نظریه محاسبات

حسن عسکرزاده

قابل توجه دانشجویان و اساتید محترم

با توجه به مشکلات موجود در تایپ ناشی از یکسان سازی فرمت متن فصلهای مختلف همچنین ضرورت ارایه سوالات تستی و تشریحی بیشتر ،نگارش غلط گیری شده همچنین سوالات تستی و تشریحی و پاورپوینت این جزوه از آدرس www.askarzadeh.ir قابل دانلود کردن میباشد . همچنین هماهنگی مقدماتی لازم با مراکز توزیع کتب پیام نور در جهت ارایه نسخه چاپی بهتر نیز صورت گرفته است.جهت کسب اطلاعات بیشتر با شماره ۰۹۳۵۳۹۰۰۱۲۱ یا ۰۲۱۲۲۲۹۰۳۱۸ تماس بگیرید

Ramin.samad@yahoo.com

فهرست

پیشگفتار

با توجه به کمبود منبع فارسی در خصوص درس نظریه محاسبات و جایگزین شدن منبع این درس با دروسی با محتوای متفاوت و اسم مشابه مانند محاسبات عددی جزوه حاضر با تلاش گروهی از دانشجویان پیام نور دماوند و اینجانب تهیه گردید .بخش هایی از مطالب تهیه شده از مقالات ۲۰۰۷ اقتباس و ترجمه گردیده و با توجه به مشارکت دانشجویان در جای جای جزوه و بازخوانی مطالب آن به نظر میرسد بتوان از آن در حد جزوات و کتب دیگر پیام نور به عنوان منبع درسی استفاده کرد. اقدامات دیگری مانند الصاق دو شبیه ساز تورینگ و ماشین های محدود و مجموعه ای از سوالات تستی و تشریحی میتواند به خودآموز بودن و خودخوان بودن اثر کمک شایانی نماید اما محدویت صفحات کتاب مانع درج مطالب اضافه در آن گردیده است. مشکلات مطالعه این جزوه در درجه اول حجم نسبتا زیاد آن است که باید با توجه به واحد درسی محدود گردد و مشکل بعدی اشکالات نسبتا زیاد تایپی است که منشاء آن ویژگیهای مایکروسافت ورد در تایپ راست به چپ و ترکیب آن با فرمولهای نسبتا زیاد کتاب است که امیدواریم مشکل اول با اظهار نظر اساتید با تجربه در حین تدریس اعلان و رفع اشکال شود و مشکل دوم نیز با مشارکت دانشجویان عزیز و اعلان اشکالات جزوه هر چه سریعتر اصلاح گردد.

Ramin.samad@yahoo.com

مقدمه

در کتاب نظریه محاسبات به سه موضوع اساسی میپردازیم .موضـوع اول کـه در دروس قبل نیز مطرح گردیده اتوماتا است موضوع دوم پاسخ به این سوال است که آیـا میتـوان برای حل همه مسایل الگوریتمی یافت ؟ این مبحث با عنوان محاسبه پـذیری در کتـب نظریه محاسبات مطرح میگردد و مشخص خواهد شد که برای حل بـسیاری از مـسایل به نظر ساده ی اطراف ما هیچ الگوریتمی وجود ندارد. موضوع سـوم نظریـه پیچیـدگی محاسبات است که در پاسخ به اینکه آیا منابع حافظه و پردازشـی لازم بـرای حـل یـک مسئله وجود دارد یا خیر این کتاب از ۷ فصل تشکیل شده است .در ابتـدای هـر فـصل مقدمه ای اختصاصی آورده شده است که خواننده دلایل مطالعه فصل را بهتر درک کنـد و اگر این کتاب مرجع درسی خارج از مقطع کارشناسی پیام نور بود بـا توجـه بـه ایـن مقدمه میتوان فصولی از کتاب را حذف نمـود. بـرای دانـشجویان مهندسـی کـامپیوتر و علوم کامپیوتر که دروسی مانند نظریه زبانها و ماشینها و اتوماتا را گذرانده اند فصل اول شامل الفباها ، رشته ها ، زبانها ، گرامرها بیـشتر جنبـه یـادآوری دارنـد . فـصل دوم بـه موضوع برنامه های با حافظه متناهی و مبـدلهای آن مـی پـردازد ایـن فـصل هـم بـرای دانشجویان مهندسی کامپیوتر نقش یادآوری دارد در فصل سوم به برنامه های بازگشتی و مبدل های پشته ای خواهیم پرداخت .فصل چهـارم شـامل تـشریح ماشـین تورینـگ است .محاسبات با منابع محدود در فصل پنجم ارایه شده و در فـصل شـشم محاسـبات احتمالی مورد بررسی قرار خواهد گرفت و در پایان محاسبات مـوازی در فـصل هفـتم بررسی خواهد شد .به احتمال زیاد شما این کتاب را به همراه یک لـوح فـشرده شـامل مطالب کمک آموزشی و دو محیط پیاده سازی شبیه ساز ماشـین تورینـگ و شـبیه سـاز ماشینهای متناهی دریافت خواهید کرد .اما اگر این مهم بـه دلایـل مـدیریتی و اجرایـی تحقق نیافت سایت www.askarzadeh.ir به منظور ارایـه مطالـب فـوق و مباحـث مربوط به مولف شامل نمونه سوالات تستی و تشریحی همچنین اصلاح غلطهای تـایپی و محتوایی این جزوه در نظر گرفته خواهد شد البته تکمیل سایت فوق قطعا بعد از مهر ماه ۸۶ میسر خواهد شد.

ذ

نه

فصل اول

نگاه کلی به نظریه محاسبات

اهداف کلی

در پایان فصل، دانشجو با مفاهیم زیر آشنا می‌شود:

اهداف رفتاری

الفباها ، رشته ها و نمایش ها

زبانهای صوری و گرامرها

برنامه ها و مسایل

تقلیل پذیری

مقدمه

محاسبات به منظور پردازش اطلاعات طراحی شده اند که گاهی ساده تر از تخمین زمان رانندگی بین دو شهر و گاهی پیچیده تر از پیشگویی وضعیت هوا هستند.هدف از مطالعه محاسبات فراهم کردن بینشی نسبت به مشخصه های محاسبات است.چنین بینشی برای پیشگویی پیچیدگی محاسبات خواسته شده و برای انتخاب راههایی که باید در پیش بگیریم مفید بوده و برای توسعه و طراحی ابزارهایی که حل مسایل را آسان میکند استفاده می شود.

