در حالت معکوس، فرض کنید که L بازگشتی باشد. آنگاه یک الگوریتم عضویت به ازای آن وجود دارد. اما می‌توان با مکمل‌گیری از نتیجه ، از آن بعنوان الگوریتم عضویت برای $\overline{L}$ استفاده کرد. بنابراین، $\overline{L}$ بازگشتی است. ابه دلیل آنکه هر زبان بازگشتی، شمارش‌پذیر بازگشتی است، اثبات در همین‌جا به پایان می‌رسد. ■

بر این اساس، مستقیماً می‌توان نتیجه گرفت که خانواده زبان‌های شمارش‌پذیر بازگشتی و خانواده زبان‌های بازگشتی با هم تفاوت دارند. زبان L در قضیه ۱۱-۳ جزء خانواده اول محسوب می‌شود، اما فاقد ویژگی زبان‌های خانواده دوم است.

قضیه ۱۱-۵

یک زبان شمارش‌پذیر بازگشتی وجود دارد که بازگشتی نیست. به بیان دیگر، خانواده زبان‌های بازگشتی یکی از زیرمجموعه‌های مناسب خانواده زبان‌های شمارش‌پذیر بازگشتی است.

اثبات: زبان L از قضیه ۱۱-۳ را در نظر بگیرید. این زبان شمارش‌پذیر بازگشتی است، اما مکمل آن چنین نیست. بنابراین، براساس قضیه ۱۱-۴ بازگشتی نیست. این زبان نمونه‌ای از زبان‌های موردنظراست. ■

براساس این قضیه اظهار می‌کنیم که برای برخی زبان‌های کاملاً به خوبی تعریف شده نمی‌توان یک الگوریتم عضویت ایجاد کرد.

تمرین‌ها

۱. اثبات کنید که مجموعه تمام اعداد حقیقی قابل شمارش نیست.

۲. اثبات کنید که مجموعه تمام زبان‌هایی که شمارش‌پذیر بازگشتی نیستند، قابل شمارش نمی‌باشد.

۳. فرض کنید که L یک زبان متناهی باشد. نشان دهید که آنگاه L^* شمارش‌پذیر بازگشتی است. یک روال شمارش به ازای L^* پیشنهاد کنید.

۴. فرض کنید L یک زبان مستقل‌ازمتن باشد. نشان دهید که آنگاه L^* شمارش‌پذیر بازگشتی است و یک روال شمارش برای آن پیشنهاد کنید.

۵. نشان دهید که اگر یک زبان شمارش‌پذیربازگشتی نباشد، مکمل آن هم بازگشتی نیست.

۶. نشان دهید که خانواده زبان‌های شمارش‌پذیربازگشتی تحت اجتماع بسته است.

۷. آیا خانواده زبان‌های شمارش‌پذیربازگشتی تحت اشتراک بسته است؟

۸. نشان دهید که خانواده زبان‌های بازگشتی تحت اجتماع و اشتراک بسته است.

۹. نشان دهید که خانواده زبان‌های بازگشتی و شمارش‌پذیربازگشتی تحت معکوس بسته هستند.

۱۰. آیا خانواده زبان‌های بازگشتی تحت الحاق بسته است؟

۱۱. اثبات کنید که مکمل یک زبان مستقل‌ازمتن حتماً بازگشتی است.

۱۲. فرض کنید L_1 بازگشتی و L_2 شمارش‌پذیربازگشتی باشد. نشان دهید که $L_2 - L_1$ لزوماً شمارش پذیر بازگشتی است.

۱۳. فرض کنید L چنان باشد که یک ماشین تورینگ وجود دارد که اعضای L را در ترتیب مناسب شمارش می‌کند. نشان دهید که در اینصورت، L بازگشتی است.

۱۴. اگر L بازگشتی باشد، لزوماً L^* هم بازگشتی است.

۱۵. یک رمزگذاری مخصوص برای ماشین‌های تورینگ انتخاب کرده و به کمک آن، یکی از اعضای زبان $\overline{L}$ در قضیه ۱۱-۳ را پیدا کنید.

۱۶. فرض کنید که S_1 یک مجموعه شمارش‌پذیر و S_2 یک مجموعه شمارش‌ناپذیر بوده و $S_1 \subset S_2$. نشان دهید که در اینصورت، S_2 حتماً حاوی تعداد نامتناهی از اعضایی است که در S_1 وجود ندارند.

۱۷. در تمرین ۱۶، نشان دهید که $S_2 - S_1$ عملاً شمارش‌پذیر نیست.

۱۸. به چه دلیل استدلال قضیه ۱۱-۲، با فرض متناهی بودن S، نادرست است.

۱۹. نشان دهید که مجموعه تمام اعداد غیرمنطقی شمارش‌پذیر نیست.

۱۱-۲ گرامرهای بدون محدودیت

برای درک ارتباط بین زبان‌ها و گرامرهای شمارش‌پذیربازگشتی، به تعریف کلی گرامر در فصل ۱ بازمی‌گردیم. در تعریف ۱-۱ ابتدا از فرم‌های مختلف قوانین تولید استفاده کردیم و سپس، با اعمال برخی محدودیت‌ها، انواع خاصی از گرامرها را ایجاد کردیم. گرامر اصلی، در صورتی که بدون هیچ محدودیتی در نظر گرفته شود، باعث ایجاد دسته‌ای از گرامرها به نام گرامرهای بدون محدودیت می‌شود.

تعریف ۱۱-۳

گرامر مفروض $G = (V, T, S, P)$ بدون محدودیت خوانده می‌شود اگر تمامی قوانین آن به فرم

$$u \rightarrow v,$$

باشند که در آن، u عضو $(V \cup T)^+$ و v عضو $(V \cup T)^*$ می‌باشد.

در گرامرهای بدون محدودیت، اساساً هیچ شرط و محدودیتی برای قواعد تولید قائل نمی‌شویم. بعلاوه، هر تعداد غیر پایانی و پایانی را می‌توان با هر ترتیبی در طرفین راست و چپ قرار داد. فقط یک شرط باید مورد توجه قرار گیرد: λ نمی‌تواند در سمت چپ قواعد تولید رخ دهد.

همانطور که بعداً خواهیم گفت، گرامرهای بدون محدودیت بسیار قدرتمندتر ازفرم‌های محدودشده‌ای از قبیل گرامرهای منظم و مستقل‌ازمتنی هستند که تا به حال مطالعه کرده‌ایم. در واقع، گرامرهای بدون محدودیت متناظر با بزرگ‌ترین خانواده زبان‌ها بوده و بنابراین قاعدتاً بوسیله ابزار مکانیکی قابل تشخیص می‌باشند؛ به این معنا که گرامرهای بدون محدودیت صرفاً خانواده زبان‌های شمارش‌پذیر بازگشتی را ایجاد می‌کنند. این ادعا را در دو بخش اثبات می‌کنیم که اولی به راحتی قابل انجام است، اما برای دومی نیازمند ایجاد مقدماتی از قبیل یک روش ساخت مفصل هستیم.

قضیه ۱۱-۶

هر زبانی که بوسیله یک گرامر بدون محدودیت ایجاد شود، شمارش‌پذیر بازگشتی است.

اثبات: این گرامر در واقع روالی برای شمارش قانونمند تمامی رشته‌های داخل یک زبان ارائه می‌کند. بعنوان مثال، می‌توان تمام w موجود در L را لیست کرد بطوریکه

$$S \Rightarrow w,$$

یعنی، w در یک مرحله مشتق شود. به دلیل متناهی بودن مجموعه قوانین این گرامر، تعداد رشته‌های این‌چنین، متناهی هم خواهد بود. سپس، تمام w های موجود در L را لیست می‌کنیم که در دو مرحله

$$S \Rightarrow x \Rightarrow w,$$

مشتق می‌شوند و الی آخر. می‌توان این مشتق‌ها را روی یک ماشین تورینگ شبیه‌سازی کرده و به این ترتیب، یک روال شمارش برای زبان مذکور ایجاد کرد. از این رو، زبان فوق شمارش‌پذیر بازگشتی است. ■

این شباهت بین زبان‌های شمارش‌پذیر بازگشتی و گرامرهای بدون محدودیت تا حدودی قابل پیش‌بینی بود. گرامر مذکور برای تولید رشته‌ها از یک فرآیند الگوریتمی به خوبی تعریف شده استفاده می‌کند، بنابراین می‌توان عمل مشتق‌گیری را هم روی یک ماشین تورینگ انجام داد. برای اثبات حالت عکس، نحوه تقلید یک ماشین تورینگ بوسیله یک گرامر بدون محدودیت را توصیف می‌کنیم.

ماشین تورینگ دلخواه $M=(Q,\Sigma,\Gamma,\delta,q_0,\square,F)$ را در اختیار داشته و می‌خواهیم گرامر G را ایجاد کنیم بطوریکه $L(G)=L(M)$. هرچند ایده زیربنایی ساخت ساده می‌باشد، پیاده‌سازی آن در با استفاده از نمادهای صوری بسیار سخت و طاقت‌فرساست.

بدلیل آنکه محاسبه ماشین تورینگ بوسیله توالی از پیکربندی‌های

$$q_0w \overset{*}{\mapsto} xq_fy, \tag{۳-۱۱}$$

قابل توصیف است، می‌توان آنرا به گونه‌ای تنظیم کرد که گزامر نظیر به صورتی باشد که

$$q_0w \overset{*}{\Rightarrow} xq_fy, \tag{۴-۱۱}$$

اگر و تنها اگر رابطه (۱۱-۳) برقرار باشد. این کار به راحتی قابل انجام است، اما بخش مشکل‌تر ایجاد ارتباط بین (۱۱-۴) و نتیجه مورد نظر ما، یعنی

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

به ازای تمامی w هایی است که در رابطه (۱۱-۳) صدق می‌کنند. به این منظور، گرامری می‌سازیم که در مجموع ویژگی‌های زیر را دارا باشد:

۱. S بتواند به ازای تمامی $w\in\Sigma^*$، q_0w را اشتقاق کند.

۲. (۱۱-۴) امکان‌پذیر است اگر و تنها اگر(۱۱-۳) برقرار باشد.

۳. وقتی رشته مفروض xq_fy با ضابطه $q_f\in F$ تولید می‌شود، رشته مذکور بوسیله گرامر به w اصلی تبدیل شود.

بنابراین، توالی کامل اشتقاق‌ها به صورت زیر است

$$S \overset{*}{\Rightarrow} q_0w \overset{*}{\Rightarrow} xq_fy \overset{*}{\Rightarrow} w. \tag{۵-۱۱}$$

مرحله سوم اشتقاق فوق مشکل‌تر از بقیه مراحل است. اگر گرامر در مرحله دوم اصلاح شده باشد، آنگاه چگونه می‌تواند w را به خاطر بیاورد؟ برای حل این مشکل، رشته‌ها را به صورتی رمزگذاری

می‌کنیم که نسخه رمزگذاری شده از همان ابتدا دو کپی از w در اختیار داشته باشد. اولین کپی ذخیره شده و دومین کپی در مراحل (۱۱-۴) مورد استفاده قرار می‌گیرد. با ورود یکی از پیکربندی‌های نهایی، گرامر هر چیزی غیر از w ذخیره شده را پاک می‌کند.

برای ایجاد دو کپی از w و کنترل سمبل حالت M (که نهایتاً بایستی بوسیله گرامر حذف شود)، متغیرهای $V_{\square a}$ و $V_{a\square}$ را به ازای تمامی $a\in\Sigma\cup\{\square\}, b\in\Gamma$، و تمامی i هایی را معرفی می‌کنیم که $q_i\in Q$. متغیر V_{ab} دو سمبل a و b را رمزگذاری می‌کند، در حالیکه V_{ab} مسئول رمزگذاری a و b و همچنین حالت q_iاست.

برای بدست آوردن فرم رمزگذاری شده اولین مرحله (۱۱-۵) می‌توان از

$$S \to V_{\square\square}S \mid SV_{\square\square} \mid T, \tag{۶-۱۱}$$

$$T \to TV_{aa} \mid V_{a0a}, \tag{۷-۱۱}$$

به ازای تمامی $a\in\Sigma$، استفاده کرد. گرامر قادر است تا بوسیله این قوانین تولید، نسخه رمزگذاری شده‌ای از هر یک از رشته‌های q_0w با هر تعداد دلخواه خالی در ابتدا و انتهای آنها بوجود آورد.

در مرحله دوم، به ازای هر انتقال

$$\delta(q_i,c)=(q_j,d,R)$$

در M ، قوانین گرامر

$$V_{aic}V_{pq} \to V_{ad}V_{pjq}, \tag{۸-۱۱}$$

را به ازای تمامی $a,p\in\Sigma\cup\{\square\}, q\in\Gamma$ قرار می‌دهیم. به ازای هر

$$\delta(q_i,c)=(q_j,d,L)$$

در M ، رابطه

$$V_{pq}V_{aic} \to V_{pjq}V_{ad}, \tag{۹-۱۱}$$

را به ازای تمامی $a,p\in\Sigma\cup\{\square\}$, $q\in\Gamma$ در G قرار می‌دهیم.

اگر در مرحله دوم، M وارد یکی از حالات پایانی شود، آنگاه گرامر باید هر چیزی غیر از w را، که در اولین شاخص‌های V ها ذخیره شده‌اند، از بین ببرد. بنابراین، به ازای هر $q_i\in F$، قوانین

$$V_{ajb} \to a, \tag{۱۰-۱۱}$$

را به ازای هر $b\in\Gamma$, $a\in\Sigma\cup\{\square\}$ اضافه می‌کنیم. به این ترتیب، اولین پایانی در رشته تولید شده و آن هم باعث بازنویسی مابقی رشته توسط

$$cV_{ab} \to ca, \tag{۱۱-۱۱}$$

$$V_{ab}c \to ac, \tag{۱۱-۱۲}$$

به ازای تمامی $a,c \in \Sigma \cup \{\square\}$ ، $b \in \Gamma$ می‌شود. اما با در اختیار داشتن تولید خاص

$$\square \to \lambda \tag{۱۱-۱۳}$$

می‌توان حالتی را کنترل کرد که M به خارج از بخشی از نوار حرکت می‌کند که در اشغال ورودی w قرار دارد. برای آنکه همه چیز بتواند در این حالت هم کار کند، بایستی ابتدا با استفاده از (۱۱-۶) و (۱۱-۷)

$$\square \ldots \square q_0 w \, \square \ldots \square,$$

را که نماینده تمامی ناحیه استفاده شده از نوار است، تولید کنیم. سپس، بوسیله (۱۱-۱۳) اقدام به حذف خالی‌های اضافی می‌کنیم.

مثال زیر نحوه ایجاد این ساختار پیچیده را نمایش می‌دهد. به دقت هر یک از مراحل مثال را بررسی کنید تا با وظیفه هر یک از قوانین یا ضرورت وجودی آنها آشنا شوید.

مثال ۱۱-۱

فرض کنید $M = (Q, \Sigma, \Gamma, \delta, q_0, \square, F)$ یک ماشین تورینگ با ضابطه

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

و

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

باشد. ماشین مذکور $L(aa^*)$ را می‌پذیرد.

اینک محاسبه

$$q_0 aa \mapsto aq_0 a \mapsto aaq_0 \square \mapsto aq_1 a \square, \tag{۱۱-۱۴}$$

را در نظر بگیرید که باعث پذیرش رشته aa می‌شود. برای اشتقاق این رشته از G ، ابتدا با استفاده از قوانین تعریف شده (۱۱-۶) و (۱۱-۷) ، رشته شروع مناسب را بدست می‌آوریم،

$$S \Rightarrow SV_{\square\square} \Rightarrow TV_{\square\square} \Rightarrow TV_{a\square}V_{\square\square} \Rightarrow V_{a0a}V_{a\square}V_{\square\square}.$$

فرم جمله‌ای اخیر نقطه آغاز خوبی برای بخشی از اشتقاق است که محاسبه ماشین تورینگ را دنبال می‌کند. این بخش شامل ورودی اصلی $aa\square$ در توالی اولین شاخص‌ها و پیکربندی شروع $q_0aa\square$ ، مابقی شاخص‌ها است. سپس،

$$V_{a0a}V_{aa} \to V_{aa}V_{a0a},$$

و

$$V_{a0a}V_{\square\square} \to V_{aa}V_{\square 0\square},$$

را اعمال می‌کنیم که نمونه‌های خاصی از (۱۱-۸) هستند، و همچنین

$$V_{aa}V_{\square 0\square} \to V_{a1a}V_{\square\square}$$

را که از (۱۱-۹) بدست می‌آیند. مراحل بعدی اشتقاق هم به صورت زیر است.

$$V_{a0a}V_{aa}V_{\square\square} \Rightarrow V_{aa}V_{a0a}V_{\square\square} \Rightarrow V_{aa}V_{aa}V_{\square 0\square} \Rightarrow V_{aa}V_{a1a}V_{\square\square}.$$

توالی اولین شاخص‌ها یکسان بوده و همواره ورودی شروع را به خاطر می‌آورد. توالی شاخص‌های دیگر به صورت

$$0aa\square, a0a\square, aa0\square, a1a\square,$$

است که متناظر با توالی پیکربندی‌های (۱۱-۱۴) می‌باشد.

در نهایت، (۱۱-۱۰) تا (۱۱-۱۳) در مراحل آخر مورد استفاده قرار می‌گیرند.

$$V_{aa}V_{a1a}V_{\square\square} \Rightarrow V_{aa}aV_{\square\square} \Rightarrow V_{aa}a\square \Rightarrow aa\square \Rightarrow aa.$$

از ساختار پیشنهادی که توسط (۱۱-۶) تا (۱۱-۱۳) توصیف شد، می‌توان بعنوان پایه اثبات نتیجه زیر استفاده کرد.

قضیه ۱۱-۷

برای هر زبان شمارش‌پذیربازگشتی L ، یک گرامر بدون‌محدودیت G وجود دارد، بطوریکه $L = L(G)$.

اثبات: ساختار توصیف شده تضمین می‌کند که

$$x \mapsto y,$$

پس،

$$e(x) \Rightarrow e(y),$$

که در آن، $e(x)$، دلالت بر رمزگذاری یک رشته براساس قرارداد مفروض دارد. سپس، با استقراء روی

تعداد مراحل می‌توان نشان داد که

$$e(q_0w)\overset{*}{\Rightarrow}e(y)$$

اگر و تنها اگر

$$q_0w\overset{*}{\mapsto}y.$$

در ادامه باید نشان داد که هر نوع پیکربندی شروع ممکن هم قابل تولید بوده و بعلاوه، w به درستی ساخته می‌شود. اگر و تنها اگر M وارد یکی از پیکربندی‌های پایانی شود. جزئیات این کار، که البته به راحتی هم قابل حدس است، بعنوان تمرین در انتهای همین بخش آمده است. ■

دو قضیه بالا نشان می‌دهند که خانواده زبان‌های مرتبط با گرامرهای بدون محدودیت هم ارز با خانواده زبان‌های شمارش‌پذیر بازگشتی است.

تمرین‌ها

۱. گرامر بدون محدودیت

$$S \to S_1B,$$
$$S_1 \to aS_1b,$$
$$bB \to bbbB,$$
$$aS_1b \to aa,$$
$$B \to \lambda$$

چه زبانی را تولید می‌کند.

۲. وقوع رشته تهی در سمت چپ یکی از قوانین یک گرامر بدون محدودیت چه مشکلاتی بوجود می‌آورد؟

۳. گونه‌ای از گرامرها را در نظر بگیرید که در آن، نقطه شروع هر یک از اشتقاق‌ها، مجموعه متناهی از چندین رشته است و نه فقط یک رشته. این مفهوم را مدون کرده و سپس ارتباط این دست گرامرها را با گرامرهای بدون محدودیتی که در فصل حاضر مطالعه کردیم، اثبات کنید.

۴. در مثال ۱۱-۱، اثبات کنید که گرامر ایجاد شده قادر به تولید رشته‌های دارای یک b نمی‌باشد.

۵. جزئیات اثبات قضیه ۱۱-۷ را ارائه دهید.

۶. یک ماشین تورینگ برای $L(01(01)^*)$ بسازید. سپس با استفاده از ساخت قضیه ۱۱-۷، یک گرامر بدون محدودیت به ازای آن ایجاد کنید. با استفاده از گرامر تولید، یک اشتقاق برای 0101 ارائه دهید.

۷. نشان دهید که به ازای هر گرامر بدون محدودیت، یک گرامر بدون محدودیت متناظر وجود دارد که تمامی قوانین آن به فرم

$$u \to v,$$

می‌باشند، بطوریکه $u,v \in (V \cup T)^*$ و $|u| \le |v|$، یا

$$A \to \lambda,$$

که در آن $A \in V$.

۸. نشان دهید که نتیجه تمرین ۷، حتی با افزودن شرایط جدید $|u| \le 2$ و $|v| \le 2$، باز هم به قوت خود باقی است.

۹. برخی نویسندگان تعریفی از گرامرهای بدون محدودیت ارائه می‌کنند که تا حدودی با تعریف ۱۱-۳ ما، تفاوت دارد. در این تعریف، قوانین یک گرامر بدون محدودیت باید به فرم

$$x \to y,$$

باشند که در آن،

$$x \in (V \cup T)^*V(V \cup T)^*,$$

و

$$y \in (V \cup T)^*.$$

وجه تمایز این تعریف با تعریف ما در اینجاست که در سمت چپ باید حداقل یک غیر پایانی وجود داشته باشد.

تعریف جدیدی را نشان دهید که به ازای هر گرامر از یکی از این دو نوع ، یک گرامر متناظر از نوع دیگر وجود دارد.

۱۱-۳

می‌توان در حد فاصل بین گرامرهای محدود مستقل‌ازمتن و گرامرهای بدون محدودیت عمومی، طیف عظیمی از گرامرهای "تقریباً محدود" را جای داد. تمامی گرامرهای واقع در این محدوده نکته جالب‌توجهی برای بررسی ندارند و از اینرو، فقط به گرامرهای حساس‌به‌متن می‌پردازیم. این گرامرها زبان‌های مرتبط با دسته محدودی از ماشین‌های تورینگ، یعنی اتوماتای کراندار خطی، را تولید می‌کنند که در بخش ۱۰-۵ توضیح داده شد.

تعریف ۱۱-۴

گرامر مفروض $G = (V, T, S, P)$ حساس‌به‌متن خوانده می‌شود اگر تمامی قوانین آن به فرم

$$x \to y,$$

باشند که در آن، $x, y \in (V \cup T)^+$ و

$$|x| \le |y|. \qquad (۱۱-۱۵)$$

تعریف فوق یکی از جنبه‌های این نوع گرامرها، به نام ویژگی **غیرانقباضی** را مطرح می‌کند. بر این اساس، طول فرم‌های جمله‌ای متوالی هرگز کاهش نمی‌یابد. شاید نتوان دلیل دقیقی برای حساس به متن بودن این دست گرامرها مطرح کرد، اما اثبات شده که (بعنوان مثال، مراجعه شود به Salomma 1973) تمامی این گرامرها را می‌توان به فرم نرمال بازنویسی کرد نوشت که در آن، تمامی قوانین به فرم

$$xAy \to xvy$$

باشند. به عبارت دیگر، می‌توان گفت که قاعدهٔ تولید

$$A \to v$$

فقط در مواردی قابل استفاده است که A در متنی رخ دهد که رشته x در سمت چپ و رشته y در سمت راست آن قرار داشته باشد. هرچند که ما از اصطلاح ناشی از این تفسیر خاص استفاده می‌کنیم، شکل ظاهری آن فعلاً مورد توجه ما نبوده و فقط از تعریف ۱۱-۴ استفاده می‌کنیم.

زبان‌های حساس‌به‌متن و اتوماتای کراندار خطی

ناگفته پیداست که گرامرهای حساس‌به‌متن با خانواده زبان‌های همنام خود مرتبط هستند.

تعریف ۱۱-۵

زبان مفروض L **حساس به متن** خوانده می‌شود اگر گرامر حساس‌به‌متن G وجود داشته باشد، بطوریکه $L = L(G)$ یا $L = L(G) \cup \{\lambda\}$.