مطالعه محاسبات روشن می کند که مسائلی وجود دارند که حل شدنی نیستند. و در میان مشکلاتی که حل میشوند ،مواردی نیز وجود دارد که به مقدار نشدنی از منابع نیازمندند به عنوان مثال یک میلیون سال زمان محاسبات برای حل یک مسئله غیر عملی است حتی اگر برای آن الگوریتمی وجود داشته باشد .ممکن است دانستن این که مسئله ای قابل حل نیست به نظر دلسرد کننده بیاید، اما در عوض منابع ما برای حل

چنین مشکلاتی به هدر نمیرود .
مطالعه محاسبات هم ابزارهای تشخیص مسائل قابل حل را فراهم میکند و هم ابزاها و روشهای حل مسائل فوق را مهیا مینماید. بعلاوه مطالعه این نظریه دقت و تجربه ما را برای توسعه معنای خوب ودقیق مجموعه اصطلاحات را برای ارتباط مستقیم افکار درباره محاسبات توسعه میدهد.
هربحث رسمی پیرامون محاسبات و برنامه ها به یک فهم واضح از این نظریه ها و نیز نظریه های وابسته نیاز دارد. در این فصل به معرفی برخی مفاهیم اصلی استفاده شده در این کتاب خواهیم پرداخت .بخش اول این فصل راجع به نظریه رشته ها و نقشی که رشته ها در نمایش اطلاعات دارنداختصاص یافته است.بخش دوم مفهوم زبانها را مطابق نظریه رشته ها بازگو میکند وگرامرها را برای توصیف زبانه معرفی مینماید.بخش سوم با نظریه برنامه ها و مفهوم غیرجبری در برنامه ها سروکار دارد. بخش چهارم مفهوم مسایل را به صورت رسمی بیان میکند، و ارتباط بین مشکلات و برنامه ها را مطرح میکند.بخش پنجم توانایی کاهش پذیری میان مشکلات را بیان میکند.

۱-۱ الفباها،رشته ها ،و نمایش ها

توانایی نمایش اطلاعات در برابر مراوده و پردازش اطلاعات بسیارسخت است. جوامع بشری زبان های گفتگو را برای ارتباط برقرار کردن دریک سطح بنیانی خلق کردند و مکتوبات را تا رسیدن به یک سطح بالاتر توسعه دادند.
برای مثال زبان انگلیسی درشکل بیان شده خودش به برخی مجموعه های متناهی ازصداهای اصلی بعنوان یک مجموعه اولیه وابسته است. کلمات با لفظ دنباله متناهی از این قبیل صداها مشخص میشوند. جملات از دنباله متناهی از کلمات ناشی میشوند. مکالمات از دنباله متناهی از جملات و همینطور الی آخر بدست می آیند.
انگلیسی نوشتاری از برخی مجموعه های متناهی از نمادها بعنوان مجموعه ابتدایی استفاده میکند. جملات از ترتیب متناهی از کلمات مشتق میشوند. پاراگرافها (بندها) از ترتیب متناهی از جملات وغیره وغیره بدست می آیند.
بعلاوه برای نمایش عناصرمجموعه های دیگر راههای مشابهی ایجاد شده اند. برای

مثال عدد طبیعی میتواند بوسیله مجموعه متناهی از ارقام دهدهی نمایش داده شود.
محاسبات، نظیر زبانهای طبیعی، در بیشترین شکل عمومی خود چشم به توزیع اطلاعات دارند. در نتیجه، تابع محاسبات بعنوان دستکاری کننده اعداد صحیح ، گراف ها ، برنامه ها وخیلی از نمونه های دیگر. اگر چه ، در واقعیت محاسبات فقط رشته هایی از نمادها که اشیا را نمایش میدهند را دستکاری میکند. بحث قبلی تعاریف زیر را ایجاب میکند.

الفباها ورشته ها

یک مجموعه مرتب غیرتهی ، متناهی یک الفبا نامیده خواهد شد اگر عناصرش نمادها یا کارکترها باشند (به عنوان مثال عناصری با نمایش گرافیکی "اولیه"). یک دنباله متناهی نمادها از یک الفبای معین یک رشته روی الفبا نامیده خواهدشد. یک رشته که از یک ترتیب $a_1, a_2, \ldots, a_n$ از نمادها تشکیل شده با کنار هم گذاشتن $a_1a_2\ldots a_n$ معنی خواهد داد. رشته هایی که صفر نماد دارند ، رشته های تهی نامیده میشوند که با ε مشخص خواهد شد.

مثال ۱-۱-۱ $\Sigma_1 = \{0, \ldots, 9\}$ $\Sigma_2 = \{a, \ldots, z\}$ Σ الفبا هستند. abb یک رشته روی الفبای Σ_1 و۱۲۳ یک رشته روی Σ_2 است. ba12 یک رشته روی Σ_1 نیست ، زیرا شامل نمادهایی است که در Σ_1 نیستند. به طور مشابه ...۳۱٤ یک رشته روی Σ_2 نیست زیرا یک دنباله متناهی نیست. از طرف دیگر، ε یک رشته روی هر الفبایی هست.

مجموعه تهی Ø یک الفبا نیست چون عنصری ندارد . مجموعه اعداد طبیعی یک الفبا نیست چون متناهی نیست.

اجتماع $\Sigma_1 \cup \Sigma_2$ یک الفباست فقط در صورتیکه نمادهایش را به ترتیب جایگزین کنیم. یک الفبای اصلی ۲ یک الفبای دودویی نامیده میشود و رشته های روی یک الفبای دودویی رشته های دودویی نامیده میشوند. به صورت مشابه یک الفبای اصلی ۱ یک

الفبای یگانی نامیده میشود و رشته های روی یک الفبای یگـانی را رشـته هـای یگـانی نامند.

طول یک رشته α با $|\alpha|$ مشخص میشود و برابر تعداد نمادها در رشته است.

مثال ۱-۱-۲ $\{0, 1\}$یک الفبای دودویی اسـت و $\{1\}$ یـک الفبـای یگـانی. ۱۱یـک رشته دودویی روی الفبای $\{0, 1\}$ و ۱یک رشته یگانی روی الفبـای $\{1\}$اسـت. ۱۱یـک رشته با طول ۲ است ،$|\varepsilon|=0$ ، و $|01| + |1|$ برابر ۳ خواهد بود.