رشته تهی را مجدداً در این تعریف می‌گنجانیم. تعریف ۱۱-۴ ضمناً به معنای غیرمجاز بودن $x \to \lambda$ می‌باشد و بنابراین گرامرهای حساس‌به‌متن هرگز قادر به تولید زبانهای دارای رشته تهی نمی‌باشند. با این وجود، هر زبان مستقل‌ازمتن فاقد λ را می‌توان با استفاده از یکی از حالات خاصی گرامرهای حساس‌به‌متن ایجاد کرد. برای این منظور، مثلاً می‌توان از فرم‌های نرمال گریباخ یا چامسکی استفاده کرد که هر دو در تعریف زبان‌های (نه گرامرهای) حساس‌به‌متن برقرار است. از اینرو، می‌توان ادعا کرد که خانواده زبان‌های مستقل‌ازمتن زیرمجموعه‌ای از خانواده زبان‌های حساس‌به‌متن است.

مثال ۱۱-۳

زبان $L = \{a^n b^n c^n : n \geq 1\}$ یک زبان حساس‌به‌متن است. برای اثبات، از یک گرامر حساس‌به‌متن در زبان مذکور مانند گرامر

$$S \to abc \mid aAbc,$$
$$Ab \to bA,$$
$$Ac \to Bbcc,$$
$$bB \to Bb,$$
$$aB \to aa \mid aaA$$

استفاده می‌کنیم. برای تحقیق در مورد عملکرد این روش، یکی از مشتق‌های $a^3b^3c^3$ را بررسی می‌کنیم.

$$\begin{aligned} S &\Rightarrow aAbc \Rightarrow abAc \Rightarrow abBbcc \\ &\Rightarrow aBbbcc \Rightarrow aaAbbcc \Rightarrow aabAbcc \\ &\Rightarrow aabbAcc \Rightarrow aabbBbccc \\ &\Rightarrow aabBbbccc \Rightarrow aaBbbbccc \\ &\Rightarrow aaabbbccc. \end{aligned}$$

در این راه‌حل از متغیرهای A و B عملاً به عنوان پیغام‌رسان استفاده شده است. یک A در سمت چپ تولید شده و سپس با حرکت به سمت راست به طرف اولین c می‌رود و در آنجا یک b و c دیگر ایجاد می‌کند. سپس A پیغام‌رسان B را به سمت چپ بازگردانده و a نظیر را ایجاد می‌کند. این فرآیند تا حدود زیادی شبیه روش برنامه‌ریزی ماشین‌های تورینگ برای پذیرش زبان مفروض L است. ■

به دلیل مستقل‌ازمتن نبودن زبان مثال فوق، می‌توان نتیجه گرفت که خانواده زبان‌های مستقل‌ازمتن یکی از زیرمجموعه‌های خانواده زبان‌های حساس‌به‌متن است. مثال ۱۱-۲ هم نشان می‌دهد که پیداکردن یک گرامر حساس‌به‌متن، حتی در مثال‌های نسبتاً ساده هم، کار آسانی نیست. در اغلب موارد، ساده‌ترین راه ایجاد برنامه برای یک ماشین تورینگ و سپس، یافتن گرامر متناظر با آن است. با بررسی چند مثال می‌توان دریافت که در صورت حساس‌به‌متن بودن زبان، ماشین تورینگ متناظر با آن نیازها و شرایط فضایی قابل پیش‌بینی دارد. بنابراین، در برخی موارد می‌توان آنرا به عنوان یک اتومات کراندار خطی در نظر گرفت.

قضیه ۱۱-۸

به ازای هر زبان L حساس‌به‌متن دارای λ، یک اتومات کراندار خطی M وجود دارد بطوریکه $L = L(M)$.

اثبات: اگر L حساس‌به‌متن باشد، آنگاه یک گرامر حساس‌به‌متن به ازای $L - \{\lambda\}$ وجود خواهد داشت. نشان می‌دهیم که اشتقاق‌های این گرامر را می‌توان بوسیله یک اتومات کراندار خطی شبیه‌سازی کرد. این اتومات کراندار خطی دو شیار دارد که در یکی، رشته ورودی w و در دیگری، فرم‌های جمله‌ای اشتقاق‌شده با استفاده از G قرار می‌گیرد. یک نکته بسیار مهم در استدلال فوق آن است که طول هیچکدام از فرم های جمله‌ای ممکن از $|w|$ بیشتر نخواهد بود. نکته قابل توجه دیگر اینکه اتوماتای کراندار خطی، براساس تعریف، نامعین هستند. توجه به این امر در استدلال‌ها ضروری است؛ چون در غیر اینصورت، می‌توان ادعا کرد که تولید صورت گرفته همواره قابل پیش‌بینی بوده و بنابراین باید از دنبال کردن هر مسیر انتخابی که تولیدی ندارد، اجتناب کرد. بنابراین، محاسبه مورد بحث در قضیه ۱۱-۶ را می‌توان بدون نیاز به فضا انجام داد، مگر اینکه در ابتدا بوسیله w اشغال شده باشد؛ بنابراین، محاسبه فوق بوسیله یک اتومات کراندار خطی قابل انجام است. ■

قضیه ۱۱-۹

اگر زبان L بوسیله یک اتومات کراندار خطی مفروض به نام M پذیرفته شود، آنگاه یک گرامر حساس‌به‌متن وجود دارد که L را تولید می‌کند.

اثبات: روش ساخت مشابه روش ساخت قضیه ۱۱-۷ است. تمامی قوانین تولیدی در قضیه ۱۱-۷، به استثنای (۱۱-۱۳)، غیر انقباضی است،

$$\square \rightarrow \lambda.$$

اما این قاعدهٔ‌تولید را می‌توان حذف کرد. چون قانون تولید فوق فقط در صورتی لازم است که ماشین تورینگ به خارج از حدود ورودی اصلی برسد، که البته در اینجا چنین نیست. گرامر حاصل از ساخت، بدون در نظر گرفتن این حاصل غیرضروری، غیرانقباضی است و به این ترتیب استدلال به پایان می‌رسد. ■

ارتباط بین زبان‌های حساس‌به‌متن و بازگشتی

بر اساس قضیه ۱۱-۹ می‌توان گفت که هر زبان حساس‌به‌متنی بوسیله یک ماشین تورینگ پذیرفته شده و بنابراین، شمارش‌پذیر بازگشتی محسوب می‌شود. در قضیه ۱۱-۱۰ از این اصل استفاده شده است.

قضیه ۱۱-۱۰

تمامی زبان‌های حساس‌به‌متن L، بازگشتی هستند.

اثبات: زبان حساس‌به‌متن L با گرامر حساس‌به‌متن مربوطه G را در نظر بگیرید. به اشتقاق زیر از w توجه کنید.

$$S \Rightarrow x_1 \Rightarrow x_2 \Rightarrow \cdots \Rightarrow x_n \Rightarrow w.$$

می‌توان با حفظ کلیت، تمام فرم‌های جمله‌ای یک اشتقاق را غیرمشابه در نظر گرفت؛ به این معنا که به ازای تمامی $i \neq j$، $x_i \neq x_j$. اما مشکل در اینجا می‌باشد که تعداد مراحل هر اشتقاق تابعی کراندار از $|w|$ است. می‌دانیم که

$$|x_j| \leq |x_{j+1}|,$$

چون G غیرانقباضی است. فقط لازم به توضیح است که یک m مفروض وجود دارد که فقط کراندار به G و w بوده و به ازای تمامی jهای

$$|x_j| < |x_{j+m}|,$$

به شرطی که $m = m(|w|)$ یک تابع کراندار از $|V_T|$ و $|w|$ باشد. این بدان علت است که متناهی بودن $|V_T|$ ضمناً به معنای وجود تعداد بسیار متناهی رشته با طول مفروض است. بنابراین، طول یکی از اشتقاق‌های $w \in L$ حداکثر $|w|m(|w|)$ می‌باشد.

به این ترتیب، یک الگوریتم عضویت برای L بدست می‌آید. از اینرو، تمامی اشتقاق‌ها تا طول حداکثر $|w|m(|w|)$ را بررسی می‌کنیم. به دلیل متناهی بودن مجموعه قوانین G، فقط تعداد متناهی از این اشتقاق‌ها وجود دارد. اگر یکی از آنها w را تولید کند، آنگاه $w \in L$، و در غیر اینصورت چنین نیست. ■

قضیه ۱۱-۱۱

یک زبان بازگشتی وجود دارد که حساس‌به‌متن نمی‌باشد.

اثبات: مجموعه تمام گرامرهای حساس‌به‌متن روی $T = \{a,b\}$ را در نظر بگیرید. از قراردادی استفاده می‌کنیم که در آن، هر گرامر دارای مجموعه غیر پایانی‌ها به فرم

$$V = \{V_0, V_1, V_2, \ldots\}$$

باشد. تمامی گرامرهای حساس‌به‌متن فقط بوسیله قوانین خود مشخص می‌شوند. بنابراین، می‌توان آنها را به صورت یک تک رشته مثل

$$x_1 \rightarrow y_1;\ x_2 \rightarrow y_2; \ldots;\ x_m \rightarrow y_m$$

در نظر گرفت. حال در این رشته، هم‌ریختی

$$\begin{aligned} h(a) &= 010, \\ h(b) &= 01^20, \\ h(\rightarrow) &= 01^30, \\ h(;) &= 01^40, \\ h(V_i) &= 01^{i+5}0 \end{aligned}$$

را استفاده می‌کنیم. بنابراین، هرگرامر حساس‌به‌متنی را می‌توان صرفاً بوسیله رشته‌ای از $L((011^*0)^*)$ نمایش داد. بعلاوه، چون با در اختیار داشتن یکی از این رشته‌ها، حداکثر یک گرامر حساس‌به‌متن متناظر با آن وجود دارد، این نمایش را می‌توان معکوس نمود.

اینک یک ترتیب محض روی $\{0,1\}^+$ معرفی می‌کنیم، می‌توان رشته‌هایی را با ترتیب $w_1, w_2, \ldots$ نوشت. رشته مفروض w_i ممکن است نتواند یک گرامر حساس‌به‌متن را تعریف کند: اگر تعریف کند، آن را گرامر G_i می‌نامیم. سپس، بوسیله

$$L = \{w_i : \text{گرامر حساس‌به‌متن } G_i \text{ را تعریف می‌کند و } w_i \notin L(G_i)\}$$

زبانی به نام L را تعریف می‌کنیم. L به خوبی تعریف شده و در واقع بازگشتی است. برای بررسی صحت بازگشتی بودن، یک گرامر عضویت را ایجاد می‌کنیم. با معلوم بودن w_i، بررسی می‌کنیم آیا w_i گرامر حساس به متن G_i را تعریف می‌کند یا خیر. در صورت منفی بودن پاسخ، آنگاه $w_i \notin L$.

اگر این رشته یک گرامر را تعریف کند، آنگاه $L(G_i)$ بازگشتی است و می‌توان با استفاده از الگوریتم عضویت قضیه ۱۱-۱۰ تعیین کرد که آیا $w_i \in L(G_i)$ یا خیر. در صورت منفی بودن پاسخ، آنگاه w_i به L تعلق دارد.

اما L حساس‌به‌متن نیست. در صورت حساس‌به‌متن بودن، باید یک w_j وجود داشته باشد بطوریکه $L = L(G_j)$. سپس تحقیق می‌کنیم که آیا w_j در $L(G_j)$ قرار دارد. اگر فرض کنیم که $w_j \in L(G_j)$، آنگاه براساس تعریف w_j در L قرار ندارد. اما چون $L = L(G_j)$، با تضاد مواجه می‌شویم. بالعکس، اگر فرض کنیم که $w_j \notin L(G_j)$، آنگاه براساس تعریف $w_j \in L$ و مجدداً به تضاد می‌رسیم. بنابراین نتیجه می‌گیریم که L حساس‌به‌متن نیست. ■

بر اساس نتیجه قضیه ۱۱-۱۱، اتوماتای کراندار خطی عملاً ضعیف‌تر از ماشین‌های تورینگ بوده و از اینرو، فقط قادر به پذیرش یکی از زیرمجموعه‌های مناسب زبان‌های بازگشتی می‌باشند. این نتیجه در واقع ناشی از قدرتمندتر بودن اتوماتای کراندار خطی در مقایسه با اتوماتای بالابه‌پایین است. زبان‌های مستقل‌ازمتن، در صورتیکه بوسیله گرامرهای مستقل‌ازمتن تولید شوند، یکی از زیرمجموعه‌های زبان‌های حساس‌به‌متن می‌باشند. همانطور که از بسیاری از مثال‌ها می‌توان حدس زد، این زبان‌ها زیرمجموعه مناسبی محسوب می‌شوند. مشاهده می‌کنیم که به دلیل هم ارزی اساسی بین اتوماتای کراندار خطی و زبان‌های حساس‌به‌متن از یک سو و اتوماتای پشته‌ای و زبان‌های مستقل‌ازمتن از سوی دیگر، هر زبان پذیرفته شده بوسیله یک اتومات پشته‌ای، بوسیله یک اتومات کراندار خطی هم پذیرفته می‌شود. اما زبان‌هایی هم وجود دارند که بوسیله اتوماتای کراندار خطی پذیرفته می‌شوند، اما هیچ اتوماتای پشته‌ای به ازای آن وجود ندارد.

تمرین‌ها

۱. * - گرامرهای حساس‌به‌متنی را برای زبان‌های زیر پیدا کنید.

الف) $L = \{a^{n+1}b^n c^{n-1} : n \geq 1\}$.

ب) $L = \{a^n b^n a^{2n} : n \geq 1\}$.

ج) $L = \{a^n b^m c^n d^m : n \geq 1, m \geq 1\}$. ◔

د) $L = \{ww : w \in \{a,b\}^+\}$.

ه) $L = \{a^n b^n c^n d^m : n \geq 1\}$.

۲. * - گرامرهای حساس‌به‌متنی را برای زبان‌های زیر پیدا کنید.

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

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

۳. نشان دهید که خانواده زبان‌های حساس‌به‌متن تحت اجتماع بسته است.

۴. نشان دهید که خانواده زبان‌های حساس‌به‌متن تحت معکوس بسته است. ◔

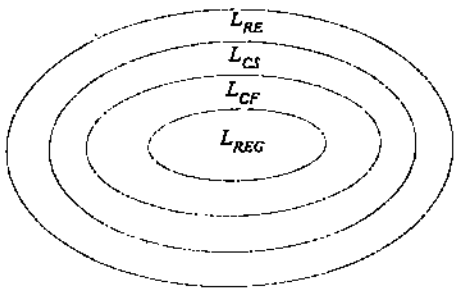
۵. به ازای m در قضیه ۱۱-۱۰، حدهای مشخصی را برای m بعنوان تابعی از $|w|$ و $|VUT|$ ارائه دهید.

۶. بدون ساخت آشکار زبان $L = \{wuw^R : w, u \in \{a,b\}^+, |w| \geq |u|\}$، نشان دهید که یک گرامر حساس‌به‌متن برای زبان فوق وجود دارد. ◔

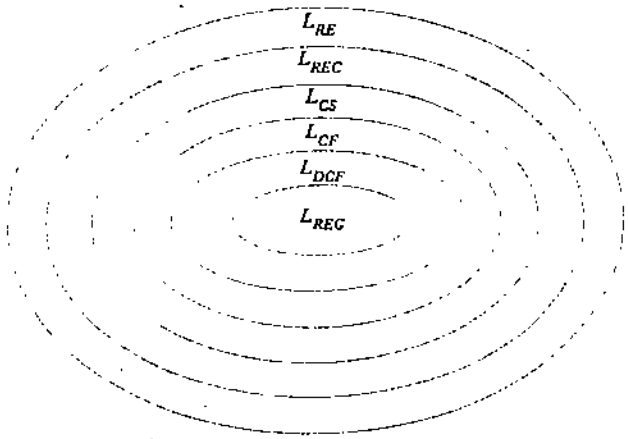
۱۱-۴ سلسله مراتب چامسکی

تا به اینجا با تعدادی از خانواده زبان‌ها از قبیل زبان‌های شمارش‌پذیربازگشتی (L_{RE})، زبان‌های حساس‌به‌متن (L_{CS})، زبان‌های مستقل‌ازمتن (L_{CF}) و زبان‌های منظم (L_{REG}) آشنا شدیم. یک روش برای نمایش رابطه بین این خانواده‌ها استفاده از **سلسله‌مراتب چامسکی** است. نوآم چامسکی که یکی از پایه‌گذاران نظریه زبان‌های صوری محسوب می‌شود، اولین کسی بود که زبان‌ها را در چهار گروه، از نوع صفر تا نوع سه، دسته‌بندی کرد. هر چند اصطلاحی که او در ابتدا برای نامگذاری بکار برد، هنوز هم باقی مانده و مکرراً مورد استفاده قرار می‌گیرد، انواع عددی عملاً اسامی دیگری برای خانواده زبان‌هایی است که تا به اینجا مورد مطالعه قرار دادیم. زبان‌های نوع 0 زبان‌هایی هستند که بوسیله گرامرهای محدود نشده، یعنی زبان‌های شمارش‌پذیربازگشتی، ایجاد می‌شوند. نوع 1 شامل زبان‌های حساس‌به‌متن، نوع 2 شامل زبان‌های مستقل‌ازمتن و نوع 3 شامل زبان‌های منظم می‌باشد. همانطور که قبلاً هم مشاهده کردیم، هر یک از خانواده زبان‌های نوع i یکی از زیرمجموعه‌های مناسب خانواده نوع $i-1$ محسوب می‌شوند. نمودار شکل ۱۱-۳ این رابطه را به روشنی نمایش می‌دهد. در شکل ۱۱-۳ سلسله‌مراتب اولیه چامسکی را مشاهده می‌کنید. می‌توان خانواده زبان‌های دیگری را که بعداً یافت شدند، به این شکل اضافه کرد. با در نظر گرفتن خانواده زبان‌های مستقل‌ازمتن معین (L_{DCF}) و زبان‌های بازگشتی (L_{REC})، سلسله مراتب فوق گسترش یافته و به شکل ۱۱-۴ تبدیل می‌شود.

هرچند می‌توان خانواده زبان‌های دیگری را هم تعریف کرد و در شکل ۱۱-۴ قرار داد، با این وجود روابط آنها همواره مشابه ساختار جاسازی شده شکل ۱۱-۳ و ۱۱-۴ مشخص نیست. در برخی موارد، روابط به طور کامل شناخته نشده و هنوز در هاله‌ای از ابهام قرار دارد.



شکل ۱۱-۳



شکل ۱۱-٤

مثال ۱۱-۳

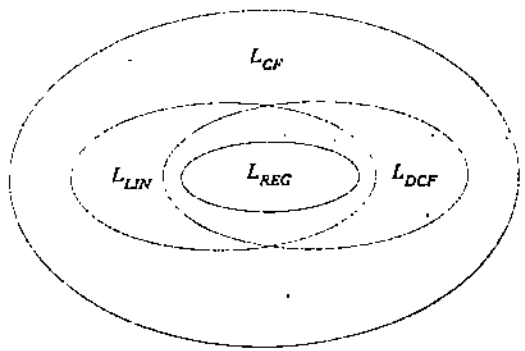
قبلاً زبان مستقل‌ازمتن

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

را معرفی کردیم. همچنین اثبات کردیم این زبان معین و غیرخطی است. از طرف دیگر، زبان

$$L = \{a^n b^n\} \bigcup \{a^n b^{2n}\}$$

خطی و نامعین است. به این معنا که رابطه بین زبان‌های منظم، خطی، مستقل‌ازمتن معین و مستقل‌ازمتن نامعین مطابق شکل ۱۱-۵ است.



شکل ۱۱-۵

■

یک مسأله هنوز بدون حل باقی مانده است. در تمرین ۸ بخش ۱۰-۵ با مفهوم اتومات کراندار خطی معین آشنا شدیم. حال پرسشی را مطرح می‌کنیم که قبلاً در مورد دیگر اتوماتا مطرح کرده بودیم: نامعین بودن چه نقشی در این میان برعهده دارد؟ متأسفانه، نمی‌توان به راحتی به این پرسش پاسخ داد. در حال حاضر نمی‌دانیم که آیا خانواده زبان‌های مورد پذیرش بوسیله اتوماتای کراندار خطی معین یکی از زیرمجموعه‌های مناسب زبان‌های حساس‌به‌متن محسوب می‌شود یا خیر.

به طور خلاصه، تا به اینجا روابط بین خانواده زبان‌های مختلف و اتوماتای مربوط به آنها را بررسی کردیم. به این منظور، سلسله مراتبی از زبان‌ها ایجاد کرده و اتوماتا را بر حسب قدرت بعنوان پذیرنده‌های زبان دسته‌بندی کردیم. ماشین‌های تورینگ قدرتمندتر از اتوماتای کراندار خطی هستند. این اتوماتا هم قدرتمندتر از اتوماتای پشته‌ای هستند. در انتهای سلسله مراتب فوق، پذیرنده‌های معین قرار دارند که پیش از همه مورد بررسی قرار گرفتند.

تمرین‌ها

۱. مثال‌هایی را در کتاب جمع‌آوری کنید که در آنها اثبات شده که روابط تمامی زیرمجموعه‌های نمایش داده شده در شکل ۱۱-۴ عملاً روابط مناسبی هستند.

۲. دو مثال (به استثنای مثال ۱۱-۳) از زبان‌هایی پیدا کنید که علیرغم خطی بودن، مستقل‌ازمتن معین نمی‌باشند.

۳. دو مثال (به استثنای مثال ۱۱-۳) از زبان‌هایی پیدا کنید که علیرغم مستقل‌ازمتن معین بودن، غیرخطی نمی‌باشند.

پاسخ‌های تشریحی و نکات کلیدی تمرینات منتخب

فصل ۱

بخش ۱-۱

۵- برای اثبات تساوی دو مجموعه، باید نشان دهیم که یک عضو در مجموعه اول وجود دارد اگر و تنها اگر در مجموعه دوم هم وجود داشته باشد. فرض کنید $x \in \overline{S_1 \cup S_2}$. آنگاه $x \notin S_1 \cup S_2$، یعنی اینکه x نمی‌تواند در S_1 یا در S_2 باشد؛ بنابراین $x \in \overline{S}_1 \cap \overline{S}_2$. بالعکس، اگر $x \in \overline{S}_1 \cap \overline{S}_2$، آنگاه x در S_1 و S_2 وجود ندارد؛ یعنی اینکه $x \in \overline{S_1 \cup S_2}$.

۶- برای اثبات می‌توان از استقراء روی تعداد مجموعه‌ها استفاده کرد. فرض کنید $Z = S_1 \cup S_2 \ldots \cup S_n$. آنگاه $S_1 \cup S_2 \ldots \cup S_n \cup S_{n+1} = Z \cup S_{n+1}$ است. براساس قانون استاندارد دمورگان،

$$\overline{Z \cup S_{n+1}} = \overline{Z} \cap \overline{S}_{n+1}.$$

براساس فرض استقراء، این رابطه برای n مجموعه صدق می‌کند؛ یعنی،

$$\overline{Z} = \overline{S}_1 \cap \overline{S}_2 \cap \ldots \overline{\cap S}_n.$$

بنابراین،

$$\overline{Z \cup S_{n+1}} = \overline{S}_1 \cap \overline{S}_2 \cap \ldots \overline{\cap S}_n \cap \overline{S}_{n+1}.$$

و به این ترتیب، گام استقراء به پایان می‌رسد.

۸- فرض کنید $S_1 = S_2$ باشد. آنگاه $S_1 \cap \overline{S_2} = \overline{S_1} \cap S_2 = S_1 \cap \overline{S_1} = \emptyset$ و کل عبارت مجموعه تهی می‌باشد. حال فرض کنید که $S_1 \neq S_2$ و یکی از اعضای x در S_1 باشد ولی در S_2 نباشد. آنگاه $x \in \overline{S_2}$، بطوریکه $S_1 \cap \overline{S_2} \neq \emptyset$ است. بنابراین، عبارت نمی‌تواند مساوی با مجموعه تهی باشد.

۱۳- اگر x در S_1 باشد و x در S_2 باشد، آنگاه x در $(S_1 \cup S_2) - S_2$ وجود ندارد. به همین دلیل، یک شرط لازم و کافی این است که دو مجموعه جدا از هم باشند.

۱۷- (ج) چون،

$$\frac{n!}{n^n} = \frac{n}{n}\frac{n-1}{n}\cdots\frac{2}{n}\frac{1}{n}$$

حاصل ضرب عوامل، کوچکتر یا مساوی یک است. بنابراین، $n! = O(n^n)$ می‌باشد.

۳۰- در این مسأله باید از برهان خلف استفاده کنیم. فرض کنید که $2-\sqrt{2}$ گویا باشد. آنگاه از

$$2-\sqrt{2} = \frac{n}{m}$$

داریم

$$\sqrt{2} = \frac{2m-n}{m},$$

که با فرض گویا بودن $\sqrt{2}$ در تضاد است.

۳۳- از استقراء کمک می‌گیریم. فرض کنید که هر عدد صحیح کوچکتر از n را می‌توان به صورت حاصل‌ضرب اعداد اول نوشت. اگر n اول باشد، دیگر نیازی به اثبات وجود ندارد؛ در غیر اینصورت، می‌توان آنرا به صورت حاصل‌ضرب

$$n = n_1 n_2,$$

نوشت. دقت کنید که در رابطه بالا، هر دو عامل اول کوچکتر از n هستند. براساس فرض استقراء، هر دو عدد فوق و از جمله n را می‌توان به صورت حاصل‌ضرب عوامل اول نوشت.

بخش ۱-۲

۲- بسیاری از موجودیت‌های رشته را می‌توان بوسیله استقراء اثبات کرد. فرض کنید که به ازای تمامی $u \in \Sigma^*$ و تمامی v های با طول n، رابطه $(uv)^R = v^R u^R$ برقرار باشد. حال رشته‌ای مثل $w = va$ با طول $n+1$ را در نظر بگیرید. آنگاه

$$\begin{aligned}(uw)^R &= (uva)^R \\ &= a(uv)^R, \quad \text{بر اساس تعریف معکوس} \\ &= av^R u^R, \quad \text{بر اساس فرض استقراء} \\ &= w^R u^R.\end{aligned}$$

پس براساس استقراء، این نتیجه در مورد تمامی رشته‌ها برقرار است.

۴- از آنجایی‌که $abaabaaabaa$ را می‌توان به رشته‌های ab, aa, baa, ab, aa تجزیه کرد که همگی در L قرار داشته باشند، بنابراین این رشته در L^* هم قرار دارد. به همین ترتیب، $baaaaabaa$ در L^* است. با این وجود، هیچ تجزیه‌ای برای $baaaaabaaaab$ ممکن نبوده و بنابراین، این رشته در L^* قرار ندارد. رشته‌های $aaaabaaaa$ و $baaaaabaa$ در L^4 قرار دارند.

۵- $\overline{L} = \{\lambda, a, b, ab, ba\} \cup \{w \in \{a,b\}^* : |w| \geq 3\}$.

۱۱- (د) ابتدا سه a تولید کرده و سپس تعداد دلخواهی a و b را در هر جای دلخواه از رشته اضافه می‌کنیم.

$$S \to AaAaAaA,$$
$$A \to aA \mid bA \mid \lambda.$$

قانون اول سه a تولید می‌کند. قانون دوم هم قادر به تولید a و b در هر موقعیتی می‌باشد. بنابراین، گرامر فوق قادر به تولید هر رشته‌ای از $w \in \{a,b\}^*$ با شرط $n_a(w) \geq 3$ می‌باشد.

۱۲-

$$S \Rightarrow aA \Rightarrow abS \Rightarrow abaA \Rightarrow ababS$$

و بنابراین، مشاهده می‌کنیم که

$$L(G) = \{(ab)^n : n \geq 0\}.$$

۱۳- چون نمی‌توان هیچ رشته پایانی از این قوانین اشتقاق کرد، $L = \emptyset$.

۱۴- (الف) ابتدا به تعداد مساوی a و b و سپس، در صورت نیاز، یک یا چند b تولید کنید.

$$S \to AB,$$
$$A \to aAb \mid \lambda,$$
$$B \to bB \mid b.$$

(د) با در نظر گرفتن

$$L_4 = \{a^{m+3}b^m : m \geq 0\}$$

به راحتی می‌توان دید که جواب زیر بدست می‌آید.

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

۱۵- (ب) برای راحت‌تر شدن مسأله، در حالت $|w| \bmod 3 = 1$ و $|w| \bmod 3 = 2$ در نظر می‌گیریم. برای حالت اول، گرامر

$$S_1 \to aaaS_1 \mid a,$$

و برای حالت دوم

$$S_2 \rightarrow aaaS_2 \mid aa$$

را پیشنهاد می‌کنیم. این دو گرامر را می‌توان با هم ترکیب و به یک گرامر به فرم زیر تبدیل کرد.

$$S \rightarrow S_1 \mid S_2.$$

۱۸- (الف) برای حل مسأله از یک نکته ابتکاری و نتایج مثال ۱-۱۳ استفاده می‌کنیم. L_1 را زبان مثال ۱-۱۳ فرض کرده و گرامر آنرا به نحوی اصلاح کنید که S_1 متغیر شروع باشد. سپس رشته دلخواه $w \in L$ را در نظر بگیرید. اگر این رشته با یک a شروع شود، آنگاه به فرم $w = aw_1$ خواهد بود که در آن، $w_1 \in L_1$. در این حالت، از گرامر $S \rightarrow aS_1$ استفاده می‌کنیم. اگر گرامر فوق با یک b آغاز شود، می‌توان آنرا بوسیله $S \rightarrow S_1S$ اشتقاق کرد.

۲۳- گرامر اول می‌تواند $S \Rightarrow SS \Rightarrow aa$ را اشتقاق کند، اما گرامر دوم قادر به اشتقاق این رشته نیست.

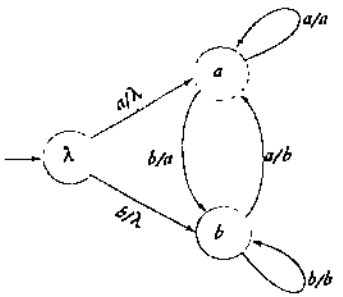
بخش ۱-۳

۱-

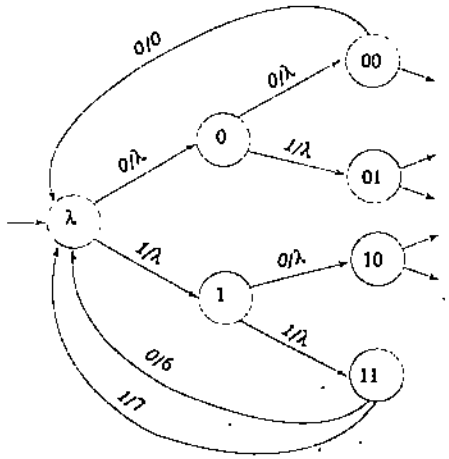
integer → sign magnitude
sign → + | − | λ
magnitud → digit | Digit magnitude
digit → 0|1|2|3|4|5|6|7|8|9

این می‌تواند یک نسخه استاندارد از C در نظر گرفته شود، چون در آن هیچ محدودیتی در مورد طول اعداد صحیح قائل نشده است. این در حالی است که اغلب کامپایلرهای واقعی در مورد تعداد ارقام محدودیت قائل می‌شوند.

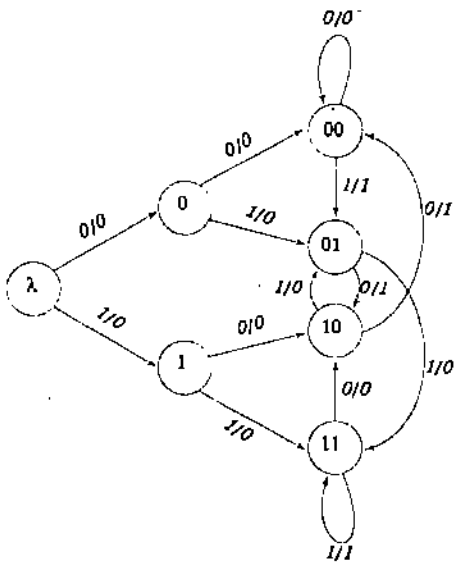
۸- اتومات باید ورودی را برای یک دوره زمانی به خاطر بسپارد تا بتوان آنرا برای خروجی بعداً دوباره تولید کرد. برای یادآوری می‌توان از روش برچسب‌دار کردن حالت بوسیله اطلاعات مربوطه استفاده کرد. به این ترتیب، برچسب حالت بعداً بعنوان خروجی تولید می‌شود.



۱۱- برای یادآوری حالت‌ها، می‌توان آنها را به صورتی برچسب‌دار کرد که به راحتی به خاطر سپرده شوند. پس از انجام یک مجموعه سه بیتی، خروجی را تولید کرده و ضمن بازگشت به ابتدا، سه بیت بعدی را پردازش می‌کنیم. با ادامه همین روند، می‌توان به جواب کامل دست یافت.



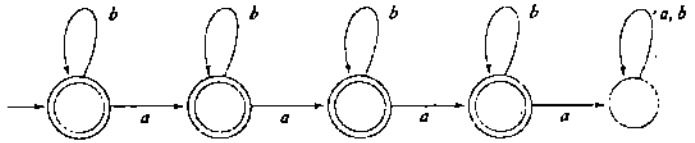
۱۲- در این مورد، مترجم باید دو سمبل ورودی قبلی را به یاد آورده و با انجام برخی انتقالات، امکان پی‌گیری اطلاعات لازم را فراهم کند.



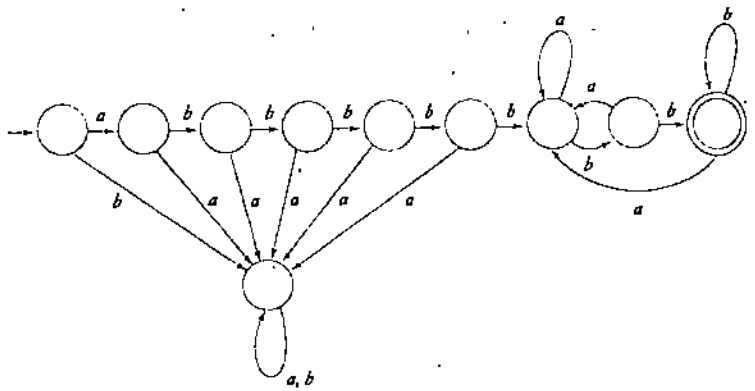
فصل ۲

بخش ۲-۱

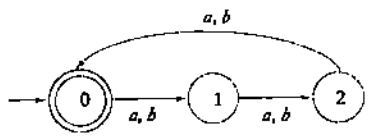
۲- (ج) مسأله را به چهار قسمت تقسیم کنید که هریک دارای یک حالت پذیرش باشد: فاقد a ، دارای یک a ، دو a و سه a به این ترتیب، چهارمین dfa ، a را به حالت تله غیرقابل قبول وارد می‌کند. یکی از جواب‌ها را در زیر مشاهده می‌کنید:



۵- (الف) اولین شش سمبل را بررسی کنید. اگر صحیح نباشند، رشته رد می‌شود. اگر پیشوند صحیح باشد، دو سمبل آخر خوانده شده را بررسی می‌کنیم. در صورتیکه پسوند آنها bb باشد، dfa در یک حالت پذیرش قرار می‌گیرد.

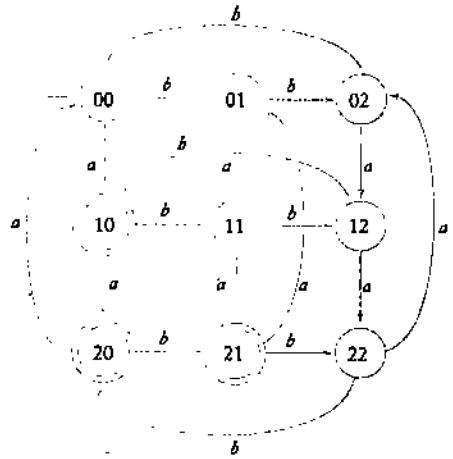


۷- (الف) از حالت‌های دارای برچسب $|w| \bmod 3$ استفاده کنید. به این ترتیب جواب به راحتی بدست می‌آید.

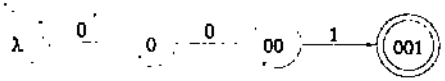


(د) در این مسأله از نه حالت استفاده می‌کنیم که در هر حالت، بخش اول آنها با $n_a(w) \bmod 3$ و بخش دوم با $n_b(w) \bmod 3$ برچسب‌دار شده باشد. به این ترتیب، به راحتی می‌توان به

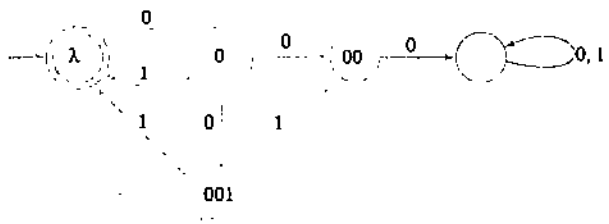
انتقالات و حالات پایانی دست یافت.



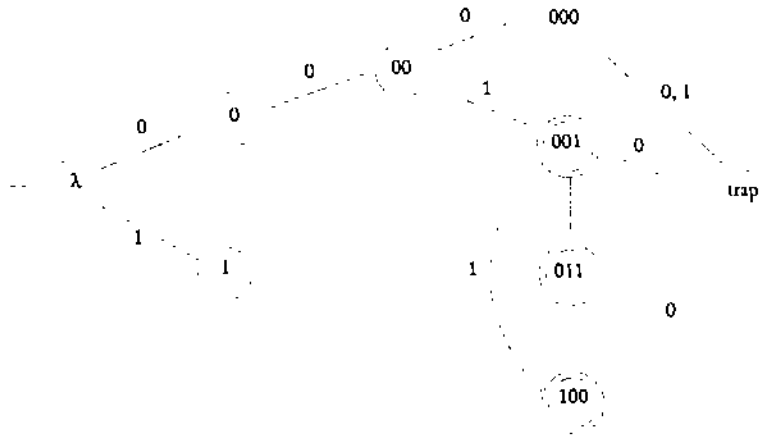
۹- (الف) برای ساخت بخش اصلی dfa ، صفرهای متوالی را شمارش کنید.



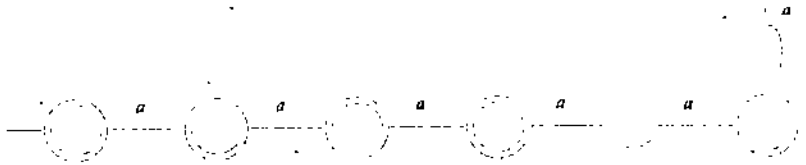
سپس، برای نگهداری صفرهای متوالی،انتقال‌های دیگری اضافه کنید و رشته‌های غیرقابل‌پذیرش را به حالت تله بفرستید.



(د) در این‌مورد باید تمام ترکیبات سه بیتی را یادآوری کنیم. اینکار نیازمند ایجاد ۸ حالت به اضافه یک حالت برای شروع است. راه‌حل مسأله، علیرغم طولانی بودن، راحت و قابل‌فهم است. نمای تقریبی از جواب مسأله ذیلاً ارائه شده است.



۱۳- راحت‌ترین روش برای حل این مسأله، ساخت یک dfa برای زبان $L = \{a^n : n = 4\}$ و سپس تکمیل راه‌حل است.

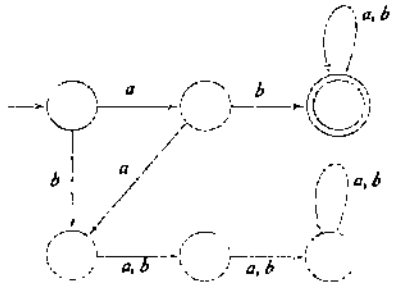


۱۴- رئوس را با دو مقدار $|w| \bmod 3$ و $|w| \bmod 5$ برچسب‌دار کنید. سپس رئوس دارای برچسب‌های 03، 20 و غیره را به عنوان حالت‌های پایانی در نظر بگیرید.

۲۳- (الف) از تناقض استفاده می‌کنیم. فرض کنید G_M در هیچکدام از مسیرهای خود از حالت‌های شروع به از حالت‌های پایانی، حلقه‌ای نداشته باشد. بنابراین، در هر قدم تعداد متناهی مرحله وجود دارد و از اینرو تمامی رشته‌های پذیرفته شده حتماً طول متناهی خواهند داشت. به طور ضمنی می‌توان نتیجه گرفت که زبان مورد نظر متناهی است.

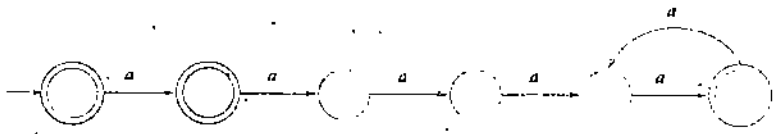
(ب) مجدداً از تناقض استفاده می‌کنیم. فرض کنید که G_M در یکی از مسیرهای خود از حالت شروع به یکی از حالت‌های پایانی، دارای یک حلقه باشد. بنابراین، می‌توان از این حلقه برای تولید قدم با طول دلخواه با یکی از رشته‌های مورد پایانی استفاده کرد. اما، همانطور که می‌دانید، استفاده از رشته‌های با طول دلخواه در زبان‌های متناهی مجاز نیست.

۲۵- چندین راه‌حل وجود دارد که یکی از آنها در اینجا داده شده است.



بخش ۲-۲

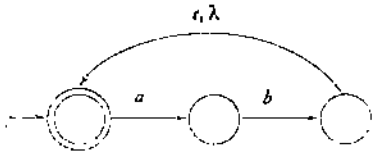
۳- مکمل زبان شکل ۲-۸، $\{a^n : n \neq 3$ ، n فرد است $\}$ می‌باشد. یکی از dfaهای مربوط به این زبان به شکل زیر می‌باشد.



توجه داشته باشید که نمی‌توان برای مجموعه حالت پایانی شکل ۲-۸ مکمل ارائه کرد.

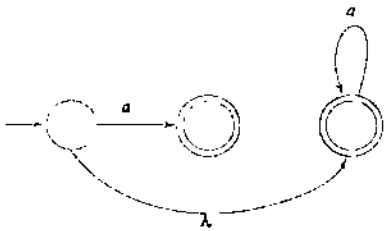
۵- $\delta^*(q_0, a) = \{q_0, q_1, q_2\}, \delta^*(q_1, \lambda) = \{q_0, q_1, q_2\}$.

۸- راه‌حل چهار حالتی به راحتی بدست می‌آید. اما پیداکردن راه‌حل سه حالتی نیاز به کمی تجزیه و دقت بیشتر دارد. یکی از پاسخهای مسأله در زیر داده شده است.



۹- خیر. در رشته abc سه سمبل مختلف وجود دارد و به هیچ روشی نمی‌توان این رشته را با کمتر از سه حالت پذیرفت.

۱۶- این مسأله یکی از نمونه مسائلی است که باید راه‌های مختلفی را برای آن امتحان کرد. البته تعداد زیادی از این روش‌ها نادرست هستند. در زیر یکی از روش‌های درست ارائه شده است.

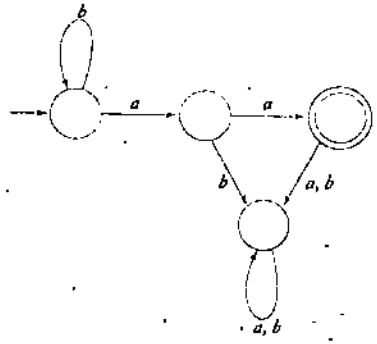


۱۸- فقط یک حالت شروع p_0 را معرفی کنید. سپس انتقال زیر را اضافه کنید.

$$\delta(p_0,\lambda)=Q_0.$$

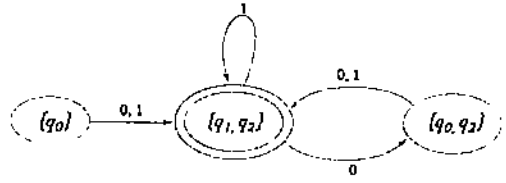
در مرحله بعد، وضعیت حالت شروع را از Q_0 حذف کنید. به راحتی می‌توان مشاهده کرد که nfa حاصل هم‌ارز با nfa اصلی است.

۲۱- یک حالت تله غیرقابل قبول را معرفی کرده و تمام انتقالات تعریف نشده را به این حالت جدید وصل می‌کنیم. جواب:



بخش ۲-۳

۲- فقط لازم است روال nfa به dfa را اجرا کنید. به این ترتیب dfa زیر بدست می‌آید.



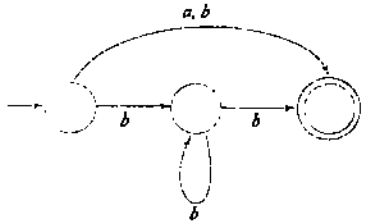
۷- یک حالت پایانی جدید به نام p_f ایجاد کرده و به ازای هر $q\in F$، انتقالات

$$\delta(q,\lambda)=\{p_f\}$$

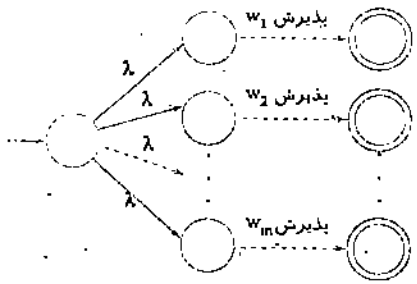
را اضافه کنید. سپس p_f را تنها حالت پایانی در نظر بگیرید. به راحتی می‌توان بحث کرد که اگر در ابتدا $\delta^*(q_0,w)\in F$ باشد، آنگاه پس از اصلاح داریم $\delta^*(q_0,w)=\{p_f\}$ است و بنابراین dfa اولیه و اصلاح شده با هم متناظر هستند.

بدلیل آنکه برای این ساخت باید چند انتقال λ را انجام داد، نمی‌توان آنرا در dfa ها اعمال کرد. غالباً، فقط یک حالت پایانی در dfa وجود ندارد. برای تحقیق در مورد صحت و سقم این ادعا، dfa یی بسازید که $\{\lambda,a\}$ را پذیرش کند.

۸- بدست آوردن پاسخ مستلزم کمی فکر و دقت است. یکی از پاسخ‌ها را در زیر مشاهده می‌کنید.

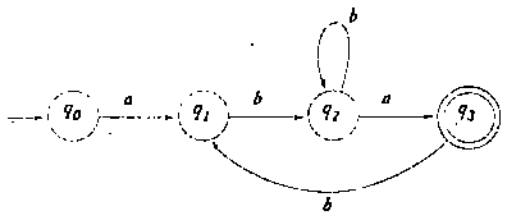


۱۱- فرض کنید که $L=\{w_1,w_2,\ldots,w_m\}$ باشد. آنگاه nfa :

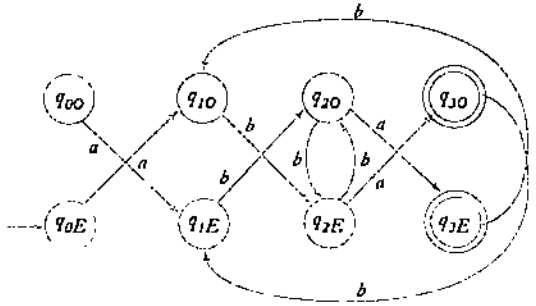


L را می‌پذیزد و به همین دلیل، زبان مذکور منظم است.