رشته عبارت است از یک دنباله α و به دنبال آن یک دنبالـه β کـه بـا $\alpha\beta$ مـشخص میشود. رشته$\alpha\beta$ الحاقی از α و β نامیده میشود. نشان گذاری α^i برای بدست آوردن رشته با الحاق i کپی از رشته α بکار میرود .

مثال ۱-۱-۳ از الحاق رشته ۰۱ با رشته ۱۰۰ رشته ۰۱۱۰۰ بدست مـی آیـد . الحـاق $\alpha\varepsilon$ بوسیله ε با هر رشته α ،والحاق $\varepsilon\alpha$ بوسیله رشته α با εرشـته α را میدهـد . یعنـی $\varepsilon\alpha=\alpha$.

اگر $\alpha=01$،پس $\alpha^0 = \varepsilon$ ، $\alpha^1 = 01$ ، $\alpha^2 = 0101$ ،و $\alpha^3 = 010101$.

یک رشته α یک زیر رشته از رشته β گفته میـشود اگـر بـرایγ و δ یـی داشـته باشـیم $\beta=\gamma\alpha\delta$. یک زیر رشته α از رشته β یک پیشوند از β گفته میشود اگربرای ρ یـی $\alpha\rho$ $=\beta$ باشد . پیشوند یک پیشوند سره ازβ گفته میشود اگر$\rho\neq\varepsilon$. یک زیر رشته α از یـک رشته β یک پسوند از β گفته میشود اگر برای γیی $\beta=\gamma\alpha$ باشد. پـسوند یـک پـسوند سره از β گفته میشود اگر $\gamma\neq\varepsilon$.

مثـال ۱-۱-٤ ε ، ۰ ، ۱ ،۰۱ ،۱۱ و۰۱۱ زیـر رشـته هـایی از ۰۱۱ هـستند. ε ، ۰ و۰۱ پیشوندهای سره ای از ۰۱۱ هستند. ε ، ۱ و۱۱ پسوندهای سره ای از ۰۱۱ هـستند. ۰۱۱ یک پیشوند و پسوند از ۰۱۱ است.

اگر برای برخی نمادهای $a_1, a_2, \ldots, a_n$ رامعکوس $a_n a_{n-1} \ldots a_1$ را α بنامیم آنگاه α نامند،که با علامت α^{rev} مشخص میشود. β جایگشتی از α گفته میشود اگر β بتواند از α بوسیله نمادهای دوباره مرتب شده در α بدست بیاید .

مثال ۱-۱-۵ : α را رشته ۰۰۱ میگیریم .α^{rev} =۱۰۰ . رشته های ۰۰۱ ، ۰۱۰ و ۱۰۰ جایگشتهایی از α هستند.

مجموعه ای از همه رشته ها روی الفبای Σ با Σ^* مشخص خواهد شد. Σ^+ با مجموعه $\Sigma^* - \{\epsilon\}$ مشخص خواهد شد.

مرتب سازی رشته ها

شاید کاربردی ترین عمل روی اطلاعات جستجو باشد.به علت اهمیت این عمل یافتن راههایی برای سازماندهی اطلاعات جهت تسهیل در جستجو توجه خاصی را می طلبد. از جستجوی ترتیبی ،جستجوی دودویی ،مرتب سازی سریع ومرتب سازی ادغامی میتوان به عنوان نمونه هایی از این روش ها نام برد. این روش ها در اغلب موارد به وجود یک رابطه که موجودیت ها در مسئله را معین میکند متکی هستند.

غالبا ارتباط برای رشته های واحدی استفاده میشود که آنها را به طور الفبایی مقایسه میکند، بعنوان مثال بازگرداندن اسامی مرتب شده در دفترتلفن. ارتباط و مرتب سازی میتواند در روش ذیل تعریف شود.

هر الفبای Σ را در نظر میگیریم. یک رشته α در Σ^* نسبت به یک رشته β به طورالفبایی کوچکتر گفته میشود یا بطورمشابه β در Σ^* نسبت به α به طورالفبایی بزرگتر گفته میشود اگر α وβ در Σ^* باشند و در هر دو مورد ذیل صدق کنند.

الف : α یک پیشوند سره ازβ است.

ب : برای ی در Σ^* و a و b در Σ چنین است که a مقدم است بر b در Σ ، رشته a یک پیشوند از است و رشته b پیشوندی از است.

یک زیرمجموعه مرتب از Σ^* به طور الفبایی مرتب گفته میشود ، اگر در Σ^* به طور الفبایی کوچکتر نباشد نسبت به هر وقت که بر در زیر مجموعه مقدم است .

مثال ۱-۱-٦ : Σ را الفبای دودویی {۰, ۱} میگیریم. رشته ۰۱ در Σ^* به طورالفبایی نسبت به رشته ۰۱۱۰۰ کوچکتراست زیرا ۰۱ یک پیشوند سره از ۰۱۱۰۰ است. از سوی دیگر ۰۱۱۰۰ به طورالفبایی از ۰۱۱۱ کوچکتراست زیرا دورشته در سه نماد اول مطابقت دارند و چهارمین نماد در ۰۱۱۰۰ کوچکتراست از چهارمین نماد در ۰۱۱۱.

مجموعه {ε و ۰ و ۰۰ و ۰۰۰ و ۰۰۱ و ۰۱ و ۰۱۰ و ۰۱۱ و ۱ و ۱۰ و ۱۰۰ و ۱۰۱ و ۱۱ و ۱۱۰ و ۱۱۱} ، از رشته هایی که طول بزرگتر از ۳ ندارند ،یک ترتیب الفبایی را میدهد.

برای دنباله های متناهی ترتیب الفبایی مناسب است زیرا در چنین مجموعه مرتبی هر رشته میتواند سرانجام بدست آید . همچنین به دلایل مشابه برای مجموعه نامتناهی از رشته های یگانی تربیت الفبایی مناسب است. اگرچه دربرخی موارد دیگرترتیب الفبایی مناسب نیست زیرا میتواند منجر به دستیابی برخی از رشته ها که مقدم بر تعداد بیکران ازرشته های دیگر است شود. برای مثال چنین وضعیتی برای رشته ۱ در مجموعه به طور الفبایی مرتب * {۱و۰} بوجود می آید ، به عبارت دیگر ۱ مقدم است بر رشته های ۰ و ۰۰ و ۰۰۰ و... این نقص تعریف زیرین را از ترتیب استاندارد برای رشته ها موجب میشود . در ترتیب استاندارد هر رشته بر یک عدد متناهی از رشته ها مقدم است.

یک رشته در Σ^* نسبت به رشته به طور استاندارد کوچکتر یا به طور لغتی کوچکترگفته میشود یا به طور هم ارز (متشابها) ، در Σ^* نسبت به به طور استاندارد بزرگتر یا به طور لغتی بزرگترگفته میشود اگر در هر دو مورد ذیل صدق کنند.

الف : α کوتاهتر از β است.

ب : α و β دارای طول یکسان هستند اما α به طور الفبایی کوچکتر از β است.

یک زیرمجموعه مرتب از Σ^* متعارفا مرتب یا به طور لغتی مرتب گفته میشود، اگر β در Σ^* نسبت به α متعارفا کوچکتر نباشد هر وقت در زیر مجموعه α مقدم است بر β.

مثال ۱-۱-۷ الفبای $\Sigma = \{0, 1\}$ راملاحظه کنید. رشته ۱۱ در Σ^* متعارفا کوچکتراست از رشته۰۰۰ زیرا ۱۱ نسبت به ۰۰۰ یک رشته کوتاهتر است . ازطرف دیگر، ۰۰ از ۱۱ متعارفا کوچکتر است ،زیرارشته ها از لحاظ طول یکسانند و۰۰ بطورالفبایی کوچکتراز ۱۱ است.

مجموعه $\Sigma^* = \{\varepsilon, 0, 1, 00, 01, 10, 11, 000, 001, \ldots\}$ درترتیب استاندارد خودش مفروض است.

نمایش ها

تعاریف قبلی الفباها و رشته ها مفروض است ، نمایش اطلاعات میتواند بعنوان نگاشتی از اشیاء داخل رشته ها در مطابقت با برخی قوانین درنظرگرفته شود . به عبارت دیگر به طور رسمی گفته شده یک نمایش یا رمزگذاری روی یک الفبای Σ ازیک مجموعه Dیک تابع fاز D برابر ۲ به توان Σ^* است که وضعیت زیر را برقرار میکند :

برای هر جفت از عناصر مجزا e_1 و e_2 در D، $f(e_1)$ و $f(e_2)$ مجموعه های غیرتهی غیرمتصل هستند.

اگر Σ یک الفبای یگانی است آنگاه نمایش آن یک نمایش یگانی گفته میشود. اگرΣ یک الفبای دودویی باشد ،آنگاه نمایش آن یک نمایش دودویی گفته میشود.

درآنچه که درزیر آمده هرعنصردر f(e) متعلق به یک نمایش یا رمزگـذاری از e خواهـد بود.

مثال ۱,۱,۸ یک نمایش دودویی روی {۱و۰} از اعداد طبیعی است اگر f_1:

$f_1(0) = \{0, 00, 000, 0000, \ldots\}$ $f_1(1) = \{1, 01, 001, 0001, \ldots\}$

$f_1(2) = \{10, 010, 0010, 00010, \ldots\}$ $f_1(3)=\{11, 011, 0011, 00011, \ldots\}$

$f_1(4) = \{100, 0100, 00100, 000100,\ldots\}$ و غیره.

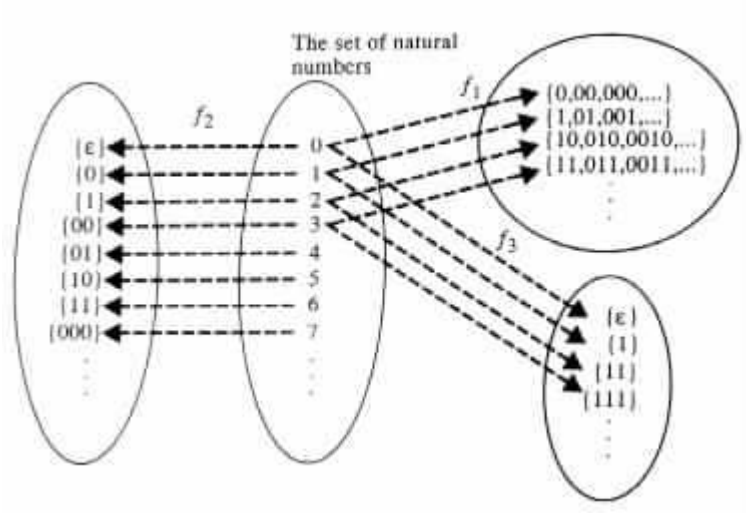
بطور مشابه ، f_2 یک نمایش دودویی روی {۱و۰} از اعداد طبیعـی اسـت اگـر بوسـیله i امین بطور متعارف کوچکترین رشته دودویی به i امین عـدد طبیعـی مجموعـه سـازگار اختصاص داده شود . در چنین موردی

$f_2(0) = \{\varepsilon\}$, $f_2(1) = \{0\}$, $f_2(2) = \{1\}$, $f_2(3) = \{00\}$, $f_2(4) = \{01\}$, $f_2(5) = \{10\}$,$f_2(6) = \{11\}$, $f_2(7) = \{000\}$, $f_2(8) = \{1000\}$, $f_2(9) = \{1001\}$, …

از طرف دیگر، f_3 یک نمایش یگانی روی {۱} از اعداد طبیعی است اگر بوسیله i امـین بطور الفبایی (قانونا) کوچکترین رشته یگانی به iامین عـدد طبیعـی مجموعـه سـازگار اختصاص داده شود . درچنین موردی

$f_3(0) = \{\varepsilon\}$, $f_3(1) = \{1\}$, $f_3(2) = \{11\}$, $f_3(3) = \{111\}$, $f_3(4) = \{1111\}$

سه نمایش f_1، f_2 و f_3 در شکل ۱-۱-۱ شرح داده شده اند.



شکل ۱-۱-۱ نمایشی برای اعداد طبیعی.

تا انتهای کتاب تابع f_1 از مثال ۱-۱-۸ نمایش دودویی ازاعـداد طبیعـی فـرض میـشود مگراینکه در جایی طور دیگری اظهار شود.