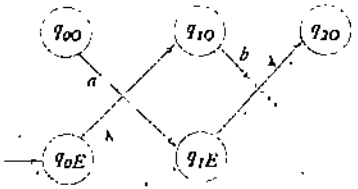
۱۴- تحقیق در این‌مورد کمی مشکل است. نکته مسأله، استفاده از یک dfa برای L است. بعلاوه، باید آنرا به گونه‌ای اصلاح کرد که هر وقت که تعداد زوج یا تعداد فردی از سمبل‌ها خوانده می‌شود، به خاطر سپرده شود. برای اینکار تعداد حالت‌ها را دو برابر کرده و O یا E را به برچسب‌ها اضافه کنید. بعنوان مثال، اگر بخشی از dfa به صورت



باشد، نظیر آن به شکل زیر خواهد بود.



حال، با انجام چند انتقال λ، انتقالات زوج از یکی از حالت‌های E را با یکی از حالت‌های O جایگذاری کنید.



پس از چندین مثال متوجه می‌شوید که اگر dfa اصلی $a_1a_2a_3a_4$ را بپذیرد، اتومات جدید و در نتیجه $even(L)$ هم ... $\lambda a_2\lambda a_4$ را خواهد پذیرفت.

۱۵- فرض کنید dfa یی داشته باشیم که L را بپذیرد. سپس

الف) تمام حالت‌های $\overline{Q}$ را شناسایی می‌کنیم که از q_0 قابل دسترسی بوده و هر پیشوند دو سمبلی v را بخواند؛ یعنی

$$\overline{Q}=\{q\in Q:\delta^*(q_0,v)=q, |v|=2\}.$$

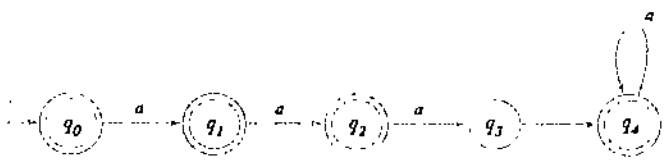
ب)یک حالت شروع جدید به نام p_0 را معرفی کرده و

$$\delta(p_0,\lambda)=\overline{Q}$$

را اضافه می‌کنیم. nfa ی جدید $chop2(L)$ را می‌پذیرد. هرچند به راحتی می‌توان به ساخت موردنظر دست یافت، برای رسیدن به جواب کامل باید عبارت آخر را اثبات کنیم.

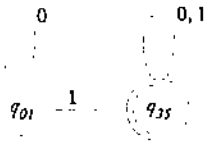
بخش ۲-۴

۲- (ج)



dfa فوق به دلیل زیر کمینه است. $q_3\notin F$ و $q_4\in F$ و بنابراین q_3 و q_4 ادغام‌ناپذیر هستند. سپس، $\delta^*(q_2,a)\notin F$ و $\delta^*(q_4,a)\in F$ و بنابراین q_2 و q_4 هم ادغام‌ناپذیر هستند. به همین ترتیب، $\delta^*(q_1,aa)\notin F$ و $\delta^*(q_3,aa)\in F$ و بنابراین q_1 و q_3 ادغام‌ناپذیر هستند. با ادامه این روند مشاهده می‌کنیم که تمامی حالات ادغام‌ناپذیر بوده و بنابراین dfa مذکور کمینه است.

۴- در درجه اول، حالات غیرقابل‌دسترسی q_2 و q_4 را حذف کنید. سپس با استفاده از روال *نشانه‌گذاری*، زوج‌های غ ادغام‌پذیر (q_0,q_1) و (q_3,q_5) را پیدا کنید. به این ترتیب dfa کمینه زیر بدست می‌آید.



۶- از تناقض استفاده می‌کنیم. فرض کنید که $\hat{M}$ کمینه نباشد. بنابراین می‌توان dfaی کوچکتر $\tilde{M}$ را ایجاد کرد که $\overline{L}$ را بپذیرد. در $\tilde{M}$، با مکمل‌گیری از مجموعه حالت پایانی، یک dfa برای L ایجاد کنید. dfaی حاصل کوچکتر از M است که این با فرض کمینه بودن M در تضاد می‌باشد.

۱۰- با استفاده از تناقض، فرض کنید که q_b و q_c ادغام‌پذیر باشند. از آنجایی‌که q_a و q_b ادغام‌پذیر هستند و ادغام‌پذیربودن هم یک رابطه متناظر است(تمرین ۷)، q_a و q_c حتماً ادغام‌پذیر می‌باشند.

فصل ۳

بخش ۳-۱

۲- بله، چون $(\lambda+0)(0+1)(0+1)^*$ به تمام رشته‌های 0 و 1 دلالت می‌کند.

۶- (الف) مسأله را به حالت‌های $m=0,1,2,3$ تقسیم کنید. تعداد 4 یا بیشتر a تولید کرده و پس از آن به تعداد لازم b قرار دهید. جواب: $(\lambda+b+bb+bbb)aaaaa^*$.

(ج) پیدا کردن مکمل زبان ۶ (الف) چندان آسان نیست. رشته‌های به فرم a^nb^m و باشرایط $n<4$ یا $m>3$، در L وجود ندارند. اما این تعریف کاملی برای $\overline{L}$ محسوب نمی‌شود. همچنین باید حالتی را در نظر بگیریم که پس از یک a، یک b قرار می‌گیرد.

$$(\lambda+a+aa+aaa)b^*+a^*bbbbb^*+(a+b)^*ba(a+b)^*.$$

۱۰- اینکار را طی سه مرحله انجام دهید: (الف) $m=1,n\geq 3$، (ب) $n\geq 2,m\geq 2$ و (ج) $n=1,m\geq 3$. راه‌حل هریک هم به راحتی بدست می‌آید.

۱۳- تمام حالت‌های با ضابطه $|v|=2$ را شمارش کنید تا رابطه زیر بدست آید.

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

۱۶- (ج) فقط لازم است که هر یک از سمبل‌ها را یک مرتبه بدست آورید. این کار بوسیله عبارت

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

قابل تعریف است. اما در اینصورت a پیش از b قرار خواهد گرفت و الی آخر. برای بدست آوردن جواب کامل باید تمام جایگشت‌های سه سمبلی را ایجاد کنید. شش عبارت حاصله را باید با هم جمع نمود. جواب مسأله علیرغم طولانی بودن، از نظر مفهومی ساده و به راحتی قابل تعریف است.

۱۷- (ج) دو 0 در نظر بگیرید که چند 1 در میان آنها قرار گرفته باشد و سپس این کار را تکرار کنید. حالت فاقد صفر را هم در نظر بگیرید. جواب: $1^*+1^*(1^*01^*01^*)$.

۱۸- (الف) تمام رشته‌های ممکن با طول سه را ایجاد و تکرار کنید. یکی از جواب‌های کوتاه این مسأله $((a+b+c)(a+b+c)(a+b+c))^*$ می‌باشد.

۲۰- (ج) عبارت

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

صحیح است. براساس قوانین، $(r_1+r_2)^*$ به زبان $(L(r_1)\cup L(r_2))^*$ دلالت دارد، یعنی مجموعه تمام رشته‌هایی که با الحاق‌های دلخواه اعضای $L(r_1)$ به $L(r_2)$ بدست می‌آید. اما $(r_1^*r_2^*)^*$ به $(L(r_1))^*(L(r_2))^*)$ دلالت دارد که همان مجموعه مورد نظر است.

۲۳- در عبارت مربوط به یک زبان نامتناهی باید حداقل یک زیرعبارت ستاره‌دار وجود داشته باشد، وگرنه فقط به رشته‌های متناهی دلالت می‌کند. اگر یکی از زیرعبارت‌های ستاره‌دار به یک رشته غیرتهی اشاره کند، آنگاه می‌توان این رشته را به تعداد لازم تکرار کرد و به این صورت، به رشته‌های با طول دلخواه اشاره نمود.

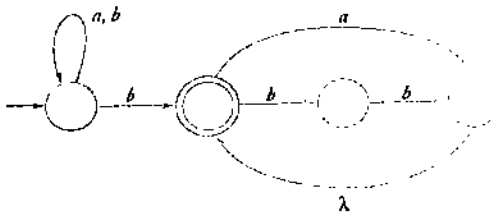
۲۵- یک نمای بسته را می‌توان بوسیله عبارت دلخواه r تولید کرد اگر و تنها اگر $n_a(r)=n_b(r)$ و $n_a(r)=n_d(r)$.

۲۷- به چند نکته توجه کنید. طول رشته بیتی باید حداقل ۶ بیت باشد. اگر طولانی‌تر از ۶ بیت باشد، ارزشی حداقل معادل ۶۴ خواهد داشت و بنابراین هر چیزی ممکن خواهد بود. اما اگر دقیقاً ۶ بیت باشد، آنگاه یا بیت دوم از چپ (۱۶) یا بیت سوم از چپ (۸) باید ۱ باشد. به این ترتیب، جواب به راحتی بدست می‌آید.

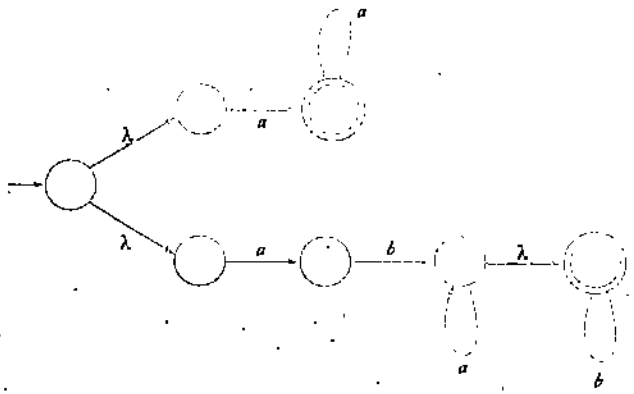
$$(111+110+101)(0+1)(0+1)+1(0+1)(0+1)(0+1)(0+1)(0+1)(0+1)(0+1)^*$$

بخش ۳-۲

۳- اگر برای حل این مسأله از اصول اولیه استفاده کنید، دیگر نیازی به ساخت یک عبارت منظم برای nfa وجود ندارد. البته، می‌توان عبارت منظم هم ایجاد کرد، اما اینکار فقط باعث پیچیدگی بیشتر جواب خواهد شد. جواب:

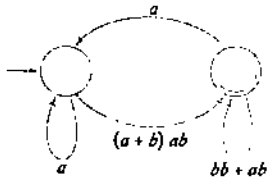


۴- (الف) کار را با



شروع کرده و سپس به ترتیب گفته شده، از الگوریتم تبدیل nfa به dfa استفاده کنید.

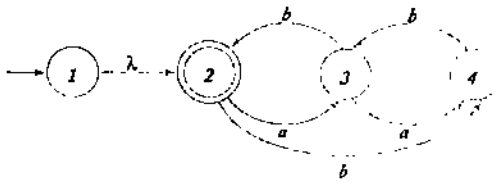
۸- با حذف رأس میانی شکل زیر بدست می‌آید.



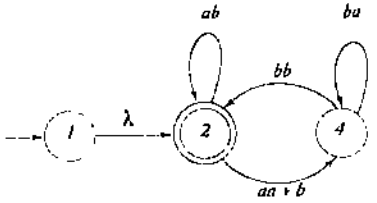
پس، بر اساس معادله(۳-۱)، زبان پذیرفته شده در $L(r)$ قرار خواهد داشت و

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

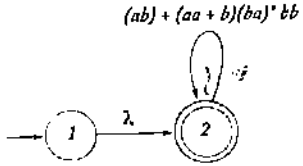
۱۰- (ب) ابتدا، باید nfa را به نحوی اصلاح کرد که در شرایط اشاره شده در ساخت قضیه ۳-۲ صدق کند. براساس یکی از این شرایط، $q_0 \notin F$ است. این کار به راحتی قابل انجام است.



سپس حالت ۳ را حذف کنید.

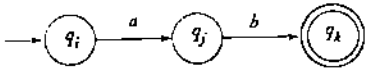


در قدم بعد، حالت ۴ را حذف کنید.

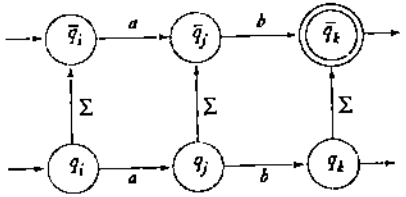


در پایان، عبارت منظم $r=(ab+(aa+b)(ba)^*bb)^*$ بدست می‌آید.

۱۶- (الف) برای حل این مسأله باید یک نکته را رعایت کنید. کار را با یک dfa با حالت‌های $q_0,q_1,\ldots$ آغاز کرده و یک اتومات "موازی" با حالت‌های $\bar{q}_0,\bar{q}_1,\ldots$ معرفی کنید. سپس سمبل نادرست را به صورت نامعین از یکی از حالت‌های اتومات اصلی به حالت نظیر در بخش موازی انتقال دهید. بعنوان مثال، اگر بخشی از dfa اصلی مشابه



باشد، آنگاه dfa مربوط به موازی آن، nfa خواهد بود که بخش نظیر آن به صورت زیر می‌باشد.



به راحتی می‌توان استدلال کرد که dfa اصلی L را می‌پذیرد اگر و تنها اگر nfa جدید $insert(L)$ را بپذیرد.

بخش ۳-۳

۴- گرامر خطی از راست:

$$S \to aaA$$
$$A \to aA \mid B$$
$$B \to bbbC$$
$$C \to bC \mid \lambda$$

گرامر خطی از چپ

$$S \to Abbb$$
$$A \to Ab \mid B$$
$$B \to aaC$$
$$C \to aC \mid \lambda$$

۸- می‌توان با استفاده از استقراء نشان داد که اگر w یکی از فرم‌های جمله‌ای اشتقاق‌شده از G باشد، آنگاه w^R را می‌توان طی همان تعداد مراحل از $\hat{G}$ تولید کرد.

بدلیل آنکه w از اشتقاق‌های خطی‌چپ تولید شده است، به فرم $w=Aw_1$ و ضابطه $A\in V$ و $w_1\in T^*$ خواهد بود. براساس فرض استقراء، $w^R=w_1^RA$ را می‌توان از طریق $\hat{G}$ اشتقاق کرد. حال اگر $A\to Bv$ را اعمال کنیم، آنگاه

$$w \Rightarrow Bvw_1.$$

اما به دلیل وجود قانون $A\to v^RB$ در $\hat{G}$، می‌توان به صورت

$$w^R \to w_1^Rv^RB$$
$$\to (Bvw_1)^R$$

اشتقاق کرد و به این ترتیب، گام استقراء به پایان می‌رسد.

۱۱- مسأله را به دو قسمت تقسیم کنید: (الف) n و m هر دو زوج باشند و (ب) n و m هر دو فرد باشند. با توجه به قسمت اول، جواب

$$S \to aaS \mid A$$
$$A \to bbA \mid \lambda$$

به راحتی بدست می‌آید.

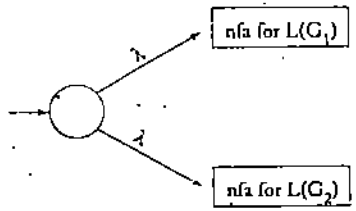
۱۳- (الف) ابتدا یک dfa برای L بسازید. این کار به راحتی قابل انجام بوده و انتقالات زیر بدست می‌آید.

$$\delta(q_0,a)=q_1, \delta(q_0,b)=q_2,$$
$$\delta(q_1,a)=q_0, \delta(q_1,b)=q_3,$$
$$\delta(q_2,a)=q_3, \delta(q_2,b)=q_0,$$
$$\delta(q_3,a)=q_2, \delta(q_3,b)=q_1,$$

که در آن، q_0 حالت شروع و پایانی است. سپس، با استفاده از قضیه ۳-۴ جواب زیر بدست می‌آید.

$$q_0 \to aq_1 \mid bq_2 \mid \lambda,$$
$$q_1 \to bq_3 \mid aq_0,$$
$$q_2 \to aq_3 \mid bq_0,$$
$$q_3 \to aq_2 \mid bq_1.$$

۱۷- مشخص است که $L(G_1)$ و $L(G_2)$ هر دو منظم هستند. می‌توان با ساخت dfa زیر، نشان داد که اجتماع آنها هم منظم است.



شرط جدا از هم بودن V_1 و V_2 ضروری است، چون در غیراینصورت دو dfa جدا از هم نخواهند بود.

فصل ۴

بخش ۴-۱

۲- (الف) روش ساخت علیرغم طولانی بودن، ساده و قابل فهم است. می‌توان با در اختیار داشتن

$$\delta(q_0,a)=q_1,\quad \delta(q_0,b)=q_1,\quad \delta(q_1,a)=q_1,\quad \delta(q_1,b)=q_t,$$

یک dfa برای $L((a+b)a^*)$ ایجاد کرد که در آن، q_t یک حالت تله و q_1 حالت پایانی است. یک dfa برای $L(baa^*)$ از روابط

$$\delta(p_0,a)=p_1,\delta(p_0,b)=p_t,\delta(p_1,a)=p_2,$$
$$\delta(p_1,b)=p_t,\delta(p_2,a)=p_2,\delta(p_2,b)=p_t$$

با حالت پایانی p_2 ایجاد کرد. از آنهم می‌توان

$$\delta((q_0,p_0),a)=(q_1,p_t)=,\delta((q_0,p_0),b)=(q_t,p_t),$$
$$\delta((q_1,p_t),a)=(q_1,p_2),\delta((q_1,p_2),a)=(q_1,p_2),$$

و الی آخر، ایجاد کرد. با تکمیل ساخت، مشاهده می‌کنیم که تنها حالت پایانی (q_1,p_1) بوده و بعلاوه، $L((a+b)a^*)\cap L(baa^*)=L(baa^*)$ است.

۷- توجه داشته باشد که

$$nor(L_1,L_2)=\overline{L_1\cup L_2}$$

بنابراین می‌توان با بسته‌بودن تحت مکمل‌گیری و اشتراک به جواب رسید.

۱۲- جواب مثبت است. کار را از بررسی مفاهیم مجموعه

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

آغاز می‌کنیم. نکته اصلی این است که به دلیل متناهی بودن L_1، $L_1\cap L_2$ هم متناهی بوده و بنابراین به ازای هر L_2، عبارت منظم است. مابقی جواب هم به راحتی از طریق بسته‌بودن تحت اجتماع و مکمل‌گیری بدست می‌آید.

۱۴- بر اساس بسته‌بودن تحت معکوس، L^R منظم است. بنابراین می‌توانیم با بسته‌بودن تحت الحاق به نتیجه برسیم.

۱۶- برای حل مسأله از $L_1=\Sigma^*$ استفاده کنید. آنگاه، به ازای هر L_2، $L_1\cup L_2=\Sigma^*$، که منظم است. بنابراین عبارت داده شده به طور ضمنی بیانگر منظم بودن L_2 است.

۱۸- می‌توان از روش ساخت زیر استفاده کرد. تمام حالت‌های P را پیدا کنید که در آنها، یک مسیر از رأس شروع به یکی از اعضای P و از آن عضو به یکی از حالتهای پایانی وجود داشته باشد. سپس، هر یک از اعضای P را به حالت پایانی تبدیل کنید.

۲۶- فرض کنید $G_1=(V_1,T,S_1,P_1)$ و $G_2=(V_2,T,S_2,P_2)$ باشد. ضمن حفظ کلیت می‌توان V_1 و V_2 را مجزا در نظر گرفت. دو گرامر را با هم ترکیب کرده و

(الف) S را بعنوان سمبل شروع جدید در نظر بگیرید و قوانین $S\to S_1\mid S_2$ را اضافه کنید.

(ب) در P_1، تمامی قوانین به فرم $A\to x$ با ضابطه $A\in V_1$ و $x\in T^*$ را با $A\to xS_2$ جایگزین کنید.

(ج) در P_1، تمامی قوانین به فرم $A\to x$ با ضابطه $A\in V_1$ و $x\in T^*$ را با $A\to xS_1$ جایگزین کنید.

بخش ۴-۲

۱- از آنجایی که در مثال ۴-۱، L_1-L_2 منظم بود، بنابراین، یک الگوریتم عضویت هم به ازای آن وجود دارد.

۲- اگر $L_1\subseteq L_2$، آنگاه $L_1\cup L_2=L_2$. بدلیل آنکه $L_1\cup L_2$ منظم است و یک الگوریتم برای تساوی دو مجموعه در اختیار داریم، از آن می‌توان بعنوان الگوریتم زیرمجموعه‌بودن مجموعه،استفاده کرد.

۵- از dfa مربوط به L با استفاده از ساخت پیشنهادی در قضیه ۴-۲، یک dfa برای L^R بسازید. سپس از الگوریتم تساوی در قضیه ۴-۷ استفاده کنید.

۱۲- باید یک نکته را بدانید و آنرا رعایت کنید. اگر در L هیچ رشته‌ای با طول زوج وجود نداشته باشد، آنگاه

$$L \cap L((aa+ab+ba+bb)^*) = \varnothing.$$

و سمت چپ این رابطه منظم است، بنابراین می‌توان از قضیه ۴-۶ استفاده کرد.

بخش ۴-۳

۲- برای آنکه dfa مربوط به L قادر به پردازش رشته میانی v باشد، باید یک قدم با طول $|v|$ در گراف انتقال وجود داشته باشد. بدلیل آنکه طول این قدم بیشتر از تعداد حالت‌های dfa می‌باشد، حتماً یک دور با برچسب y در آن وجود دارد. ناگفته پیداست که می‌توان این حلقه را به هر تعداد لازم تکرار کرد، بدون آنکه قابلیت پذیرش رشته تغییر کند.

۴- (الف) با معلوم بودن m ، $w = a^m b^m a^{2m}$ را انتخاب کنید. بنابراین، رشته y باید a^k بوده و رشته‌های تزریق شده هم

$$w_i = a^{m+(i-1)k} b^m a^{2m}$$

باشند.

اگر $i \geq 2$ را در نظر بگیریم، آنگاه $m+(i-1)k > m$ خواهد بود و بنابراین w_i در L وجود ندارد.

(ه) چون نمی‌توان به راحتی لم تزریق را مستقیماً اعمال کرد، از روش غیرمستقیم استفاده می‌کنیم. فرض کنید که L منظم باشد. آنگاه براساس بسته‌بودن زبان‌های منظم تحت مکمل‌گیری، $\overline{L}$ هم منظم خواهد بود. اما $\overline{L} = \{w : n_a(w) = n_b(w)\}$ که می‌توان نامنظم بودن آنرا اثبات کرد. براساس تناقض، L منظم نیست.

۵- (الف) p را کوچکترین عدد اول بزرگتر یا مساوی m در نظر گرفته و $w = a^p$ را در نظر بگیرید. در این صورت، y رشته‌ای از a های با طول k است، بطوریکه

$$w_i = a^{p+(i-1)k}.$$

اگر $i-1 = p$ را در نظر بگیریم، آنگاه $p+(i-1)k = p(k+1)$ مرکب بوده و w_{p+1} در این زبان وجود ندارد.

۱۴- فرض نادرست است. بعنوان مثال نقض، $L_1 = \{a^n b^m : n \leq m\}$ و $L_2 = \{a^n b^m : n > m\}$ را در نظر بگیرید. مشاهده می‌کنید که هر دو نامنظم هستند. اما $L_1 \cup L_2 = L(a^* b^*)$، منظم است.

۱۵- (الف) زبان منظم است. بهترین راه برای تحقیق در این‌مورد، تقسیم مسأله به حالت‌هایی مثل $l = 0, k = 0, n > 5$ است. در اینصورت، می‌توان برای هریک عبارات منظمی را ارائه کرد.

(ب) این زبان منظم نیست. اگر $w = aaaaaab^m a^m$ را انتخاب کنیم، حریف چندین گزینه پیش‌رو

دارد. اگر y فقط از a تشکیل شده باشد، برای نقض شرط $n > 5$، از $i = 0$ استفاده می‌کنیم. اگر حریف y هایی را انتخاب کند که فقط از b تشکیل شده باشند، می‌توان شرط $k \leq l$ را نقض کرد.

۱۷- L منظم است. برای بررسی صحت آن می‌توان از $L = L_1 \cap L_1^R$ و بسته‌بودن‌های شناخته شده برای زبان‌های منظم استفاده کرد.