۱-۲ زبانهای صوری و گرامرها :

عالم رشته ها یک واسطه ی سودمند برای نمایش اطلاعات است به شرطی که در آنجا یک تابع که تفسیری برای اطلاعات منتقل شده توسط رشته ها را فراهم میکند وجود داشته باشد . یک تفسیر عکس راهنمایی است که یک نمایش فراهم میکند. به عبارت دیگر یک تفسیر, یک تابع g از $\sum^*$ به D است برای چند الفبای $\sum$ و چند مجموعه ی D. به عنوان مثال رشته ی ۱۱۱میتواند مانند عدد صدویازده و به عنوان نماینده ی یک رشته ی دسیمال تفسیر شود ,یا به عنوان یک رشته ی دو دویی نماینده ی عدد هفت باشد و یک رشته ی یکانی نشان دهنده ی عدد سه.

طرفین ارتباط یک بخش از اطلاعات کار نمایش و تفسیر را انجام میدهند. عمل نمایش بوسیله ی فرستنده و عمل ترجمه به وسیله ی دریافت کننده فراهم میشود. پردازش

اهمیت ندارد اگرچه طرفین انسانها یا برنامه ها هستند.
نتیجتا از نقطه نظر طرفین مورد بحث یک زبان میتواند فقط یک مجموعه از رشته ها باشد زیرا طرفین توابع تفسیری و نمایشی را در خودشان درج میکنند.
زبانها:

در حالت کلی اگر $\sum$ یک الفبا باشد و L یک زیر مجموعه روی $\sum^*$ آنگاه به L یک زبان روی $\sum$ یا به طور ساده تر یک زبان گفته میشود. اگر $\sum$ شناخته شده باشد به هر عنصر از L یک جمله یا کلمه یا رشته میگویند

مثال ۱-۲-۱: {0,11,001},{ε,10},{0,1}*,{0,1}* زیر مجموعه های {0,1}* هستند بنابراین آنها زبانهای روی الفبای{0,1} هستند

مجموعه ی تهیØ و مجموعه {ε} زبانهای روی هر الفبایی هستند . Ø زبانی است که شامل هیچ رشته ای نیست . { ε} یک زبان است که فقط شامل رشته ی تهی میباشد.

اجتماع دو زبان L۱ و L۲ به صورت L1 U L۲ نمایش داده میشود و بیانگر زبانی است که شامل همه ی رشته های موجود در هر یک از زبانهای L۱ یا L۲ است . که به صورت {X is in L1 or x is in L2}{X نمایش داده میشود. اشتراک L۱ و L۲ مشخص کننده ی $L_1 \cap L_2$ است و نشان دهنده ی زبانهایی است که شامل همه ی رشته هایی موجود در L۱ و L۲ میباشد و به صورت $\{x \mid x \text{ is in } L_1 \text{ and in } L_2\}$ نمایش داده میشود.

متمم گیری یک زبان L روی $\sum$ یا فقط متمم گیری L وقتیکه $\sum$ شناخته شده است مشخص کننده ی $\bar{L}$ است . که نشان دهنده ی زبانهایی است شامل همه ی رشته های روی $\sum$ که در L نیستند که بصورت $\{x \mid x \text{ is in } \sum^* \text{ but not in } L\}$ نشان داده میشود .

مثال ۱-۲-۲

زبانهای $L_1=\{\varepsilon,0,1\}$ و $L_2=\{\varepsilon,01,11\}$ را در نظر بگیرید .

اجتماع این زبانها $L_1UL_2=\{\varepsilon,0,1,01,11\}$ و اشتراکشان $L_1\cap L_2=\{\varepsilon\}$ و متمم L_1 $L_1^{-}=\{00,01,10,11,000,001,\ldots\}$ خواهد بود

برای هر زبان L داریم $\emptyset UL=L$ و $\emptyset\cap L=\emptyset$ از سوی دیگر برای هر الفبای $\sum$ داریم $(\emptyset^{*-}=\sum^{*})$ و $(\sum^{*-}=\emptyset)$.

تفاضل L_1 و L_2 نمایشگر $L_1 - L_2$ است که نشان دهنده ی زبانی است که شامل همه ی رشته های موجود در L_1 است که این رشته ها در L_2 نمیباشد و به صورت $\{x \mid x \text{ is in } L_1 \text{ but not in } L_2\}$. نمایش داده میشود. حاصلضرب L_1 و L_2 نمایانگر L_2 . L_1 است که نمایش دهنده ی مجموعه ی همه ی جفت (X,Y) هایی است که X در L_1 و Y در L_2 است و به صورت $\{(x, y) \mid x \text{ is in } L_1 \text{ and } y \text{ is in } L_2\}$ نمایش داده میشود. ترکیب L_1 وL_2 به صورت L_1L_2 نشان داده میشود ونمایانگر زبانهایی به صورت $\{xy \mid x \text{ is in } L_1 \text{ and } y \text{ is in } L_2\}$.است.

مثال ۱-۲-۳

اگر $L_1 = \{\varepsilon, 1, 01, 11\}$ و $L_2 = \{1, 01, 101\}$ باشد انگاه

$L_2 - L_1 = \{101\}$و $L_1 - L_2 = \{\varepsilon, 11\}$ از سوی دیگر اگر $L_1 = \{\varepsilon, 0, 1\}$ و $L_2 = \{01, 11\}$ انگاه حاصلضرب وترکیبشان این زبانها به ترتیب برابر است

$L_1 \times L_2 = \{(\varepsilon, 01), (\varepsilon, 11), (0, 01), (0, 11), (1, 01), (1, 11)\}$

$L_1L_2 = \{01, 11, 001, 011, 101, 111\}$

برای هر زبان L داریم $\emptyset - L = \emptyset L - \emptyset = L$ $L=L\{\varepsilon\}$

همچنین L^i برای نمایش i کپی یک زبان L استفاده می شود. در اینجا طبق تعریف L^o برابر است با $\{\varepsilon\}$. مجموعه‌ی $L^3 \cup L^2 \cup L^1$ مجموعه بسته کلین[1] (بستار کلین) نامیده می شود یا فقط بستار زبان L که با L^* نشان داده می شود. مجوعه‌ی $L^3 \cup L^2 \cup L^1$ مجموعه مثبت بسته[2] زبان L نامیده می شود و با L^+ نشان داده می شود .

L^i شامل همه رشته هایی است که بوسیله‌ی الحاق i رشته از L بدست می آیند و شامل آن رشته هایی است که بوسیله‌ی الحاق یک عدد فرضی از L* بدست می آیند.