۱۹- (الف) زبان منظم است؛ چون هر رشته‌ای که دارای دو سمبل متوالی یکسان باشد، در این زبان قرار دارد. یکی از عبارات منظم برای L $(a+b)(a+b)^*(aa+bb)(a+b)(a+b)^*$ است.

(ب) زبان منظم نیست. فرض کنید $w = (ab)^m aa(ba)^m$. حال حریف چندین گزینه مثل $y = (ab)^4$ یا $y = (ab)^4 a$ برای انتخاب دارد. در حالت اول،

$$w_0 = (ab)^{m-k} aa(ba)^m.$$

بدلیل آنکه فقط $ww^R = (ab)^l aa(ba)^l$ قابل شناسایی است، پیشوند u کوتاهتر از پسوند v بوده و w_0 در L قرار ندارد. در حالت دوم، طول w_0 فرد است، بنابراین در L هم وجود ندارد.

۲۱- فرض کنید $L_i = a^i b^i, i = 0,1,\ldots$. به ازای هر i ، L_i متناهی و در نتیجه، منظم است. اما اجتماع تمام زبان‌ها باعث ایجاد زبان غیرمنظم $L = \{a^n b^n : n \geq 0\}$ می‌شود.

۲۵- مثلث قائم‌الزاویه را به صورت $u^n r^m d^n l^m$ تعریف می‌کنیم که با استفاده از لم تزریق منظم نیست.

فصل ۵

بخش ۵-۱

۴- کاملاً مشخص است که هر رشته‌ای که بوسیله این گرامر تولید شود، تعداد برابری a و b دارد. برای اثبات اینکه شرط پیشوند بودن $n_a(v) \geq n_b(v)$ برقرار است، یک استقراء روی طول اشتقاق انجام می‌دهیم. فرض کنید که این شرط، به ازای هر فرم جمله‌ای اشتقاق‌شده از S طی n مرحله، صدق می‌کند. برای تولید یک فرم جمله‌ای طی $n+1$ مرحله، می‌توان $S \to \lambda$ یا $S \to SS$ را $S \to aSb$ را اعمال کرد. از آنجاییکه هیچکدام از ایندو فرم جمله‌ای تعداد a ها یا b ها یا محل a ها یا b های فعلاً موجود را تغییر نمی‌دهد، شرط پیشوند بودن هنوز هم به قوت خود باقی است. در روش دیگر، $S \to aSb$ را اعمال می‌کنیم. با این کار، یک a دیگر و یک b دیگر اضافه می‌شود، اما به‌دلیل‌آنکه a افزوده شده در سمت چپ b افزوده شده قرار دارد، شرط پیشوندی باز هم برقرار است. بنابراین، اگر شرط پیشوندی طی n مرحله هنوز هم باقی باشد، پس از $n+1$ مرحله هم صدق می‌کند. بدلیل آنکه شرط پیشوندی پس از یک مرحله هنوز هم وجود دارد، پایه استقراء را در اختیار داریم و می‌توان مابقی استقراء را به راحتی انجام داد.

۷- (الف) ابتدا، حالت $n = m+3$ را حل کنید. سپس چند b دیگر اضافه کنید. این کار را می‌توان بوسیله

$$S \to aaaA,$$
$$A \to aAb \mid B,$$
$$B \to Bb \mid \lambda$$

انجام داد. اما چون به این ترتیب حداقل سه a بوجود می‌آید، جواب هنوز کامل نیست. برای بررسی حالت‌های $n = 0,1,2,$ ،

$$S \to \lambda \mid aA \mid aaA,$$

را اضافه می‌کنیم.

(د) جواب بسیار به راحتی بدست می‌آید.

$$S \to aSbb \mid aSbbb \mid \lambda.$$

این قوانین به صورت نامعین باعث ایجاد bb یا bbb به ازای هر a تولیدی می‌شود.

۸- (الف) در مورد اول $n = m$ و k اختیاری است. برای ایجاد آن از روابط زیر استفاده می‌کنیم.

$$S_1 \to AC,$$
$$A \to aAb \mid \lambda,$$
$$C \to Cc \mid \lambda.$$

درمورد دوم، n اختیاری بوده و $m \le k$ می‌باشد. در اینجا از

$$S_2 \to BD,$$
$$B \to aB \mid \lambda,$$
$$D \to bDc \mid E,$$
$$E \to Ec \mid \lambda$$

استفاده می‌کنیم. در نهایت، قوانین را با $S \to S_1 \mid S_2$ شروع می‌کنیم.

(ه) مسأله را به دو قسمت تقسیم کنید: $n = k + m$ و $m = k + n$. حالت اول با

$$S \to aSc \mid S_1 \mid \lambda,$$
$$S_1 \to aS_1b \mid \lambda,$$

حل می‌شود.

۱۳- اگر L از S اشتقاق شود، آنگاه $S_1 \to SS$، L^2 را اشتقاق می‌کند.

۱۶- معمولاً نمی‌توان مستقیماً از یکی از گرامرهای L جهت ایجاد گرامری برای $\overline{L}$ استفاده کرد، بنابراین باید یک شرح ترجیحاً بازگشتی برای $\overline{L}$ بوجود آورد. تحقیق این امر چندان ساده نیست. در یکی از زیرمجموعه‌های مشخص $\overline{L}$ رشته‌هایی با طول فرد وجود دارد، اما این جواب کامل نیست.

فرض کنید رشته‌ای با طول زوج در اختیار داشته باشیم که به فرم ww^R نباشد. با حرکت همزمان از مرکز به سمت چپ و راست، سمبل‌های نظیر را با هم مقایسه کنید. در حالیکه بخشی در اطراف مرکز ممکن است به فرم ww^R باشد، در جایی به یک a در سمت چپ و به یک b در محل نظیر آن در راست مواجه می‌شویم و برعکس. بنابراین، رشته مذکور باید به فرم $uawv^Rbv$ یا $ubwv^Rav$ باشد که در آن، $u \models v \mid$ می‌باشد. در اینصورت، می‌توان گرامرهایی برای این نوع رشته‌ها ایجاد نمود. یکی از جواب به صورت زیر است:

$$S \to ASA \mid B,$$
$$A \to a \mid b,$$
$$B \to bCa \mid aCb,$$
$$C \to aCa \mid bCb \mid \lambda.$$

در قانون اول u و v و قانون سوم هم دو علائم غیرمتشابه بوجود می‌آورند و در نهایت قانون آخر داخلی‌ترین تقارن را ایجاد می‌کند.

۲۰- تنها اشتقاق ممکن با

$$S \Rightarrow aaB \Rightarrow aaAa \Rightarrow aabBba \Rightarrow aabAaba$$

آغاز می‌شود. اما این فرم‌جمله‌ای دارای پسوند aba است و بنابراین احتمالاً به جمله $aabbabba$ ختم نخواهد شد.

۲۳- $E \to E + E \mid E.E \mid E^* \mid (E) \mid \lambda \mid \varnothing \mid a \mid b.$

بخش ۵-۲

۲- یکی از جواب‌ها به صورت زیر است.

$$S \to aA, A \to aAB \mid b, B \to b.$$

۶- دو اشتقاق چپ‌ترین برای $w = aab$ وجود دارد.

$$S \Rightarrow aaB \Rightarrow aab,$$
$$S \Rightarrow AB \Rightarrow AaB \Rightarrow aaB \Rightarrow aab.$$

۹- می‌توان براساس dfa مربوط به زبان‌های منظم و روش قضیه ۳-۴، یک گرامر منظم ایجاد نمود. این گرامر، جز به ازای $q_f \to \lambda$، یک s – گرامر است. اما این قانون هیچ ابهامی ایجاد نمی‌کند. چون dfa انتخابی ندارد، هرگز نمی‌توان قانون قابل اعمال را انتخاب کرد.

۱۴- ابهام گرامر در اشتقاق‌های

$$S \Rightarrow aSb \Rightarrow ab$$
$$S \Rightarrow SS \Rightarrow abS \Rightarrow ab$$

کاملاً نمایان است. یکی از گرامرهای غیرمبهم نظیر

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

می‌باشد. تحقیق در مورد غیرمبهم بودن این گرامر چندان راحت نیست. برای اثبات، دو حالت خاص $w = aabb$ را که با شروع از $A \to aAb$ اشتقاق می‌شود و $w = abab$ که فقط با شروع از $A \to aAb$ اشتقاق می‌شود، در نظر بگیرید. رشته‌های پیچیده‌تر را از ایندو حالت ایجاد می‌کنیم تا بتوان آنها را فقط به یک روش تجزیه نمود.

۲۰- جواب:

$$S \to aA \mid aAA,$$
$$A \to bAb \mid bb.$$

فصل ۶

بخش ۶-۱

۳- با استفاده از قانون قضیه ۶-۱، B را در گرامر اول جایگذاری کنید. آنگاه B غیرمفید شده و قوانین مربوطه را می‌توان حذف نمود. بر اساس قضایای ۶-۱ و ۶-۲ ایندو گرامر متناظر هستند.

۸- تنها متغیر قابل تهی A است، بنابراین با حذف قوانین λ داریم:

$$S \to aA \mid a \mid aBB,$$
$$A \to aaA \mid aa,$$
$$B \to bC \mid bbC,$$
$$C \to B.$$

$C \to B$ قانون‌واحد بوده و با حذف آن اشتقاق‌های زیر بدست می‌آید.

$$S \to aA \mid a \mid aBB,$$
$$A \to aaA \mid aa,$$
$$B \to bC \mid bbC,$$
$$C \to bC \mid bbC.$$

نهایتاً اینکه B و C بی‌فایده هستند. بنابراین داریم:

$$S \to aA \mid a,$$
$$A \to aaA \mid aa.$$

زبان تولید شده بوسیله این گرامر $L((aa)^*a)$ می‌باشد.

۱۳- $L(\hat{G}) = L(G) - \{\lambda\}$.

۱۵- بعنوان یک نمونه می‌توان به

$$S \to aA,$$
$$A \to BB,$$
$$B \to aBb \mid \lambda$$

اشاره کرد. باحذف قوانین λ جواب زیر بدست می‌آید.

$$S \to aA \mid a,$$
$$A \to BB \mid B,$$
$$B \to aBb \mid ab.$$

۱۷- این امر بدیهی است، چون با حذف قوانین بی‌فایده، هیچ چیزی به گرامر افزوده نمی‌شود.

۲۲- در گرامر $S \to aA$؛ $A \to a$ هیچ قانون بی‌فایده، قانون‌واحد یا قانون λ به چشم نمی‌خورد. اما چون $S \to aa$ یکی از گرامرهای متناظر است، گرامر فوق کمینه نمی‌باشد.

بخش ۶-۲

۵- ابتدا باید قوانین λ را حذف کنیم. به این ترتیب، داریم:

$$S \to AB \mid B \mid aB,$$
$$A \to aab,$$
$$B \to bbA \mid bb.$$

با اینکار قانون‌واحد معرفی می‌شود که بر اساس ساخت قضیه ۶-۶، قابل پذیرش نمی‌باشد. به راحتی می‌توان اقدام به حذف این قانون‌واحد نمود.

$$S \to AB \mid bbA \mid aB \mid bb,$$
$$A \to aab,$$
$$B \to bbA \mid bb.$$

سپس، با اعمال ساخت قضیه فوق گرامر زیر بدست می‌آید:

$$S \to AB \mid V_bV_bA \mid V_aB \mid V_bV_b,$$
$$A \to V_aV_bV_b,$$
$$B \to V_bV_bA \mid V_bV_b,$$

$$S \to AB \mid V_a A \mid V_a B \mid V_a V_b,$$
$$A \to V_a V_a,$$
$$B \to V_a A \mid V_b V_b,$$
$$V_a \to V_a V_b,$$
$$V_a \to V_a V_b,$$
$$V_a \to a,$$
$$V_b \to b.$$

۸- فرم کلی یکی از قوانین یکی از گرامرهای خطی را در نظر بگیرید.

$$A \to a_1 a_2 \ldots a_n B b_1 b_2 \ldots b_m.$$

متغیر جدید V_1 با قوانین

$$V_1 \to a_2 \ldots a_n B b_1 b_2 \ldots b_m$$

و

$$A \to a_1 V_1$$

را معرفی کنید.

این فرآیند را تکرار کرده و این دفعه متغیر V_2 و

$$V_2 \to a_3 \ldots a_n B b_1 b_2 \ldots b_m$$

را معرفی کرده و این کار را آنقدر ادامه دهید تا هیچ پایانه‌ای در طرف چپ باقی نماند. سپس به کمک فرآیندی مشابه، پایانه‌های واقع در طرف راست را حذف کنید.

۹- با استفاده از CNF ... ساختی می‌توان به این فرم نرمال دست پیدا کرد. به دلیل امکان‌پذیر بودن $a = \lambda$، قوانینی به فرم $A \to BC$ هم مجاز هستند. برای $A \to a$، متغیرهای جدید V_1, V_2 و قوانین $V_1 \to \lambda, V_2 \to \lambda$ $A \to aV_1V_2$ را ایجاد کنید.

۱۲- جواب‌ها: $S \to aV_b \mid aS \mid aV_aS, V_a \to a, V_b \to b$.

۱۵- فقط $A \to bABC$ به فرم مورد نظر نیست، بنابراین $A \to bAV$ و $V \to BC$ را معرفی می‌کنیم. مورد آخر به فرم صحیح نیست، اما پس از جایگذاری B داریم:

$$S \to aSA,$$
$$A \to bAV,$$
$$V \to bC,$$
$$B \to b,$$
$$C \to aBC.$$

بخش ۶-۳

۲- چون aab یکی از پیشوندهای رشته مثال ۶-۱۱ است، می‌توان از V_0 محاسبه شده در آن مثال استفاده کرد. بدلیل آنکه $S \in V_{13}$، رشته aab در زبان تولید شده بوسیله این گرامر قرار داشته و بنابراین، قابل تجزیه است.

برای تجزیه، قوانینی را تعیین می‌کنیم که در اثبات $S \in V_{13}$ مورد استفاده قرار گرفتند:

$S \in V_{13}$ زیرا برای قانون $S \to AB$ داریم $A \in V_{11}$ و $B \in V_{23}$

$A \to a$ زیرا $A \in V_{11}$

$B \in V_{23}$ زیرا برای قانون $B \to AB$ داریم $A \in V_{22}, B \in V_{33}$

$A \to a$ زیرا $A \in V_{22}$

$B \to b$ زیرا $B \in V_{33}$

تمام قوانین لازم برای تأیید عضویت را در اختیار داریم! سپس می‌توان از این قوانین برای تجزیه

$$S \Rightarrow AB \Rightarrow aB \Rightarrow aAB \Rightarrow aaB \Rightarrow aab$$

استفاده کرد.

فصل ۷

بخش ۷-۱

۲- کلید استدلال، پرش از q_0 به q_1 است، که به صورت نامعین انجام شده و نباید در میانه رشته انجام شود. با این وجود، اگر تبدیل در نقطه دیگری انجام شود یا اگر ورودی به فرم ww^R نباشد، نمی‌توان به یک پیکربندی پذیرش دست یافت. فرض کنید که محتوای رشته در زمان تبدیل به صورت $x_1x_2\ldots x_kz$ باشد. برای پذیرش یک رشته باید به پیکربندی (q_1, λ, z) برسیم. با بررسی تابع انتقال، مشاهده می‌کنیم که فقط در صورتی می‌توان به این پیکربندی دست یافت که در نقطه تبدیل، بخش خوانده نشده ورودی $x_1x_2\ldots x_k$ باشد، یعنی اگر ورودی اصلی به فرم ww^R بوده و تبدیل دقیقاً در میانه رشته ورودی رخ دهد.

۴- (الف) جواب با قرار گرفتن دو سمبل روی پشته برای هریک از a ها، بدست می‌آید؛ در حالیکه هریک از b ها از یک نشانه استفاده می‌کند. جواب:

$$\delta(q_0, \lambda, z) = \{(q_f, z)\},$$
$$\delta(q_0, a, z) = \{(q_1, 11z)\},$$
$$\delta(q_0, a, 1) = \{(q_1, 11)\},$$
$$\delta(q_1, b, 1) = \{(q_1, \lambda)\},$$
$$\delta(q_1, \lambda, z) = \{(q_f, z)\}.$$

(و) در اینجا با استفاده از ویژگی نامعین بودن، یک، دو یا سه نشانه با ضابطه

$$\delta(q_0,a,z)=\{(q_1,1z),(q_1,11z),(q_1,111z)\}$$

و

$$\delta(q_0,a,1)=\{(q_1,11),(q_1,111),(q_1,1111)\}$$

ایجاد می‌کنیم.

مابقی راه‌حل هم اساساً مشابه ۴ (الف) است.

۹- این pda به هیچ وجه از پشته استفاده نمی‌کند و بنابراین، عملاً تبدیل به یک پذیرنده متناهی می‌شود. از اینرو، می‌توان انتقالات حالت را مستقیماً از pda گرفته و روابط زیر را بدست آورد.

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

۱۱- فرآیند را ادامه داده و هربار یکی از مسیرها را در نظر بگیرید. انتقال از q_0 به q_2 بوسیله یک a قابل انجام است. در مسیر بعدی باید به دنبال یک a بگردیم که پس از آن یک یا چند b قرار گرفته و به یک a ختم می‌شود. اینها تنها انتخاب‌ها هستند. بنابراین pda زبان

$$L=\{a\}\cup L(abb^*a)$$

را می‌پذیرد.

۱۴- در این مورد، حالت‌های کافی برای بررسی انتقال از a به b و بالعکس وجود ندارد. برای رفع این نقیصه، با قرار دادن یک سمبل در پشته، محل استقرار خود در دنباله را یادآوری می‌کنیم. بعنوان مثال، یکی از جواب‌ها

$$\delta(q_0,a,z)=\{(q_0,1)\},$$
$$\delta(q_0,a,1)=\{(q_0,1)\},$$
$$\delta(q_0,b,1)=\{(q_0,2)\},$$
$$\delta(q_0,a,2)=\{(q_0,2)\},$$
$$\delta(q_0,\lambda,2)=\{(q_f,2)\}$$

می‌باشد. فقط دو حالت در اختیار داریم که یکی حالت شروع q_0 و دیگری حالت پذیرش q_f است. آنچه معمولاً بوسیله حالت‌های مختلف انجام می‌شد، در اینجا بوسیله سمبل داخل پشته انجام می‌شود.

۱۶- در این مسأله از حالت‌های درونی برای یادآوری سمبلی که باید روی پشته قرار گیرند، استفاده می‌شود. بعنوان مثال،

$$\delta(q_i,a,b)=\{(q_j,cde)\}$$

با

$$\delta(q_i,a,b)=\{(q_{jc},de)\}$$
$$\delta(q_{jc},\lambda,d)=\{(q_j,cd)\}$$

جایگذاری می‌شود. بدلیل آنکه δ فقط تعداد متناهی عضو داشته و هریک فقط مقدار متناهی اطلاعات را به پشته اضافه می‌کنند، از این ساختار می‌توان در تمامی pda ها استفاده کرد.

بخش ۷-۲

۳- می‌توانید از ساخت قضیه ۷-۱ یا از زبان مربوطه $\{a^{n+2}b^{2n+1}:n\geq 0\}$ استفاده کنید. در صورت بکارگیری زبان اخیر، جواب زیر بدست می‌آید.

$$\delta(q_0,a,z)=\{(q_1,z)\},$$
$$\delta(q_1,a,z)=\{(q_2,z)\},$$
$$\delta(q_2,a,z)=\{(q_2,1z)\},$$
$$\delta(q_2,a,1)=\{(q_2,11)\},$$
$$\delta(q_2,b,1)=\{(q_3,1)\},$$
$$\delta(q_3,b,1)=\{(q_3,\lambda)\},$$
$$\delta(q_3,\lambda,z)=\{(q_f,z)\},$$

که در آن، q_0 حالت شروع و q_f حالت پایانی است.

۴- ابتدا با تبدیل گرامر به فرم نرمال گریباخ، قوانین $S\to aSSS;S\to aB;B\to b$ را بدست آورید. سپس ساختار قضیه ۷-۱ را اجرا کنید.

$$\delta(q_0,\lambda,z)=\{(q_1,Sz)\},$$
$$\delta(q_1,a,S)=\{(q_1,SSS),(q_1,B)\},$$
$$\delta(q_1,b,B)=\{(q_1,\lambda)\},$$
$$\delta(q_1,\lambda,z)=\{(q_f,z)\}.$$

۷- با استفاده از قضیه ۷-۲ و با معلوم بودن یک npda دلخواه، می‌توان یک گرامر مستقل‌ازمتن متناظر ایجاد نمود. از این گرامر می‌توان با استفاده از قضیه ۷-۱، یک npda سه حالتی ایجاد نمود. به دلیل خاصیت جابجایی تناظر، npda های اولیه و ثانوی هم‌ارز هستند.

۹- ابتدا یک گرامر، مثل $S \to aSB \mid b, B \to b$، به فرم نرمال گریباخ برای L بدست می‌آوریم. سپس، با استفاده از ساخت قضیه ۷-۱، یک npda با سه حالت q_0, q_1, q_f ایجاد می‌کنیم. در صورتی می‌توان اقدام به حذف حالت q_1 کرد که از یک سمبل ویژه پشته مانند z_1 برای نشانه‌گذاری آن استفاده شود. جواب کامل به صورت زیر خواهد بود:

$$\delta(q_0, \lambda, z) = \{(q_0, Sz_1)\},$$
$$\delta(q_0, a, S) = \{(q_0, SB)\},$$
$$\delta(q_0, b, S) = \{(q_0, \lambda)\},$$
$$\delta(q_0, b, B) = \{(q_0, \lambda)\},$$
$$\delta(q_0, \lambda, z_1) = \{(q_f, \lambda)\}.$$

۱۱- برای شروع، باید حداقل یک a وجود داشته باشد. سپس، $\delta(q_0, a, \Lambda) = \{(q_0, \Lambda)\}$، بدون تغییر پشته اقدام به خواندن a ها می‌کند. در پایان، pda به محض مواجهه با اولین b، به حالت q_1 رفته و از این حالت فقط می‌تواند یک انتقال λ به حالت پایانی انجام دهد. بنابراین، یک رشته مفروض پذیرفته می‌شود اگر و تنها اگر شامل یک یا چند a بوده و پس از آن فقط یک b قرار داشته باشد.

بخش ۷-۳

۴- در اولین نگاه، این زبان ممکن است نامعین به نظر برسد، چون وجود پیشوند a مستلزم وجود دو نوع پسوند متفاوت است. با این وجود، به دلیل امکان ساخت یک dpda، زبان مذکور معین است. در صورتیکه اولین سمبل ورودی یک a باشد، این dpda به یکی از حالات پایانی خواهد رفت. در صورت وجود سمبل‌های بعدی، از این حالت خارج شده و سپس $a^n b^n$ را می‌پذیرد. جواب کامل:

$$\delta(q_0, a, z) = \{(q_3, 1z)\},$$
$$\delta(q_3, a, 1) = \{(q_1, 11)\},$$
$$\delta(q_1, a, 1) = \{(q_1, 11)\},$$
$$\delta(q_1, b, 1) = \{(q_1, \lambda)\},$$
$$\delta(q_1, \lambda, z) = \{(q_2, z)\},$$

که در آن، $F = \{q_2, q_3\}$ می‌باشد.

۹- جواب به راحتی بدست می‌آید. کافی است فقط a ها و b ها را روی پشته قرار دهید. وجود c بیانگر تغییر وضعیت از حالت ذخیره به تطابق است. به این ترتیب، می‌توان همه چیز را به صورت معین انجام داد.

۱۱- دو حالت وجود دارد؛ یک حالت شروع غیرپذیرش q_0 و یک حالت پایانی q_1. مادامیکه یک z در

بالای پشته قرار نداشته باشد، pda در حالت q_1 قرار خواهد داشت. اما در صورت وجود z در بالای پشته، pda حالت‌ها را به q_0 تغییر خواهد داد. مابقی کار اساساً مشابه مثال ۷-۳ است. بنابراین داریم: $\delta(q_1, a, 0) = \{(q_1, 00)\}$، $\delta(q_0, a, z) = \{(q_1, 0z)\}$ و الی آخر، که در آن $\delta(q_1, \lambda, z) = \{(q_0, z)\}$. حال به راحتی مشاهده می‌کنید که pda مذکور معین است.

۱۵- این امر بدیهی است، چون هر زبان منظمی را می‌توان با یک dfa پذیرفت و چنین dfa یک dpda با پشته استفاده نشده است.

۱۶- در این مسأله باید مطابق آنچه در شرح کلی قضیه ۴-۱ گفتیم، اقدام به ترکیب یک dpda با dfa نمود و پشته هم مطابق L_1 مدیریت می‌شود. به راحتی می‌توان مشاهده کرد که در نتیجه، یک dpda بوجود می‌آید.

بخش ۷-۴

۲- رشته‌های $aabb$ و $aabbbbaa$ را در نظر بگیرید. در حالت اول، اشتقاق باید با $S \Rightarrow aSb$ و در دومی با $S \Rightarrow SS$ آغاز شود. اما اگر فقط به چهار سمبل اول نگاه کنیم، نمی‌توان در مورد انتخاب مناسب از بین یعنی دو انتخاب تصمیم‌گیری کرد. بنابراین گرامر در $LL(4)$ وجود ندارد. بدلیل آنکه می‌توان برای رشته‌های با طول دلخواه مثال‌های مشابهی ارائه کرد، این گرامر به ازای هر k در $LL(k)$ وجود ندارد.

۴- به سه سمبل اول نگاه کنید. اگر aaa ، aab یا aba باشند، آنگاه رشته فقط در $L(a^*ba)$ وجود دارد. اما اگر سه سمبل اول abb باشد، آنگاه هر رشته قابل تجزیه‌ای باید در $(abbb^*)$ وجود داشته باشد. در هر حالت، می‌توان یک LL گرامر یافت و آندو را با هم ترکیب کرد. یکی از جواب‌ها:

$$S \to S_1 \mid S_2,$$
$$S_1 \to aS_1 \mid ba,$$
$$S_2 \to abbB,$$
$$B \to bB \mid \lambda,$$

می‌باشد. با نگاهی به سه سمبل متوجه می‌شویم که باید از $S \Rightarrow S_1$ استفاده کنیم یا از $S \Rightarrow S_2$. بنابراین گرامر $LL(3)$ می‌باشد.

۷- به ازای هر CFL معین یک dpda وجود دارد. چنانچه این dpda به گرامر تبدیل شود، گرامر به‌دست‌آمده غیرمبهم خواهد بود.

۹- (الف)

$$S \to aSc \mid S_1 \mid \lambda,$$
$$S_1 \to bS_1c \mid \lambda.$$

گرامر بالا تاحدودی به فرم s – گرامر است. تازمانیکه سمبل جاری خوانده شده a باشد، باید $S \to aSc$ را اعمال نمود؛ اما اگر سمبل مذکور b باشد، ما هم باید از $S \to S_1$ استفاده کنیم؛ اگر هم c باشد، فقط می‌توان از $S \to \lambda$ استفاده کرد. گرامر $LL(1)$ می‌باشد.

فصل ۸

بخش ۸-۱

۳- فرض کنید $w = a^m b^m b^m a^m a^m b^m$ باشد. حال حریف چندین گزینه پیش رو دارد. اگر، بعنوان مثال، $v = a^k$ و $y = a^l$، با ضابطه واقع بودن x و y در پیشوند a^m، آنگاه

$$w_0 = a^{m-k-l} b^m b^m a^m a^m b^m,$$

که در L وجود ندارد. گرچه انتخاب‌های دیگری هم وجود دارند، اما در همه آنها می‌توان رشته را از زبان پمپاژ کرد.

۷- (الف) از لم تزریق استفاده کنید. با معلوم بودن m، $w = a^{m^2} b^{m^2}$را انتخاب کنید. در اینصورت، تنها انتخاب‌های مهم v و y، $v = a^k$ و $y = b^l$ است که در آن، k و l غیرصفر می‌باشند. فرض کنید که $l = 1$ باشد. سپس $i = 2$ را انتخاب کنید تا تعداد a های موجود در w_2 به $m^2 + k$ و تعداد b های آن به $m+1$ برسد. اما

$$(m+1)^2 = m^2 + 2m + 1$$
$$> m^2 + k,$$

بنابراین، w_2 در این زبان قرار ندارد. در مورد $l > 1$ هم می‌توان استدلال‌های مشابهی را مطرح کرد. بنابراین، زبان مذکور مستقل از متن نمی‌باشد.

(و) با داشتن m، $w = a^{m^2} b^{m^2+1} c^{m^2+2}$ را انتخاب کنید که به راحتی از زبان پمپاژ می‌شود.

۸- (ب) این زبان مستقل‌ازمتن نیست. با استفاده از لم تزریق و ضابطه $w = a^m b^m a^m b^m$، انتخاب‌های مختلف x و y را در نظر بگیرید.

شاید تعجب داشته‌باشد که این زبان مستقل‌ازمتن است. ابتدا برای w_1، k تا سمبل مربوط به آن را بر روی پشته قرار می‌دهیم و سپس با رسیدن به c در ورودی، عمل تطابق سمبل ورودی با سمبل روی پشته صورت می‌گیرد اگر حتی یک مورد عدم تطابق رخ دهد رشته را پذیرش می‌کنیم.

۱۲- از لم تزریق مربوط به زبان‌های خطی استفاده کنید. با در اختیار داشتن یک m دلخواه، $w = a^m b^{2m} a^m$ را در نظر بگیرید. در اینصورت، در v و y فقط a به چشم می‌خورد و بنابراین می‌توان w را به راحتی از زبان مذکور پمپاژ کرد.

۱۵- این زبان خطی نیست. از لم تزریق و

$$w = (\ldots(a)\ldots) + (\ldots(a)\ldots)$$

استفاده کنید که در آن،)...(و (...(به ترتیب، به معنای m پرانتز چپ و راست است. اگر $|u| \geq 1$، به راحتی می‌توان پمپاژ کرد تا به ازای یک پیشوند مفروض v، به $n_1(v) < n_2(v)$ دست یافت که باعث ایجاد یک عبارت نامناسب می‌شود. برای صورت‌های تجزیه شده دیگر هم می‌توان از استدلال‌های مشابه استفاده کرد.

۲۰- از $w = a^{m^2}$ استفاده کنید که در آن، p و q اعداد اولی هستند که $p > m$ و $q > m$ می‌باشد. اگر $|vy| = k$، آنگاه

$$|w_{i+1}| = pq + ik.$$

اگر $i = pq$ را انتخاب کنیم، آنگاه

$$w_{i+1} = a^{pq(1+k)},$$

که در این زبان وجود ندارد.

بخش ۸-۲

۱- مکمل مذکور مستقل‌ازمتن بوده و از دو حالت تشکیل می‌شود: $n_a(w) \neq n_b(w)$ و $n_a(w) \neq n_c(w)$. هر یک از اینها را می‌توان متقابلاً به $n_a(w) < n_c(w)$ و $n_a(w) > n_c(w), n_a(w) < n_b(w)$ تقسیم کرد. همچنانکه می‌توان از ساخت CFG هم دریافت، تمامی حالت‌های فوق مستقل‌ازمتن هستند. بنابراین زبان کامل از اجتماع این چهار حالت بوجود آمده و با توجه به بسته‌بودن تحت اجتماع مستقل‌ازمتن است.

۵- با معلوم بودن گرامر مستقل‌ازمتن G و جایگزینی هر یک از قوانین $A \to x$ با $A \to x^R$، گرامر مستقل از متن دلخواه $\hat{G}$ را ایجاد کنید. سپس می‌توان با استقراء روی تعداد مراحل یکی از اشتقاق‌ها نشان داد که اگر w یکی از فرم‌های جمله‌ای G باشد، آنگاه w^R یکی از فرم‌های جمله‌ای $\hat{G}$ خواهد بود.

۹- با معلوم بودن دو گرامر خطی $G_1 = (V_1, T, S_1, P_1)$ و $G_2 = (V_2, T, S_2, P_2)$ با ضابطه $V_1 \cap V_2 = \emptyset$، گرامر مرکب $\hat{G} = (V_1 \cup V_2, T, S, P_1 \cup P_2 \cup S \to S_1 | S_2)$ را ایجاد کنید. پس $\hat{G}$ خطی است و $L(\hat{G}) = L(G_1) \cup L(G_2)$ می‌باشد.

برای اثبات اینکه زبان‌های خطی تحت الحاق بسته نیستند، زبان خطی $L = \{a^n b^n : n \geq 1\}$ را در نظر بگیرید. با اعمال لم تزریق می‌توان نشان داد که زبان L^2 خطی نیست.

۱۳- فرض کنید $G_1 = (V_1, T, S_1, P_1)$ یکی از گرامرهای خطی L_1 و $G_2 = (V_2, T, S_2, P_2)$ یکی از گرامرهای خطی‌چپ L_2 باشد. با جایگزینی هر یک از قوانین به شکل $V \to x, x \in T^*$ با $V \to S_1 x$، گرامر $\hat{G}_2$ را از G_2 بدست آورید. ضمن انتخاب S_2 بعنوان یک سمبل شروع، گرامرهای G_1 و $\hat{G}_2$ را با هم ترکیب کنید. سپس می‌توان نشان داد که در این گرامر،

$$S_2 \Rightarrow S_1 w \Rightarrow uw$$

اگر و فقط اگر $u \in L_1$ و $w \in L_2$ باشد.

۱۵- زبان‌های $L_1 = \{a^n b^n c^m\}$ و $L_2 = \{a^n b^m c^m\}$ نامبهم هستند. اما اشتراک آنها به هیچ وجه مستقل از متن نمی‌باشد.

۲۱- $\lambda \in L(G)$ اگر و تنها اگر S یک متغیر میرا باشد.

بخش ۹-۱

۲- یکی از راه‌حل‌های سه‌حالتی که کل ورودی را بررسی می‌کند به صورت

$$\delta(q_0, a) = (q_1, a, R),$$
$$\delta(q_1, a) = \delta(q_1, b) = (q_1, a, R),$$
$$\delta(q_1, \square) = (q_2, \square, R),$$

با ضابطه $F = \{q_2\}$ می‌باشد.

همچنین می‌توان فقط با بررسی اولین سمبل و نادیده‌گرفتن مابقی ورودی به پاسخ مسأله رسید، بعنوان مثال

$$\delta(q_0, a) = (q_2, a, R).$$

توجه داشته باشید که در یک ماشین تورینگ، لازم نیست قبل از پذیرش ورودی، کل آن را مورد بررسی قرار داد.

۷- (الف)

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

با ضابطه $F = \{q_3\}$.

(ب)

$$\delta(q_0, a) = \delta(q_0, b) = (q_1, \square, R),$$
$$\delta(q_0, \square) = (q_2, \square, R),$$
$$\delta(q_1, a) = \delta(q_1, b) = (q_0, \square, R),$$

با ضابطه $F = \{q_2\}$.

۱۰- جواب از نظر مفهومی ساده است، ولی جزئیات طولانی و خسته‌کننده‌ای دارد. طرح کلی جواب

تا حدودی به این صورت است:

(الف) در هر یک از دو سر رشته، یک سمبل نشانه c قرار دهید.

(ب) ترکیب دو سمبلی ca واقع در سمت چپ را با ac و ترکیب دو سمبلی ac واقع در سمت راست را با ca جایگزین کنید. این کار را آنقدر تکرار کنید تا دو c در میان رشته با هم تلاقی کنند.

(ج) یکی از c ها را حذف کرده و مابقی رشته را حذف کنید تا خلأ پر شود.

اینکار یکی از روش‌های طولانی و خسته‌کننده‌ای است که معمولاً در ماشین تورینگ برای انجام کارهای بسیار ساده استفاده می‌شود.

۱۲- چون دقیقاً نمی‌دانیم که در کجا باید متوقف شویم، نمی‌توان فقط از یک جهت شروع به جستجو کرد. باید با حرکات عقب و جلو، نشانه‌ها را در حدهای چپ و راست منطقه جستجوشده، قرار داد و نشانه‌ها را به خارج انتقال داد.

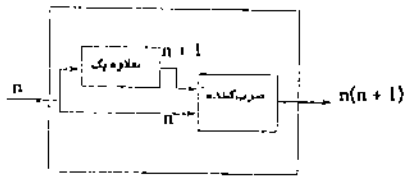
۱۹- اگر مجموعه حالات پایانی F بیش از یک عضو داشته باشد، حالت جدید پایانی q_f و انتقالات

$$\delta(q, a) = (q_f, a, R)$$

را برای تمام $q \in F$ و $a \in \Gamma$ معرفی کنید.

بخش ۹-۲

۳- (الف) می‌توان ماشین تورینگ را شامل دو بخش اصلی در نظر گرفت: یک ماشین به نام بعلاوه یک که فقط ورودی را با یک جمع می‌زند و یک ضرب‌کننده که دو عدد را در هم ضرب می‌کند. از نظر کلی، این دو بخش به شکل ساده‌ای با هم ترکیب می‌شوند.



۵- (ج) ابتدا، ورودی را به بخش‌های مساوی تقسیم کنید. برای اینکار از تمرین ۱۰ بخش ۹-۱ استفاده کنید. سپس، هریک از سمبل‌های واقع در یک بخش را با سمبل نظیر آن در بخش دیگر مقایسه کنید تا به یک ناهمخوانی برسید.

۸- یکی از جواب‌ها:

$$\delta(q_0, a) = (q_i, a, R),$$
$$\delta(q_0, c) = (q_0, c, R) \quad ،c \in \Sigma - \{a\} \text{ به ازای تمام}$$
$$\delta(q_0, \square) = (q_f, \square, R).$$

حالت q_0 هر حالتی است که در آن، دستورالعمل *searchright* اعمال شده باشد.

بخش ۹-۳

۲- این واقعیت را نادیده گرفته‌ایم که ماشین‌های تورینگ، طبق آنچه تاکنون تعریف کردیم، معین هستند؛ در حالی که pda می‌تواند نامعین هم باشد. بنابراین، هنوز هم نمی‌توان ادعا کرد که ماشین‌های تورینگ قدرتمندتر از اتوماتای پشته‌ای هستند.

فصل ۱۰

بخش ۱۰-۱

۴- (الف) ماشین دارای یک تابع انتقال

$$\delta: Q\times\Gamma \to Q\times\Gamma\times\{L,R,S\}$$

با این محدودیت است که به ازای تمامی انتقالات (R یا L) $\delta(q_i,a)=(q_j,b,L)$، شرط $a=b$ باید صدق کند.

(ب) برای شبیه‌سازی $\delta(q_i,a)=(q_j,b,L)$ با ضابطه $a\neq b$ از ماشین استاندارد، انتقالات جدید $\delta(q_i,a)=(q_{jL},b,S)$ و $\delta(q_{jL},b)=(q_j,b,L)$ را به ازای تمام $c\in\Gamma$ تعریف می‌کنیم و الی آخر.

۶- شبه‌خالی B را معرفی می‌کنیم. هرگاه ماشین اصلی در پی نوشتن □ باشد، ماشین جدید B را می‌نویسد. آنگاه، به ازای هر $\delta(q_i,\square)=(q_j,b,L)$، $\delta(q_i,B)=(q_j,b,L)$ را اضافه کرده و الی آخر. البته، انتقال اصلی $\delta(q_i,\square)=(q_j,b,L)$ را باید حفظ کرد تا بتوان خالی‌هایی را که در ابتدا روی نوار قرار دارند، مدیریت کرد.

۹- این کار باعث کاهش قدرت ماشین نمی‌شود. به ازای هر سمبل $a\in\Gamma$، شبه‌سمبلی مثل A را معرفی می‌کنیم. اگر بخواهیم این a را نگه داریم، ابتدا A را نوشته و سپس، به سراغ سلول مورد نظر رفته و A را با a جایگزین می‌کنیم.

۱۱-

$$\delta(q_i,\{a,b\})=(q_j,c,R)$$

را با

$$\delta(q_i,d)=(q_j,c,R)$$

به ازای تمام $d\in\Gamma-\{a,b\}$ جایگزین می‌کنیم.

بخش ۱۰-۲

۲- برای ارائه تعریف صوری، از $\Gamma_2=\Gamma\times\Gamma\times\ldots\times\Gamma$ و $\delta: Q\times\Gamma_2\to Q\times\Gamma_2\times\{L,R\}^n$ استفاده

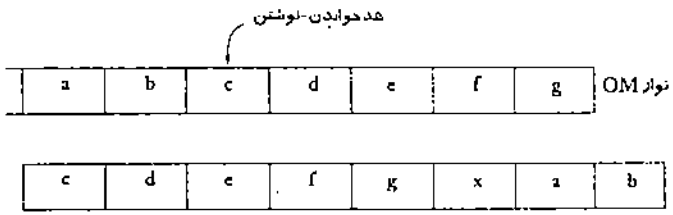
کنید که در آن، m تعداد هدهای خواندن- نوشتن است. نکته مهم این است که بدانیم وقتی دو هد خواندن- نوشتن روی کوچکترین سلول واحد قرار می‌گیرند، چه اتفاقی می‌افتد. در تعریف رسمی باید راه‌حلی برای مشکلات احتمالی ارائه شود.

برای شبیه‌سازی ماشین اصلی (OM) بوسیله یک ماشین تورینگ استاندارد (SM)، $m+1$ شیار برای SM در نظر می‌گیریم. در یک شیار، محتویات نوار OM را نگه می‌داریم، در حالیکه از m شیار بعدی جهت نمایش موقعیت هدهای نوار OM استفاده می‌کنیم.

محتوای نوار OM	□	a	b	c	d	□
موقعیت هد نوار # 1	□		x			□
موقعیت هد نوار # 2	□				x	□

SM با اسکن‌کردن و بروزرسانی منطقه فعال خود اقدام به شبیه‌سازی هریک از حرکات OM می‌کند.

۵- این تمرین نشان می‌دهد که یک ماشین صفی متناظر با یک ماشین تورینگ استاندارد بوده و بنابراین، صف ابزار ذخیره قدرتمندتری نسبت به پشته است. بعنوان مثال، برای شبیه‌سازی یک TM استاندارد بوسیله یک ماشین صفی می‌توان سمت راست OM را در جلوی صف و سمت چپ را در عقب آن نگه داشت.



برای حرکت به سمت راست فقط لازم است که سمبل جلویی صف را حذف کرده و چیزی را در پشت قرار داد. با این وجود، چون حرکت به چپ به سختی انجام می‌شود، باید محتویات صف را چندین مرتبه گردآوری کرد تا همه چیز در جای مناسب خود قرار گیرد. به این ترتیب، می‌توان از نشانه‌های اضافی Y و Z برای اشاره به حدها استفاده کرد. بعنوان مثال، جهت شبیه‌سازی

$$\delta(q_i,c)=(q_j,z,L)$$

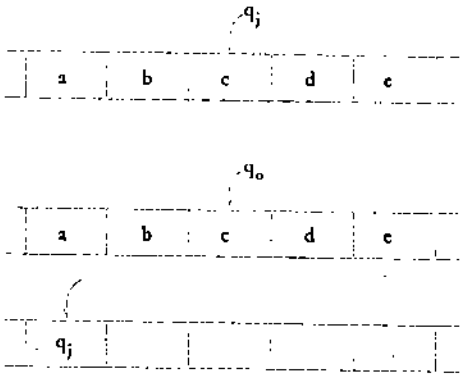
مراحل زیر را انجام می‌دهیم.

(الف) c را از جلو حذف کرده و zY را به عقب اضافه کنید.

(ب) محتویات را دوران دهید تا به $bzYdefgxa$ برسید.

(ج) Z را به عقب اضافه کنید، سپس دوران داده و Y و Z را در حین انتقال به جلو حذف کنید.

۹- فقط به دو نوار نیاز داریم، یکی بعنوان قرینه نوار OM و دیگری برای ذخیره‌سازی حالت OM.



SM فقط به دو حالت نیاز دارد: یک حالت پذیرش و یک حالت غیرپذیرش.

بخش ۱۰-۳

۳- (الف) از سمت چپ ورودی آغاز کنید. با قرار دادن ماشین در حالت مناسب، سمبل را یادآوری کنید. سپس آنرا با X جایگذاری کنید.

(ب) هد خواندن- نوشتن را به سمت راست جابجا کرده و (به صورت نامعین) در مرکز ورودی متوقف شوید.

(ج) سمبل این محل را با سمبل یادآوری شده مقایسه کنید. در صورت همخوانی، Y را در سلول بنویسید. در غیر اینصورت، ورودی را رد کنید.

(د) حال که مرکز ورودی را با Y نشانه‌گذاری کردید، می‌توانید به صورت نامعین به پیش بروید و ضمن حرکت‌های متوالی به راست و چپ، سمبل‌ها را با هم مقایسه کنید.

برای بدست آوردن جواب کاملاً معین، ابتدا مرکز ورودی را پیدا می‌کنیم (به این منظور، مثلاً نشانه‌ها در هر یک از دو سر ورودی قرار داده شده و آنقدر آنها را به داخل انتقال می‌دهیم تا با هم تلاقی پیدا کنند).

۶- یک مقدار برای n در نظر بگیرید. برای اینکار، ضمن توقف به صورت نامعین در یک n مفروض، $1,2,\ldots$ را ایجاد کنید. تعیین کنید آیا طول ورودی مضربی از n هست یا خیر. در صورت مثبت بودن جواب، ورودی را بپذیرید. اما اگر $a^n \in L$ باشد، آنگاه یک n با این شرط وجود دارد.

۷- یکی از پشته‌ها محتویات نوار واقع در سمت راست یک نقطه مرجع مفروض و پشته دیگر نوار، محتویات نوار واقع در سمت چپ را نگه خواهد داشت. سپس می‌توان با بیرون کشیدن و هل دادن پشته، به سمت راست و چپ حرکت کرد.

بخش ۱۰-۴

۳- شکل کلی الگوریتم به صورت زیر است.

الف) یک کپی از رشته قبل ایجاد کنید.

ب) راست‌ترین 0 را پیدا کرده و آنرا به یک 1 تغییر دهید. سپس تمامی 1های واقع در سمت راست آنرا به 0 تغییر دهید.

ج) اگر هیچ 0 وجود نداشته باشد، تمام 1 ها را به 0 تغییر داده و یک 1 در سمت چپ اضافه کنید.

د) مجدداً به مرحله (الف) بازگردید.

۸- فرض کنید $S_1 = \{s_1, s_2, \ldots\}$ و $S_2 = \{t_1, t_2, \ldots\}$ باشد. آنگاه، برای شمارش اجتماع آنها می‌توان از

$$S_1 \cup S_2 = \{s_1, t_1, s_2, t_2, \ldots\}$$

استفاده کرد. هر $s_i = t_j$ مفروض را فقط یک مرتبه در لیست قرار می‌دهیم. بنابراین، اجتماع دو مجموعه شمارش‌پذیر است. به ازای $S_1 \times S_2$، از مرتب‌سازی شکل ۱۰-۱۷ استفاده کنید.

بخش ۱۰-۵

۲- ابتدا، ورودی را به دو تقسیم کرده و نتیجه را به یکی از بخش‌های نوار، انتقال دهید. سپس می‌توان از فضای آزاد ایجاد شده، که در ابتدا در تصرف ورودی قرار داشت، جهت ذخیره‌سازی تقسیم‌کننده‌های متوالی استفاده نمود.

۴- (a) از یک ماشین سه شیاری مطابق شکل زیر استفاده کنید. درشیار سوم، مقدار آزمایشی جاری $|w|$ را نگه‌داری می‌کنیم. روی شیار دوم، تقسیم‌کننده‌ها را در هر $|w|$ سلول قرار می‌دهیم. سپس محتویات سلول‌های واقع در بین نشانه‌ها را با هم مقایسه می‌کنیم.