مثال ۱-۲-٤

در نظر بگیرید جفت زبانهای L_1 و L_2 را در نظر بگیرید . برای این زبانها

$L_1 = \{\varepsilon, 0, 1\}$ $L_2 = \{01, 11\}$ $L_1^2 = \{\varepsilon, 0, 1, 00, 01, 10, 11\}$,
$L_2^3 = \{010101, 010111, 011101, 011111, 110101, 110111, 111101, 111111\}$.

بعلاوه ε در L_1^* و L_1^+ و L_2^* بوده اما در L_2^+ نیست .

عملیات فوق در یک روش مشابه به رابطه ها در $\Sigma^* \times \Delta^*$ وقتی که Σ, Δ الفباها هستند به کار برده می شود . بویژه اجتماع رابطه های R_1 و R_2 که به صورت R1UR2 نشان داده می شود و بیانگر رابطه‌ی $\{(x,y) | (x,y) is in R_1 or in R_2\}$. اشتراک R_1, R_2 به صورت $(R_1 \cap R_2)$ است و بیانگر رابطه‌ی $\{(x,y) | (x,y) is in R_1 and R_2\}$. ترکیب R ۱ با R ۲ R مشخص کننده R_1 R ۲ است که به صورت رابطه‌ی $\{(x1x2, y1y2) | (x1, y1) is in R1 and (x2, y2) is in R_2\}$ نمایش داده می شود .

مثال ۱-۲-۵ :بررسی رابطه های $R1 = \{(\varepsilon, 0)(10, 1)\}$ و $R2 = \{(1, \varepsilon), (0, 01)\}$

برای این رابطه ها $R1 \cup R2 = \{(\varepsilon, 0)(10, 1)(1, \varepsilon)(0, 01)\}$ و $R1 \cap R2 = \phi$ و

$R1R2 = \{(1,0)(0,001)(101,1)(100,101)\}$ و

$R2R1 = \{(1,0)(110,1)(0,010)(010,011)\}$

متمم گیری یک رابطه R در $\Sigma^* \times \Delta^*$ یا فقط متمم گیری R وقتی Σ, Δ شناخته شده

---

1 **kleen clouser**
2 **Positive clouser**

اند . به صورت $\overline{R}$ نمایش داده می شود و بیانگر رابطه‌ی $\{(x,y)|(x,y)i\sin\Sigma^* \times \Delta^* butnotinR\}$است . معکوس R مشخص کننده R^{-1}که به صورت رابطه‌ی $\{(x,y)|(x,y)i\sin R\}$ نمایش داده میشود.

$R^o = \{(\varepsilon,\varepsilon)\}$. $R^i = R^{i-1}R(i \geq 1)$

مثال ۱-۲-٦

اگر رابطه‌ی $\{(\varepsilon,\varepsilon),(\varepsilon,01)\}$ باشد آنگاه $R^{-1} = \{(\varepsilon,\varepsilon),(\varepsilon,01)\}$و

$R^2 = \{(\varepsilon,\varepsilon),(\varepsilon,01),(\varepsilon,0101)\}$

به زبانی که شامل تعداد متناهی قاعده کلی و تعداد متناهی قاعده استنباطی داشته و بوسیله‌ی یک سیستم صوری تعریف شود یک زبان صوری گفته می شود .

گرامرها :

بیان کردن زبانها با قواهد گرامری اغلب راحت است زیرا به جای بکار بردن تعداد زیادی جمله از تعداد کمی قوانین استفاده میکنیم به عنوان مثال امکانی که یک جمله‌ی انگلیسی شامل یک عبارت فاعلی از یک عبارت خبری پیروی می کند می تواند به وسیله یک قاعده‌ی دستوری به شکل $\langle sentence \rangle \rightarrow \langle subject \rangle \langle predicate \rangle$ بیان شود . به همین نحو یک عبارت فاعلی شامل یک عبارت اسمی به صورت $\langle subject \rangle \rightarrow noun$ بیان می شود . در یک حالت مشابه می توان استنباط کرد که Mary sang song یک جمله‌ی امکان پذیر در زبانی است که توسط قواعد دستوری زیر بیان می شود .

<sentence>→<subject><preicate>
<subject>→<noun>
<predicate>→<verb><article><noun>
<noun>→<name>
<noun>→<string>
<name>→<u-character><string>
<string>→<string><character>
<string>→<character>
<character>→ α
<character>→Z

<u-character>→A
.
.
.
<u-character>→Z
<verb>→sang
<article>→ α

قواعد دستوری بالا به جمله های انگلیسی از شکل Mary sang song برای نام ها در کنار Mary اجازه می دهند . از سوی دیگر قواعد دستوری جملاتی شبیه Mary sang song و Mary read a song را نمی پذیرند . بنابراین مجموعه قواعد دستوری بالا شامل یک سیستم دستوری ناقص برای بیان زبان انگلیسی است .

برای بررسی رفتار در این باره کافیست فقط گرامرهایی که شامل مجموعه هایی متناهی قواعد دستوری با فرم قبلی هستند بررسی شوند . چنین گرامرهایی گرامرهای نوع صفر یا عبارت قواعد ساختاری نامیده می شوند و به زبانهای صوری که بوسیله‌ی آنها تولید می‌شوند زبانهای نوع صفر می گویند .

هر گرامر نوع صفر و مانند یک سیستم ریاضیاتی شامل یک عبارت چهار تایی $< N, \Sigma, P, S >$ بیان می شود که در آن

N : یک الفباست که عناصرش علائم غیر پایانه نامیده می شوند .

Σ : یک الفبای جدا شده از N که عناصرش نمادهای پایانه نامیده می شوند .

P : یک رابطه‌ی متناهی اصلی روی $(N \cup \Sigma^*)$ که عناصرش قواعد تولیدی نامیده می‌شوند . علاوه بر این هر قاعده تولیدی (α, β) در p به صورت $\alpha \to \beta$ باید حداقل یک نماد غیر پایانه در α داشته باشد . در این چنین قواعد تولیدی به α سمت چپ قاعده تولید و به β سمت راست قاعده تولید می گویند .

S : نماد شروع که در N است .

مثال ۱-۲-۷ : $<N,\Sigma,P,S>$ یک گرامر نوع صفر است

اگر $p=\{s\to asb, s\to\varepsilon\}, \Sigma=\{a,b\}, N=\{s\}$ این گرامر یک نماد غیر پایانی منفرد s ، دو پایانه a,b و دو قاعده تولید $s\to\varepsilon, s\to asb$ دارد.هر دو قاعده‌ی تولید در سمت چپ فقط نماد غیر پایانی S دارند . سمت راست اولین قاعده‌ی تولیدی aSb و سمت راست دومین قاعده ε قرار دارد.

$<N_1,\Sigma_1,p_1,s>$ یک گرامر نیست اگر N1 مجموعه اعدا طبیعی بوده یا Σ_1 تهی باشد زیرا Σ_1, N_1 الفبا هستند .

اگر

$N_2=\{s\}$ و $\Sigma_2=\{a,b\}$ و $P_2=\{s\to asb, s\to\varepsilon, ab\to s\}$ آنگاه $<N_2,\Sigma_2,p_2,s>$ یک گرامر نیست زیرا s→ab به این نیاز که هر قاعده ی تولیدی باید شامل حداقل یک غیر پایانه ی سمت چپ باشد پاسخ نمی دهد .

در حالت کلی نمادهای غیر پایانه یک گرامر نوع صفر بوسیله‌ی S و حروف اول بزرگ در الفبای انگلیسی D,C,B,A و E مشخص می شوند . نماد آغازین با S مشخص می شود . نمادهای پایانی بوسیله‌ی اعداد و حروف اولیه‌ی کوچک در الفبای انگلیسی d,c,b,a و e مشخص می شوند . نمادهای کم ارزش بوسیله‌ی Z,Y,X مشخص می شوند . رشته های نمادهای پایانی بوسیله‌ی آخرین کاراکترهای کوچک انگلیسی (u,v,w,x,y,z) مشخص می شوند. رشته های شامل نمادهای پایانی و غیر پایانی بوسیله‌ی اولین نمادهای کوچک یونانی α,β,γ مشخص می شوند و برای راحتی ، رشته های قواعد تولیدی به فرم

$\alpha\to\beta_1$
$\alpha\to\beta_2$
. . .
$\alpha\to\beta_n$

به صورت

$\alpha \to \beta_1$
$\to \beta_2$
. . .
$\to \beta_n$

نمایش داده می شود .

مثال ۱-۲-۸ : $< N,\Sigma,P,S >$ یک گرامر نوع صفر است اگر {N={S,B و $\Sigma = \{a,b,c\}$ و P شامل دنباله‌ی قواعد تولیدی باشد .

$s \to \alpha BSc$
$\to abc$
$\to \varepsilon$
$B\alpha \to \alpha B$
$Bb \to bb$

سمت چپ سه قاعده‌ی تولیدی نخستین نمادهای غیر پایانی S است . Ba سمت چپ چهارمین قاعده‌ی تولید است . Bb سمت چپ پنجمین قاعده‌ی تولید است .

سمت راست نخستین قاعده‌ی تولید شامل نمادهای پایانی و غیر پایانی aBSc است . سمت راست (abc) دومین قاعده‌ی تولید شامل فقط نمادهای پایانی است . به استثنای حالت جزئی سمت راست ε سومین قاعده هیچ سمت راست قاعده‌ی تولید شامل غیر پایانی نیست حتی اگر این قبیل فرمها قابل پذیرش باشند .

اشتقاق ها:

گرامرها بوسیله ی اصلاح یا تغییر رشته های معین به طور مکرر زبانها را به وجود می آورد .هر تغییر مطابق با بعضی قواعد تولید گرامر در مسئله $< N,\Sigma,P,S >$ یک رشته است. یک تغییر در یک رشته γ بوسیله ی جایگزینی یک زیر رشته γ با β بر طبق قاعده تو لید $\alpha \to \beta$ مشتق می شود

در حالت کلی به یک رشته ی γ اشتقاق مستقیم یک رشته ی γ' گفته می شود

اگر γ' بتواند از γ به وسیله یک تغییر منفرد بدست بیاید . به طور مشابه یک رشته ی γ اشتقاق γ' نامیده می شود اگر γ' بتواند از γ بوسیله یک رشته ی اعداد اختیاری از اشتقاق مستقیم بدست آید.

صریحا به یک رشته ی γ اشتقاق مستقیم یک رشته ی γ' در G گفته می شود و به شکل $\gamma \Rightarrow_G \gamma'$ نمایش داده می شود اگر γ' از γ بوسیله ی جایگزینی یک زیر رشته α با β در جاییکه $\alpha \rightarrow \beta$ یک قاعده ی تولید در G است بتواند بدست آید.

بطوریکه اگر برای بعضی رشته های α و β, δ, ρ , داشته باشیم $\gamma = P$ و $\alpha \quad \delta$ $\beta \, P \, \delta$ $\gamma' =$ در اینصورت $\alpha \rightarrow \beta$ یک قاعده ی تولید در G است .

مثال ۱-۲-۹ : اگر G گرامر $< N, \Sigma, P, S >$ در مثال ۱-۲-۷ باشد آنگاه ε و aSb مستقیما از s قابل اشتقاق هستند به طور مشابه ab و a^2Sb^2 مستقیما از aSb قابل اشتقاق هستند. ε مستقیما از s قابل اشتقاق است و ab طبق قاعده ی تولیدی $\varepsilon \rightarrow s$ مستقیما از asb مشقق می شود . asb مستقیما از s و a^2Sb^2 طبق قاعده ی تولیدی $s \rightarrow asb$ مشقق می شود .

از سوی دیگر اگر G گرامر $< N, \Sigma, P, S >$ مثال ۱-۲-۸ باشد.آنگاه بر طبق قاعده ی تولید $Ba \rightarrow aB$ داریم:

$aBaBabccc \Rightarrow GaaBBabccc$, $aBaBabccc \Rightarrow_G aBaaBbccc$

بعلاوه رشته ی دیگری مستقیما از $aBaBabccc$ در G مشتق نمی شود.

γ مشتق γ' در G نامیده می شود و بصورت $\gamma \Rightarrow_G {}^*\gamma'$ مشخص می شود اگرداشته باشیم

$\gamma_0 \Rightarrow_G \ldots \Rightarrow_G \gamma'_n$ برای برخی $\gamma_0, \ldots, \gamma_n$ که در آن $\gamma_0 = \gamma$ و $\gamma_n = \gamma'$ در بعضی موارد به رشته ی $\gamma_0 \Rightarrow_G \ldots \Rightarrow_G \gamma'_n$ اشتقاق γ از γ' گفته می شود که طول آن مساوی با n است . اگر $\gamma_0 = s$ باشد به $\gamma_0, \ldots, \gamma_n$ صورتهای جمله ای از γ گفته می شود . به یک قالب جمله ای که شامل نماد پایانه ای نیست یک جمله گفته می شود .