ورودی	a	b	c	d	b	c	d
تقسیم‌کننده‌ها		x				x	
مقدار آزمایشی $\|w\|$	1	1	1				

۶- با استفاده از تمرین ۱۵ بخش ۶-۲ یک گرامر به فرم دوم استاندارد پیدا کنید. سپس نحوهٔ ساخت موجود قضیه ۷-۱ را اعمال کنید. pda حاصل از این کار در هر حرکت خود یک سمبل ورودی را مصرف کرده و در هر مرتبه هیچ یک از محتویات پشته را بیش از یک سمبل افزایش نخواهد داد.

فصل ۱۱

بخش ۱۱-۱

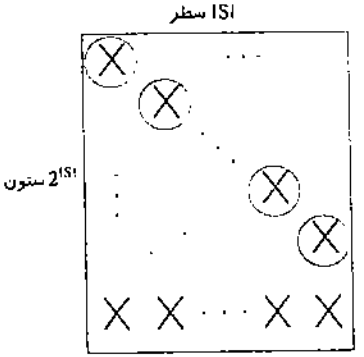
۲- می‌دانیم که اجتماع دو مجموعه شمارش‌پذیر، شمارش‌پذیر بوده و بعلاوه، مجموعه تمام زبان‌های شمارش‌پذیربازگشتی، شمارش‌پذیر است. اگر مجموعه تمام زبانهایی که شمارش‌پذیربازگشتی نیستند هم شمارش‌پذیر باشد، آنگاه مجموعه تمام زبان‌ها شمارش‌پذیر می‌باشد. اما، همچنان که می‌دانیم، در اینجا اینچنین نیست.

۶- فرض کنید که L_1 و L_2 دو زبان شمارش‌پذیر بوده و M_1 و M_2 ماشین‌های تورینگ مربوطه‌ای هستند که این‌دو زبان را می‌پذیرند. وقتی با ورودی مثل w مواجه می‌شویم، به صورت نامعین M_1 یا M_2 را برای پردازش w انتخاب می‌کنیم. ماشین تورینگ حاصله $L_1 \cup L_2$ را می‌پذیرد.

۱۱- هر زبان مستقل‌ازمتن، بازگشتی است. همچنین، بنابراین براساس قضیه ۱۱-۴ مکمل آنهم بازگشتی است. با این وجود، توجه داشته باشید که مکمل لزوماً مستقل‌ازمتن نمی‌باشد.

۱۴- به ازای هر $w \in L^*$ مفروض، هر یک از بخش‌های $w = w_1 w_2 \ldots w_m$ را در نظر بگیرید. به ازای هر بخش، تعیین کنید که آیا $w_i \in L$ هست یا خیر. بدلیل آنکه به ازای هر w، فقط تعداد متناهی از این بخش‌ها وجود دارد، می‌توان تصمیم گرفت که آیا $w \in L^*$ هست یا خیر.

۱۸- جدول شکل ۱۱-۲، به دلیل دارا بودن $|2^S|$ سطر و $|S|$ ستون، مربعی شکل نیست. از اینرو، تلاش برای استدلال شمارش‌پذیربودن 2^S به ازای تعداد از متناهی S ها با استفاده از روش قطری شدن با شکست مواجه خواهد شد.



محل شکل ($|2^S|$ سطر و $|S|$ ستون)

حاصل قطری شدن همواره روی قطر یکی از سطرهای پایینی قرار دارد.

بخش ۱۱-۲

۱- به نمونه‌ای از مشتق‌ها توجه کنید.

$$S \Rightarrow aS_1bB \Rightarrow aaS_1bbB \Rightarrow a^nS_1b^nB \Rightarrow a^{n+1}b^{n-1}B \Rightarrow a^{n+1}b^{n+1}B \Rightarrow \ldots$$

بر این اساس به راحتی می‌توان ادعا کرد که گرامر فوق قادر به مشتق‌سازی

$$L = \{a^{n+1}b^{n+k}, n \geq 1, k = -1,1,3,\ldots\}$$

می‌باشد.

۳- با بیان صوری، این گرامر را می‌توان بوسیله $G = (V, T, S, P)$ توصیف کرد که در آن، $S \subseteq (V, T)^*$ و

$$L(G) = \{x \in T^* : s \Rightarrow_G \text{ به ازای هر } s \in S\}.$$

چون همواره می‌توان به هر گرامر نامحدود مفروض، قوانین شروع $S_0 \rightarrow s_i$ را به ازای تمامی $s_i \in S$ اضافه کرد، گرامرهای نامحدود تعریف ۱۱-۳ متناظر با این فرم توسعه‌یافته می‌باشند.

۷- برای بدست آوردن این فرم از گرامرهای بدون‌محدودیت، هر گاه که $|v| > |u|$، متغیرهای ساختگی را در سمت راست قرار دهید. بعنوان مثال،

$$AB \rightarrow C$$

را با

$$AB \rightarrow CD,$$
$$D \rightarrow \lambda$$

جایگزین می‌کنیم. استدلال تناظر هم به راحتی قابل انجام است.

بخش ۱۱-۳

۱- (ج) کار با گرامرهای حساس‌به‌متن همواره چندان هم آسان نیست. استفاده از ایده پیغام‌رسان که در مثال ۱۱-۲ ارائه شد، در اغلب موارد مفید است. در این مسأله، اولین مرحله ایجاد فرم جمله‌ای a^nBc^nD است. متغیرهای B و D بعنوان نشانه‌گذار و پیغام‌رسان عمل می‌کنند تا مطمئن شویم که b و d ها به تعداد مناسب و در محل‌های مناسب ایجاد شده‌اند. تحقیق در مورد تناسب تعداد به کمک حاصل‌های

$$S \rightarrow aAcD \mid aBcD,$$
$$A \rightarrow aAc \mid aBc$$

به راحتی قابل انجام است.

در مرحله بعد، B به کمک حاصل‌های

$$Bc \rightarrow cB,$$
$$Bb \rightarrow bB$$

به سمت راست حرکت کرده و به سراغ D می‌رود.
پس از آن، می‌توان با ایجاد یک d و یک پیغام‌رسان بازگشتی، b را در محل مناسب قرار داده و در همانجا متوقف شد.

$$BD \to Ed,$$
$$cE \to Ec,$$
$$bE \to Eb,$$
$$aE \to ab.$$

در روش دیگر، می‌توان به کمک یک پیغام‌رسان دیگر که یک b ایجاد می‌کند، اما مانع از پیشرفت فرآیند می‌شود، یک d و یک نشانه‌گذار D ایجاد نمود:

$$BD \to FDd,$$
$$cF \to Fc,$$
$$bF \to Fb,$$
$$aF \to abB.$$

۴- ساده‌ترین استدلال استفاده از یک lba است. یک زبان مستقل‌ازمتن را در نظر بگیرید. آنگاه یک lba M وجود دارد که آنرا می‌پذیرد. با معلوم بودن w، ابتدا آنرا به صورت w^R بازنویسی کرده و سپس، M را در آن اعمال می‌کنیم. بدلیل آنکه $L^R = \{w : w^R \in L\}$، w^R در M پذیرفته می‌شود اگر و تنها اگر $w^R \in L$. ماشینی که یک رشته را معکوس کرده و M را اعمال می‌کند، اصطلاحاً lba نامیده می‌شود. بنابراین، L^R حساس‌به‌متن است.

۶- برای بحث از یک lba استفاده می‌کنیم. مشخص است که یک lba وجود دارد که هر رشته‌ای به فرم wuw^R را می‌پذیرد. فقط لازم است که با شروع از دو انتهای روبروی هم، سمبل‌ها را آنقدر با هم مقایسه کنید تا به یک همخوانی برسید. طولانی‌ترین w ممکن را ایجاد کرده و سپس، طول آنرا با u مقایسه کنید. بدلیل آنکه یک lba وجود دارد، زبان مذکور حساس‌به‌متن بوده و یک گرامر حساس‌به‌متن به ازای آن وجود خواهد داشت.

سوالات چهارگزینه‌ای و کلید

آزمون اول

۱. اگر w = abbab یک رشته باشد چه تعداد پیشوند برای این رشته وجود دارد؟
الف. ۴ ب. ۵ ج. ۶ د. ۷

۲. اگر $L = \{a^n b^n \mid n \geq 0\}$ باشد آنگاه L^2 کدام است؟
الف. $L^2 = \{a^n b^n a^n b^n : n \geq 0\}$
ب. $L^2 = \{a^n b^n a^m b^m : n,m \geq 0\}$
ج. $L^2 = \{a^{2n} b^{2n} : n \geq 0\}$
د. $L^2 = \{a^n b^n b^m a^m : n,m \geq 0\}$

۳. کدامیک از روابط زیر برای هر زبان L، صحیح است؟
الف. $\overline{L^*} = \overline{L}^*$ ب. $\overline{L^*} \subset \overline{L}^*$
ج. $\overline{L^*} \subset \overline{L}^*$ د. اطلاعات ناکافی است.

۴. برای DFA ای بصورت $M = (Q, \Sigma, \delta, q_0, F)$ کدامیک از گزاره‌های زیر صحیح است؟
الف. Σ می‌تواند نامحدود باشد. ب. q_0 می‌تواند وجود نداشته باشد.
ج. F می‌تواند تهی باشد. د. |Q| همواره بزرگتر از یک است.

۵. زبان پذیرش شده توسط DFA ای به نام M به صورت مجموعه $L(M) = \{w \in \Sigma^* : \delta^*(q_0, w) \in F\}$ تعریف می‌شود $\overline{L(M)}$ کدام است؟
الف. $\overline{L(M)} = \{w \in \Sigma^* : \delta^*(q_0, w) \notin F\}$

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

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

د. هرسه گزینه صحیح است.

۶. زبان مربوط به تابع انتقال DFA زیر، کدام است؟(حالت شروع q_0 است)

$Q=\{q_0,q_1,q_2,q_3\}$ ، $F=\{q_3\}$

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

$\delta(q_1,a)=q_1$ $\quad\delta(q_1,b)=q_1$

$\delta(q_2,a)=q_3$ $\quad\delta(q_2,b)=q_2$

$\delta(q_3,a)=q_3$ $\quad\delta(q_3,b)=q_2$

الف. $L=\{waw \quad : w\in\{a,b\}^*\}$

ب. $L=\{aw \quad : w\in\{a,b\}^*\}$

ج. $L=\{w\,a \ : w\in\{a,b\}^*\}$

د. $L=\{awa : w\in\{a,b\}^*\}$

۷. زبان پذیرش شده بوسیله NFA ای به صورت $M=(Q,\Sigma,\delta,q_0,F)$ در کدام گزینه آمده است؟

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

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

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

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

۸. برای پذیرش زبان زیر، یک NFA با حداقل چند حالت نیاز است؟

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

الف. ۳ ب. ۴ ج. ۵ د. ۶

۹. عبارت منظم زبان $L=\{a^nb^m : (n+m)\,is\,even\}$ کدام است؟

الف. $(aa)^*+(bb)^*$ ب. aa^*+bb^*

ج. $((aa)^*a(bb)^*b+(aa)^*(bb)^*)$ د. $((aa)^*+(bb)^*)(a+b)$

۱۰. کدام یک از گرامرهای زیر منظم است؟(S سمبل شروع)

الف. $\{S\to aSb\mid\lambda\}$ ب. $\{S\to aS \mid Sb \mid \lambda\}$

ج. $\begin{cases}S\to aS\mid\lambda\\A\to Sb\end{cases}$ د. $\{S\to Sa \mid Sb \mid \lambda\}$

۱۱. منظم بودن زبان L_1، چگونه شرطی، برای آنکه گرامر منظم G وجود داشته بطوریکه $L_1=L(G)$ است؟

الف. شرط لازم ب. شرط کافی

ج. شرط لازم وکافی د. نمی‌توان حرفی زد.

۱۲. اگر L_1 و L_2 زبان‌های منظم باشند آنگاه کدامیک از زبان‌های زیر منظم هستند؟

الف. L_1^R ، L_1^* ، L_1L_2 ، $L_1\cap L_2$ ، $L_1\cup L_2$

ب. L_1^R ، $\overline{L_1}$ ، L_1L_2 ، $L_1\cap L_2$ ، $L_1\cup L_2$

ج. L_1^R ، L_1^* ، $\overline{L_1}$ ، L_1L_2 ، $L_1\cap L_2$ ، $L_1\cup L_2$

د. L_1^* ، $\overline{L_1}$ ، L_1L_2 ، $L_1\cap L_2$ ، $L_1\cup L_2$

۱۳. برای دو زبان L_1 و L_2 زیر، L_1/L_2 کدام است؟

$(L_1/L_2=\{x: xy\in L_1 \text{ for some } y\in L_2\})$

$L_2=\{b^m : m\geq 1\}$ و $L_1=\{a^n b^m : n\geq 1, m\geq 0\}\cup\{ba\}$

الف. $\{a^n b^m : n\geq 0, m\geq 1\}$ ب. $\{a^n b^m : n\geq 1, m\geq 0\}$

ج. L_2 د. L_1

۱۴. خانوادهٔ زبان‌های منظم تحت کدامیک از عملگرهای زیر بسته می‌باشد؟

$nor(L_1,L_2)=\{w: w\notin L_1 \text{ and } w\notin L_2\}$

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

$\min(L)=\{w\in L: \text{there is no } u\in L, v\in\Sigma^+, \text{such that } w=uv\}$

الف. $L_2\setminus L_1$ ، $nor(L_1,L_2)$

ب. $\min(L)$ ، $L_2\setminus L_1$

ج. $\min(L)$ ، $nor(L_1,L_2)$

د. $\min(L)$، $L_2\setminus L_1$ ، $nor(L_1,L_2)$

۱۵. برای خانواده زبان‌های منظم کدامیک از مسائل زیر می‌تواند بررسی شود و جواب آن تعیین شود؟

P_1: آیا $L_1=L_2$ $\quad P_2$: آیا $L_1\subseteq L_2$

P_3: آیا $L=L^R$ $\quad P_4$: آیا L is empty

الف. P_3,P_2,P_1 ب. P_4,P_3,P_2

ج. P_4,P_3,P_1 د. P_4,P_3,P_2,P_1

۱۶. کدامیک از زبان‌های زیر منظم نیستند ؟

$L_1=\{ww^R : w\in\{a\}^*\}$

$L_2=\{a^p : p\geq 2, p \text{ is a prime number}\}$

$L_3=\{a^{n!} : n\geq 0\}$ $\quad L_4=\{a^n : n=2^K, K\geq 0\}$

الف. L_3,L_2,L_1 ب. L_4,L_1,L_2

ج. L_4,L_3,L_1 د. L_4,L_3,L_2,L_1

۱۷. کدامیک از گرامرهای زیر، زبان $L=\{ww^R : w\in\{a,b\}^*\}$ را تولید می‌کند؟

الف. $S\to aSa$ $\quad S\to bSb$ $\quad S\to\lambda$

ب. $S\to aSa$ $\quad S\to bSb$ $\quad S\to ab$

ج. $S\to aSSa$ $\quad S\to bSSb$ $\quad S\to\lambda$

د. $S\to Sab$ $\quad S\to abS$ $\quad S\to\lambda$

۱۸. فرض کنید Σ یک الفبا و L یک زبان بر روی Σ و N مجموعه اعداد طبیعی باشد. L یک زبان قابل تعریف بوسیلهٔ اتوماتای متناهی نیست اگر

الف. $(\exists n\in N)(\forall x\in L \text{ such that } |x|\geq n)(\exists u,v,w\in\Sigma^*) \text{such that } x=uvw,$ $|uv|\leq n, |v|\geq 1, \text{and } (\forall i\in N)(uv^iw\in L)$

ب. $(\forall n\in N)(\exists x\in L \text{ such that } |x|\geq n)(\forall u,v,w\in\Sigma^*) \text{such that } x=uvw,$ $|uv|\leq n, |v|\geq 1, \text{and } (\exists i\in N \text{ such that } uv^iw\notin L)$

ج. $(\forall n\in N)(\exists x\in L \text{ such that } |x|\geq n)(\forall u,v,w\in\Sigma^*) \text{such that } x=uvw,$ $|uv|\leq n, |v|\geq 1, \text{and } (\exists i\in N \text{ such that } uv^iw\in L)$

د. $(\exists n\in N)(\forall x\in L \text{ such that } |x|\geq n)(\exists u,v,w\in\Sigma^*) \text{such that } x=uvw,$ $|uv|\leq n, |v|\geq 1, \text{and } (\forall i\in N)(uv^iw\notin L)$

۱۹. برای گرامرمستقل‌ازمتن زیر، درخت اشتقاق مربوط به رشته w = abbbb دارای چندگره خواهد بود؟

$S\to aAB$
$A\to bBb$
$B\to A|\lambda$

الف. ۱۳ ب. ۱۱ ج. ۵ د. ۷

۲۰. کدامیک از گرامرهای زیر مبهم نمی‌باشند ؟

G_1:
$E\to E+T|T$
$T\to F$
$F\to I$
$I\to a|b|c$

G_2:
$E\to E+E|E*E$
$E\to(E)$
$E\to I$
$I\to a|b|c$

G_3:
$S\to aSb|SS|\lambda$

الف. G_1 ب. G_2, G_1
ج. G_3, G_2 د. G_3, G_2, G_1

۲۱. با توجه به گرامرزیر، کدام گزینه صحیح است؟

G:
$S\to S_1|S_2$
$S_1\to S_1c|A$
$A\to aAb|\lambda$
$S_2\to aS_2|B$
$B\to bBc|\lambda$

الف. G غیر مبهم است.

ب. G مبهم است ولی گرامر دیگری هم ارز با آن و غیر مبهم وجود دارد.
ج. G مبهم است ولی گرامر دیگری هم ارز با آن و غیر مبهم وجود ندارد.
د. L(G) منظم است.

۲۲. گرامر هم ارز گرامر زیر که دارای قاعدهٔ λ نباشد دارای چند قانون خواهد بود؟

G:
$S\to ABaC$
$A\to BC$
$B\to b|\lambda$
$C\to D|\lambda$
$D\to d$

الف. ۱۱ ب. ۱۲
ج. ۱۳ د. ۱۴

۲۳. کدامیک از گرامرهای زیر در فرم نرمال چامسکی قرار دارد؟

الف. $\begin{cases}S\to aABB\\A\to bAB|b\\B\to bB|b\end{cases}$

ب. $\begin{cases}S\to AS|a\\A\to SA|b\end{cases}$

ج. $\begin{cases}S\to abA\\A\to baS|\lambda\end{cases}$

د. $\begin{cases}S\to SAa\\A\to b\end{cases}$

۲۴. برنامه تابع انتقال رویرو کدام زبان را پذیرش می‌کند؟

$\delta(q_0,\lambda,z)=\{(q_1,z)\}$
$\delta(q_0,a,z)=\{(q_0,0z)\}$
$\delta(q_0,b,z)=\{(q_0,1z)\}$
$\delta(q_0,a,0)=\{(q_0,00)\}$
$\delta(q_0,b,0)=\{(q_0,\lambda)\}$
$\delta(q_0,a,1)=\{(q_0,\lambda)\}$
$\delta(q_0,b,1)=\{(q_0,11)\}$

الف. $L=\{a^nb^n:\ n\geq 0\}$ ب. $L=\{w\in\{a,b\}^*:n_a(w)=n_b(w)\}$
ج. $L=\{ww^R:w\in\{a,b\}^*\}$ د. $L=\{w\in\{a,b\}^*:n_a(w)=2n_b(w)\}$

۲۵. کدامیک از زبان‌های زیر زبان مستقل‌ازمتن معین است؟

$L_1=\{a^nb^n:\ n\geq 0\}$, $L_2=\{a^nb^{2n}:\ n\geq 0\}$, $L_3=L_1\cup L_2$
$L_4=\{a^nb^mc^k:\ n=m \text{ or } m=k\}$, $L_5=\{w\in\{a,b\}^*:n_a(w)\neq n_b(w)\}$

الف. L_3, L_2, L_1 ب. L_3, L_1, L_2

ج. L_5, L_2, L_1 د. L_5, L_4, L_3

۲۶. کدامیک از زبان‌های زیر مستقل‌ازمتن هستند؟

$$L_1 = \{ww \ : w \in \{a,b\}^*\} \quad , \quad L_2 = \{ww^R \ w : w \in \{a,b\}^*\}$$
$$L_3 = \{a^n ww^R b^n : w \in \{a,b\}^* , n \geq 0\} \quad , \quad L_4 = \{a^n b^m c^n \mid m < 1000, \ n \geq 0\}$$

الف. L_4, L_1 ب. L_3, L_2 ج. L_3, L_1 د. L_4, L_3

۲۷. اگر L_1 و L_2 مستقل‌ازمتن باشدآنگاه کدامیک از زبان‌های زیر مستقل‌ازمتن هستند؟

الف. $L_1^*, \ L_1 L_2, \ L_1 \cap L_2 \ , L_1 \cup L_2$

ب. $\overline{L}_1, \ L_1^*, \ L_1 L_2, \ L_1 \cap L_2$

ج. $L_1^R, \ L_1^*, \ L_1 \cup L_2, \ L_1 L_2$

د. $L_1 - L_1 \ , \ L_1 \cap L_1, \ L_1 \cup L_1, \ L_1^R, \ L_1^*, \ \overline{L}_1$

۲۸. اگر L_1 مستقل‌ازمتن و L_2 منظم باشد، آنگاه کدام زبان زیر مستقل‌ازمتن است؟

الف. $L_1 - L_2$ ب. $L_1 \cap L_2$ ج. $L_1 \cap L_2 \ , \ L_1 - L_2$ د. هیچکدام

۲۹. کدامیک از مسائل زیر در خانوادهٔ زبان‌های مستقل‌ازمتن قابل بررسی می‌باشد و می‌توان جواب آنرا تعیین کرد؟

P_1: آیا $L_1 = L_2$ $\quad$ P_2: آیا L is empty

P_3: $\lambda \in L$ $\quad$ P_4: $L = \{\lambda\}$

الف. P_3 و P_2, P_1 ب. P_4 و P_3, P_2

ج. P_4 و P_3, P_1 د. P_4 و P_2, P_1

۳۰. چنانچه بخواهیم یک ماشین تورینگ برای محاسبهٔ تابع زیر طراحی کنیم استفاده از کدامیک ازمجموعه واحدهای تورینگ آمادهٔ زیر، کار را سریع تر خواهد کرد؟

$$f(x,y) = \begin{cases} x+y & \text{i } x \geq y \\ 0 & \text{if } x < y \end{cases}$$

الف. Copier, Adder, comparer

ب. subtracter, multiplier, Adder

ج. Eraser, Adder, comparer

د. Copier , Eraser, multiplier

۳۱. تابع انتقال یک ماشین تورینگ به صورت زیر می‌باشد، تابع محاسبه شده توسط این ماشین کدام است؟

$$Q = \{q_0, q_1, q_2, q_3, q_4\}, \ F = \{q_4\}$$
$$\delta(q_0, 1) = (q_0, 1, R)$$
$$\delta(q_0, 0) = (q_1, 1, R)$$
$$\delta(q_1, 1) = (q_1, 1, R)$$
$$\delta(q_1, \square) = (q_2, \square, L)$$
$$\delta(q_2, 1) = (q_3, 0, L)$$
$$\delta(q_3, 1) = (q_3, 1, L)$$
$$\delta(q_3, \square) = (q_4, \square, R)$$

الف. $F(x,y)=x+y$ ب. $f(x)=2x$

ج. $f(x)=x+1$ د. $f(x,y)=x-y$

۳۲. تابع انتقال یک ماشین تورینگ به همراه وضعیت تغییر در سکون به صورت زیر می‌باشد:

$$\delta : Q \times \Gamma \rightarrow Q \times \Gamma \times \{L, R, S\}$$

تفسیر L, R همان معنای خود در ماشین تورینگ استاندارد را دارند و S یعنی اینکه هد (نوک خواندن / نوشتن) حرکت نکرده و Stay (سکوندار)باشد. این مدل از نظر قدرت محاسبه، چگونه است؟

الف. ضعیف تراز TM استاندارد

ب. برابر TM استاندارد

ج. قوی تراز TM استاندارد

د. یک مدل ماشین جدید می‌شود که برای برخی از توابع قوی تر است.

۳۳. کدامیک از مدل ماشین‌های تورینگ زیر قوی تر از نوع استاندارد آن می‌باشند؟

مورد اول: ماشین تورینگ چند نواره (Multitape)

مورد دوم: ماشین تورینگ چندبعدی(Multidimensional)

مورد سوم: ماشین تورینگ با چند هد(Multihead)

مورد چهارم: ماشین تورینگ نامعین (Nondeterministic)

الف. موارد اول و دوم ب. موارد دوم وسوم

ج. هرچهار مورد د. همه برابر TM استاندارد هستند.

۳۴. تعریف زیر چه اتوماتایی را تعریف می‌کند:

" این مدل اتوماتا، یک ماشین تورینگ نامعین (Nondeterministic TM) به فرم $M = (Q, \Sigma, \Gamma, \delta, q_0, \square, F)$ با همان تعریف ماشین تورینگ استاندارد می‌باشد که دو سمبل ویژهٔ '[' و ']' را برروی نوار دارد به گونه ای که $\delta(q_i, [)$ تنها شامل $(q_j, [, R)$ می‌باشدو $\delta(q_m,])$ تنها شامل $(q_n,], L)$ می‌باشد، این مدل محدود شده هم ارز با کدام مدل زیر است؟

الف. اتوماتای کراندارخطی (linear bounded Automata)

ب. اتوماتای دو پشته ای pushdown(Two pushdown Automata)

ج. اتوماتای پشته ایpushdown (pushdown Automata)

د. اتوماتای متناهی (finite Automata)

۳۵. کدامیک از گزاره‌های زیر صحیح نیست ؟

الف. هر زبان منظم، مستقل‌ازمتن است.

ب. هر زبان خطی، مستقل‌ازمتن است.

ج. هر زبان حساس‌به‌متن، بازگشتی است.

د. هر زبان حساس‌به‌متن، مستقل‌ازمتن است.

کلید آزمون اول

	الف	ب	ج	د
۱			√	
۲		√		
۳		√		
۴		√		
۵	√			
۶				√
۷	√			
۸		√		
۹			√	
۱۰				√
۱۱			√	
۱۲			√	
۱۳		√		
۱۴				√
۱۵				√
۱۶		√		
۱۷	√			
۱۸		√		
۱۹		√		
۲۰	√			
۲۱			√	
۲۲				√
۲۳		√		
۲۴		√		
۲۵			√	
۲۶				√
۲۷			√	
۲۸			√	
۲۹		√		
۳۰			√	
۳۱	√			
۳۲		√		
۳۳				√
۳۴	√			
۳۵				√

آزمون دوم

۱. اگر w = abbab یک رشته باشد، رشته v=ab عضو کدام یک از مجموعه‌های زیر برای رشته w می‌باشد؟

الف. پیشوندی، پسوندی　　ب. زیررشته، پیشوندی

ج. زیررشته ، پسوندی　　د. هر سه مجموعه

۲. برای آنکه $L^0 = \{\lambda\}$ باشد، کدامیک از شرایط زیر برای زبان L لازم است؟

الف. $L = \Sigma^*$　　ب. $L = \{\}$

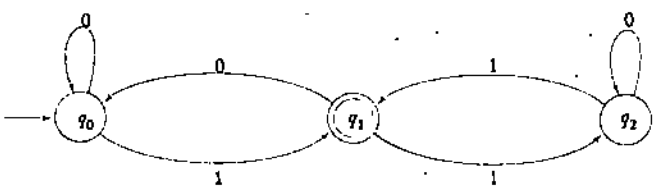
ج. $L = \{\lambda\}$　　د. $L \subseteq \Sigma^*$

۳. اگر $L = \{a^n b^m \mid n,m \geq 0\}$ باشد آنگاه L^R کدام است؟

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

ج. $L^R = \overline{L}$　　د. $L^R = \{a^n b^m : n,m < 0\}$

۴. در کدامیک از مجموعه رشته‌های زیر همگی توسط پذیرنده متناهی معین(DFA) زیر، پذیرش نمی‌شود؟



الف. 101 ,0111,11001

ب. 100 ,000100,1010010

ج. 00001,111110,111111

د. 11110001,000110,1111110

۵. زبان پذیرش شده بوسیله پذیرنده متناهی نامعین (NFA) به صورت $M = (Q,\Sigma,\delta,q_0,F)$، کدام است؟

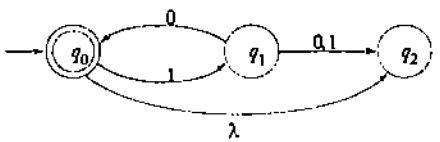
الف. $L(M) = \{w \in \Sigma^* : \delta^*(q_0,w) \cap F \neq \varphi\}$

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

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

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

۶. زبان پذیرش شده توسط پذیرنده متناهی نامعین (NFA) زیر کدام است؟



الف. $L(M) = \{(01)^n : n \geq 0\}$

ب. $L(M) = \{(10)^n : n \geq 0\}$

ج. $L(M) = \{(01)^n : n \geq 1\}$

د. $L(M) = \{(1010)^n : n \geq 0\}$

۷. برای پذیرش زبان روبرو، یک پذیرنده متناهی نامعین (NFA) با حداقل چند حالت، نیاز است؟

$L = \{ab,abc\}^*$

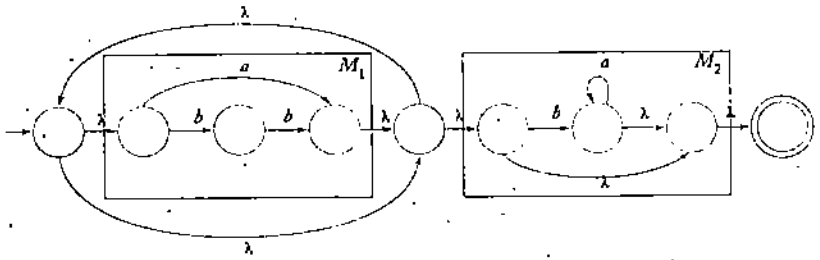
الف. ۳　　ب. ۴　　ج. ۵　　د. ۶

۸. عبارت منظم زبان $L = \{a^n b^m : (n+m)\ is\ odd\}$ کدام است؟ (odd یعنی فرد)

الف. $(aa)^* + (bb)^*$　　ب. $((aa)^* a(bb)^* b + (aa)^* (bb)^*)$

ج. $aa^* + bb^*$　　د. $((aa)^* a(bb)^* + (aa)^* (bb)^* b)$

۹. عبارت منظم مربوط به زبان پذیرش شده پذیرنده متناهی نامعین (NFA) زیر کدام است؟



الف. $r = ((a+bb^*)(ba^* + aa)^*)$　　ب. $r = ((a+bb^*)^*(aba^* + \lambda))$

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

۱۰. زبان تولید شده توسط گرامرمنظم روبروکدام است؟

$P_{G10}: S \to S_1 \mid S_2$

$S_1 \to aaS_1 \mid A$

$A \to bbA \mid \lambda$

$S_2 \to aaS_2 \mid aB$

$B \to bbB \mid b$

الف. $L(G) = \{a^n b^m : n,m\ is\ even\}$

ب. $L(G) = \{a^n b^m : n,m\ is\ odd\}$

ج. $L(G) = \{a^n b^m : n+m\ is\ even\}$

د. $L(G) = \{a^n b^m : n+m\ is\ odd\}$

۱۱. کدام گزاره زیر همواره صحیح است؟

الف. هر گرامر خطی، خطی چپ است.

ب. هر گرامر خطی، منظم است

ج. هر گرامر خطی راست، منظم است.

د. هر سه گزاره

۱۲. اگر L_1 و L_2 زبان‌های منظم باشند آنگاه کدامیک از زبان‌های زیر منظم هستند؟

الف. L_1^R ، L_1^* ، $L_1 L_2$ ، $L_1 \cap L_2$ ، $L_1 \cup L_2$

ب. L_1^R ، $\overline{L}_1$ ، L_1L_2 ، $L_1 \cap L_2$ ، $L_1 \cup L_2$

ج. L_1^* ، $\overline{L}_1$ ، L_1L_2 ، $L_1 \cap L_2$ ، $L_1 \cup L_2$

د. L_1^R ، L_1^* ، $\overline{L}_1$ ، L_1L_2 ، $L_1 \cap L_2$ ، $L_1 \cup L_2$

۱۳. فرض کنید $\Sigma = \{a,b\}$ و $\Gamma = \{b,c,d\}$ وتابع همریختی h به صورت زیر تعریف شده باشد، اگر L زبان منظمی باشد که با عبارت $r = (a+b^*)(aa)^*$ نشان داده شود آنگاه کدام عبارت منظم زیر نشان دهنده زبان منظم $h(L)$ می‌باشد؟

h: $h(a) = dbcc$
$h(b) = bdc$

الف. $r = (b+a^*)(bb)^*$

ب. $r = (dbcc + (bdc)^*)(dbccdbcc)^*$

ج. $r = (bdc + (dbcc)^*)(bdcbdc)^*$

د. $r = (b+c+d)^*$

۱۴. برای خانواده زبان‌های منظم کدامیک از مسائل زیر می‌تواند بررسی شود و جواب آن تعیین شود؟

$L_1 = L_2$ آیا : P_1 $\quad$ $L = L^*$ آیا : P_2

$L = L^R$ آیا : P_3 $\quad$ $\lambda \in L$ آیا : P_4

الف. P_3, P_2, P_1 $\quad$ ب. P_4, P_3, P_2

ج. P_4, P_3, P_1 $\quad$ د. P_4, P_3, P_2, P_1

۱۵. کدامیک از زبان‌های زیر منظم نیستند ؟

$L_1 = \{ww^R : w \in \{a,b\}^*\}$

$L_2 = \{a^p : p \geq 2, p \text{ is a prime number}\}$

$L_3 = \{a^n b^m : n \neq m\}$

$L_4 = \{a^n : n = 2^K, K \leq 2000\}$

الف. L_3, L_2, L_4 $\quad$ ب. L_4, L_3, L_2

ج. L_4, L_3, L_1 $\quad$ د. L_4, L_3, L_2, L_1

۱۶. کدامیک از گرامرهای زیر، زبان $L = \{wcw^R : w \in \{a,b\}^*\}$ را تولید می‌کند؟

الف. $S \to aSa$ $\quad$ $S \to bSb$ $\quad$ $S \to acb$

ب. $S \to aSa$ $\quad$ $S \to bSb$ $\quad$ $S \to c$

ج. $S \to aSSa$ $\quad$ $S \to bSSb$ $\quad$ $S \to c$

د. $S \to Sab$ $\quad$ $S \to abS$ $\quad$ $S \to c$

۱۷. گزاره زیر برای چه شرایطی برقرار می‌باشد؟

گزاره: اگر L_1 و L_2 زبان‌های نامنظم روی الفبای Σ باشند، آنگاه زبان $L_1 \cup L_2$ منظم است.

الف. برای تمامی زبان‌های نا منظم L_1 و L_2

ب. خی زبان‌های نامنظم L_1 و L_2

ج. تنها برای حالت $L_1 = L_2$

د. تحت هیچ شرایطی برقرار نمی‌باشد.

۱۸. برای گرامر مستقل‌ازمتن زیر، درخت اشتقاق رشته w = abbbb دارای چندگره برگ می‌باشد؟

$S \to aAB$
$A \to bBb$
$B \to A|\lambda$

الف. ۱۳ $\quad$ ب. ۱۱ $\quad$ ج. ۵ $\quad$ د. ۷

۱۹. کدامیک از گرامرهای زیر مستقل‌ازمتن می‌باشند ؟

G_1 :
$E \to E+T|T$
$T \to F$
$F \to I$
$I \to a|b|c$

G_2 :
$E \to E+E|E*E$
$E \to (E)$
$E \to I$
$I \to a|b|c$

G_3 :
$S \to aSb|SS|\lambda$

الف. G_1 $\quad$ ب. G_2, G_1 $\quad$ ج. G_3, G_2 $\quad$ د. G_3, G_2, G_1

۲۰. با توجه به گرامرمستقل‌ازمتن زیر، کدام گزینه صحیح است؟

G :
$S \to S_1|S_2$
$S_1 \to S_1 c|A$
$A \to aAb|\lambda$
$S_2 \to aS_2|B$
$B \to bBc|\lambda$

الف. G یک گرامر خطی و مبهم است.

ب. G یک گرامر خطی و غیرمبهم است.

ج. G یک گرامر غیرخطی و مبهم است.

د. یک گرامر غیر خطی و غیرمبهم است.

۲۱. گرامر هم ارز گرامر زیر که دارای قاعدهٔ بی فایده نباشد دارای چند قانون خواهد بود؟

G :
$S \to aS|A|C$
$A \to a$
$B \to bb$
$C \to aCb$

الف. 6 $\quad$ ب. 4

ج. 3 $\quad$ د. 2

۲۲. کدامیک از گرامرهای زیر در فرم نرمال گریباخ قرار دارد؟

الف. $\begin{cases} S \to aABB \\ A \to bAB|b \\ B \to bB|b \end{cases}$

ب. $\begin{cases} S \to AS|a \\ A \to SA|b \end{cases}$

ج. $\begin{cases} S \to abA \\ A \to baS|\lambda \end{cases}$

د. $\begin{cases} S \to SAa \\ A \to b \end{cases}$

۲۳. تابع انتقال ماشین پشته ای(PDA) روبرو، کدام زبان را پذیرش می‌کند؟

$$\Sigma=\{a,b\}\ ,\Gamma=\{0,1\}\ ,z=0,F=\{q_f\}$$
$$\delta(q_0,a,0)=\{(q_1,10)\}$$
$$\delta(q_0,\lambda,0)=\{(q_f,\lambda)\}$$
$$\delta(q_1,a,1)=\{(q_1,11)\}$$
$$\delta(q_1,b,1)=\{(q_2,\lambda)\}$$
$$\delta(q_2,b,1)=\{(q_2,\lambda)\}$$
$$\delta(q_2,\lambda,0)=\{(q_f,\lambda)\}$$

الف. $L=\{a^n b^n:\ n\geq 0\}$ ب. $L=\{w\in\{a,b\}^*:n_a(w)=n_b(w)\}$

ج. $L=\{ww^R:w\in\{a,b\}^*\}$ د. $L=\{w\in\{a,b\}^*:n_a(w)=2n_b(w)\}$

۲۴. کدامیک از زبان‌های زیر مستقل‌ازمتن هستند؟

$$L_1=\{a^n b^m c^n:\ m<1000,\ n\geq 0\}$$
$$L_2=\{ww^R\ \ w:w\in\{a,b\}^*\}$$
$$L_3=\{a^n ww^R b^n:w\in\{a,b\}^*\ ,n\geq 0\}$$
$$L_4=\{a^n b^m:n=2^m\}$$

الف. L_2، L_1 ب. L_3، L_2 ج. L_3، L_1 د. L_4، L_3

۲۵. اگر L_1 و L_2 مستقل‌ازمتن باشند آنگاه کدامیک از زبان‌های زیر مستقل‌ازمتن هستند؟

الف. L_1^*، L_1L_2، $L_1\cap L_2$، $L_1\cup L_2$

ب. $\overline{L}_1$، L_1^*، L_1L_2، $L_1\cap L_2$

ج. L_1^R، L_1^*، $\overline{L}_1$، $L_1\cup L_2$، $L_1\cap L_2$، L_1-L_2 د. L_1^R، L_1^*، $L_1\cup L_2$، L_1L_2

۲۶. کدامیک از مسائل زیر در خانوادهٔ زبان‌های مستقل‌ازمتن قابل بررسی می‌باشد و می‌توان جواب آنرا تعیین کرد؟

P_1: آیا $L_1=L_2$ P_2: آیا L is empty

P_3: $\lambda\in L$ P_4: $L=L^R$

الف. P_2 و P_3 ب. P_3 و P_4 ج. P_1 و P_4 د. P_1 و P_2

۲۷. با پذیرش "تز تورینگ" به عنوان تعریف یک محاسبه مکانیکی، کدام یک ازاستدلال‌های زیر بدست نمی‌آید؟

الف. هرچیزی قابل انجام برروی هر کامپیوتر رقمی، با یک ماشین تورینگ نیز قابل انجام است.

ب. هیچکس تاکنون قادر نبوده است مسئله ای پیشنهاد کند و با آنچه به اصطلاح الگوریتم می‌نامیم قابل حل باشد ولی بر روی یک ماشین تورینگ قابل حل نباشد.

ج. مدلهای دیگری برای محاسبه مکانیکی پیشنهاد شده اند، ولی تنها یکی از آنها قویتر از مدل تورینگ می‌باشد.

د. هر سه استدلال بدست می‌آید.

۲۹. کدامیک از گزاره‌های زیر صحیح است؟

گزاره اول: ماشین کراندار خطی قوی تر از ماشین پشته ای است.

گزاره دوم: ماشین متناهی از ماشین پشته ای ضعیف تر است.

گزاره سوم: ماشین کراندار خطی هم ارز با ماشین تورینگ است.

الف. اول و دوم ب. دوم و سوم ج. اول و سوم د. هر سه گزاره

۲۸. تابع انتقال یک ماشین تورینگ به صورت زیر می‌باشد، تابع محاسبه شده توسط این ماشین کدام است؟

$Q=\{q_0,q_1,q_2,q_3,q_4\}$, $F=\{q_4\}$

$\delta(q_0,a)=(q_1,x,R)$
$\delta(q_1,a)=(q_1,a,R)$
$\delta(q_1,y)=(q_1,y,R)$
$\delta(q_1,b)=(q_2,y,L)$
$\delta(q_2,y)=(q_2,y,L)$
$\delta(q_2,a)=(q_2,a,L)$
$\delta(q_2,x)=(q_0,x,R)$
$\delta(q_3,\sqcup)=(q_4,\sqcup,R)$

$\delta(q_0,y)=(q_3,y,R)$
$\delta(q_3,y)=(q_3,y,R)$

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

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

۳۰. کدامیک از زبان‌های زیر حساس‌به‌متن هستند؟

$$L_1=\{a^n b^n c^{2n}:\ n\geq 0\}$$
$$L_2=\{w\ w:w\in\{a,b\}^+\}$$
$$L_3=\{w\in\{a,b,c\}^*:n_a(w)=n_b(w)=n_c(w)\}$$
$$L_4=\{w\in\{a,b,c\}^*:n_a(w)=n_b(w)\leq n_c(w)\}$$

الف. L_3 و L_2، L_1 ب. L_4 و L_1، L_2

ج. L_4 و L_3، L_2، L_1 د. L_4 و L_1، L_1

کلید آزمون دوم

	الف	ب	ج	د		الف	ب	ج	د
۱				√	۱۶		√		
۲				√	۱۷		√		
۳	√				۱۸				√
۴		√			۱۹				√
۵	√				۲۰	√			
۶		√			۲۱			√	
۷	√				۲۲	√			
۸				√	۲۳				
۹				√	۲۴			√	
۱۰			√		۲۵			√	
۱۱			√		۲۶	√			
۱۲				√	۲۷			√	
۱۳		√			۲۸		√		
۱۴				√	۲۹	√			
۱۵	√				۳۰				√

واژه نامه

انگلیسی - فارسی

A

Automaton	اتومات
Automata	اتوماتا
Argument	استدلال
Adder	افزایشگر
Abstract	انتزاعی
Accept	پذیرفتن
Accepter	پذیرنده
Alphabet	الفبا
Algorithm	الگوریتم

B

Binary trees	درختهای باینری(دودویی)

C

Claim	ادعا
Contradiction	برهان خلف
Closure	بستار

Encoding	رمزدارکردن
Expression	عبارت
Element	عضو
Empty set	مجموعه تهی
Null set	مجموعه صفر
Equivalent	هم‌ارز
Equivalence	هم‌ارزی
Edge	یال

F

False	نادرست
Function	تابع
Final state	حالت پایانی
Formal languages	زبانهای صوری
Flip-flop	فلیپ فلاپ
Finite	متناهی

G

Generate	تولیدکردن
General purpose computers	کامپیوترهای همه منظور
Grammar	گرامر

H

Height	ارتفاع
Hypothesis	فرضیه

I

Induction	استقراء
Intersection	اشتراک
Intersection	اشتراک
Internal states	حالتهای داخلی
Identifier	شناسه
Intuition	شهود

Configure	پیکربندی
Cartesian product	حاصل ضرب دکارتی
Concatenation	الحاق
Concatenation	الحاق
Cycle	دور
Carry digit	رقم نقلی
Construct	ساختار
Child	فرزند
Compiler	کامپایلر
Computability	محاسبه‌پذیری
Concept	مفهوم
Complement	مکمل
Complementation	مکمل‌گیری
Control unit	واحد کنترل

D

Deterministic automata	اتوماهای معین(اتوماتای قطعی)
Deduction	استنتاج
Difference	تفاضل
Description	توصیف
Domain	دامنه
Digit	رقم
Discoding	رمزگشایی کردن
Distributive law	قانون پخشی
DeMorgan law	قانون دمورگان
Directed graph	گراف جهت دار
Disjoint	مجزا
Derivation	اشتقاق
Diagram	نمودار

E

Empty	تهی
Empty string	رشته تهی

Inductive assumption	فرض استقرایی
Infinite	نامتناهی

L

Label	برچسب
Labeling	برچسب‌دار کردن
Leaves	برگها
Letter	حرف
Language	زبان
Level	سطح
Logical	منطقی

M

Mathematical	ریاضی
Multiplicative constant	ضریب ثابت
Machine design	طراحی ماشین
Membership	عضویت

N

Nondeterministic automata	اتوماتای نامعین(غیرقطعی)
No carry	بدون نقلی
Next state	حالت بعدی
Nonnegative	غیرمنفی
n-unit delay transducer	مبدل تأخیر n واحدی

O

Ordered trees	درختهای مرتب
Order-of-magnitude	مرتبه رشد
Order at least	مرتبه رشد حداقلی ... را
Order at most	مرتبه رشد حداکثری ... را

P

Proof by induction	اثبات با استفاده از استقراء
Proof by contradiction	اثبات با استفاده از تناقض

Prove	اثبات کردن
Principle	اصول
Partition	بخش
Programming	برنامه سازی
Proof by contradiction	برهان خلف
Positive closure	بستار مثبت
Prefix	پیشوند
Partial function	تابع جزئی
produce	تولید کردن
Power set	مجموعه توانی
Path	مسیر
Production	قانون تولید (قانون)

R

Recursive	بازگشتی
Range	برد
Relation	رابطه
Root	ریشه
Reflexivity rule	قانون انعکاسی
Reverse	معکوس

S

Star closure	بستار ستاره
Suffix	پسوند
Stream	جریان
Sentence	جمله
Storage	حافظه
String	رشته
Substring	زیررشته
Subset	زیرمجموعه
Symbol	سمبل(نشانه)
Sentential forms	فرم‌های جمله‌ای
Symmetry rule	قانون تقارن

Start variable	متغیر شروع

T

Time slot	برش زمانی
Transition function	تابع انتقالی
Total function	تابع کلی
Translate	ترجمه کردن
Term	جمله
Temporary storage	حافظه‌های موقت
Tree	درخت
Terminal symbols	سمبل‌های پایانی(ترمینال)
Truth	صحت
Transitivity rule	قانون انتقال(تراگذاری)
Theorom	قضیه
Transducer	مترجم
Theory of computation	نظریه محاسبات

U

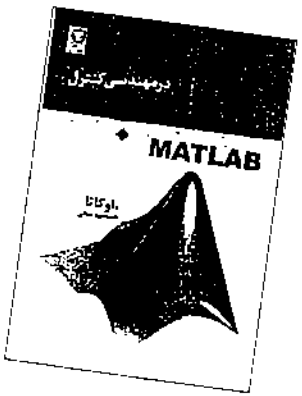
Union	اجتماع
Underscore	خط زیر
Unit-delay transducer	مبدل تأخیر واحد
Universal set	مجموعه جهانی

V

Vertices	رئوس
Variable	متغیر

نص

نص



ساده و واضح

آفیس ۲۰۰۷

نص