مثال ۱-۲-۱۰ : اگر G گرامر مثال ۱-۲-۷ باشد آنگاه a^4Sb^4 یک اشتقاق از s دارد . اشتقاق $S \Rightarrow_G {}^*a^4Sb^4$ طول ٤ دارد و بصورت

$S \Rightarrow_G aSb \Rightarrow_G a^2Sb^2 \Rightarrow_G a^3Sb \Rightarrow_G a^4Sb^4$ است

یک رشته در زبانی که گرامر G به وجود می آورد فرض می شود اگر و فقط اگرآن رشته از نماد های پایانی باشد که از نماد های آغازین مشتق شده است.زبان تولید شده توسط G بصورت L(G) نمایش داده می شود که مجموعه ی تمام رشته های نماد های پایانی است که می توانند از نماد های اولیه مشتق شوند که

مجموعه ی $\{ w|w \text{ is in } \Sigma^* \text{ and } S \Rightarrow_G^* w \}$ است . به هر رشته در زبان L(G) تولید شده به وسیله G گفته می شود.

مثال ۱-۲-۱۱ : گرامر G از مثال ۱-۲-۷ را در نظر بگیرید. ε در زبانی که G بوجود آورده است قرار دارد به علت وجود اشتقاق $S \Rightarrow_G \varepsilon$. ab در زبانی که Gبوجود آورده به دلیل وجود اشتقاق $S \Rightarrow_G asb \Rightarrow_G ab$ وجود دارد. a^2b^2 نیزدر زبانی که G بوجود آورده به دلیل اشتقاق $S \Rightarrow_G aSb \Rightarrow_G a^2Sb^2 \Rightarrow_G a^2b^2$ وجود دارد .

زبان L(G) که G بوجود آورنده ی آن است شامل همه ی رشته های به فرمa…ab…b است که در آن تعداد a ها برابر تعداد b ها است به صورت

$L(G) = \{a^i b^i | \ i \text{ یک عدد طبیعی است}\}$

aSb در L(G) نیست زیرا شامل یک نماد غیر پایانه است .a^2b در L(G) نیست زیرا نمیتواند از S در G مشتق شود.

نماد $\gamma \Rightarrow \gamma'$, $\gamma \Rightarrow \gamma$ وقتی G شناخته شده باشد استفاده $\gamma \Rightarrow_G^* \gamma'$,$\gamma \Rightarrow_G \gamma$ بترتیب به جای می شود . بعلاوه به گرامرهای نوع صفر وقتی هیچ بی نظمی رخ نداده باشد برای سادگی گرامر گفته میشود. .

مثال ۱-۲-۱۲ : اگر Gگرامر مثال ۱-۲-۸ باشد آنگاه دنباله ی زیر یک اشتقاق برای $a^3b^3c^3$ است زیر خط و بالای خط رشته ها سمت چپ و راست قواعد تولیدی هستند که به ترتیب از قواعد تولیدی در اشتقاق استفاده می کنند .

$$\underline{S} \Rightarrow \overline{aB\underline{S}c}$$
$$\Rightarrow aB\overline{aB\underline{S}c}c$$
$$\Rightarrow aBa\underline{B}\overline{abc}cc$$
$$\Rightarrow aBa\overline{aB}\underline{b}ccc$$
$$\Rightarrow a\underline{B}a\overline{ab}bccc$$
$$\Rightarrow a\overline{aB}abbccc$$
$$\Rightarrow aa\overline{aB}bbccc$$

زبان به وجود آمده بوسیله ی گرامرG شامل تمام رشته ها ی به فرم a…ab…bc…c که در آن تعداد a ها برابر تعداد b ها و c ها است و به صورت زیر می باشد .

$$L(G) = \{ a^i b^i c^i \mid i \text{ یک عدد طبیعی است} \}$$

دو قاعده ی نخستین در G برای تولید صورتهای جمله ای که الگوی aBaB….aBabc…c را دارد استفاده می شود . در هر "قالب جمله ای[1]" تعداد a ها برابر تعداد c ها و یکی بیشتر از تعداد b ها می باشد .

قاعده ی تولید ی Ba→aB در "قالب جمله ای" برای انتقال B ها به طرف راست استفاده می شود.قاعده ی تولیدی Bb→bB برای جایگزینیB با b ها به محض رسیدن به وضعیت مناسب شان استفاده می شود.

گرافهای اشتقاق:

اشتقاق های قالب جمله ای در گرامرهای نوع صفر را بوسیله ی اشتقاق یا تجزیه یا گرافها میتوان نمایش داد. هر گراف اشتقاق یک گراف ریشه دار مستقیم, منظم و بدون چرخه است که گره هایش برچسب دار هستند . برچسب هر گره یکی از نمادهای پایانی یا غیر پایانی یا یک رشته ی تهی است . گراف اشتقاقی که مطابق با یک اشتقاق S=>γ1=>…=>γn بیان میشود به صورت قیاسی در روش زیر تعریف شده است.

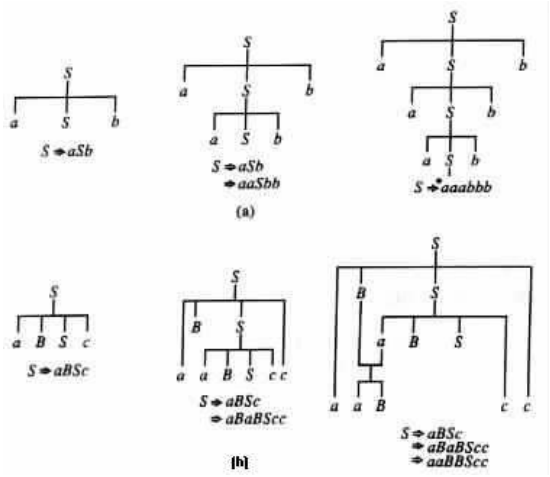
الف : اشتقاق گراف D_0 مطابق است با S و شامل یک گره ی منفرد برچسب دار با نماد شروع S است.

1 sentential forms

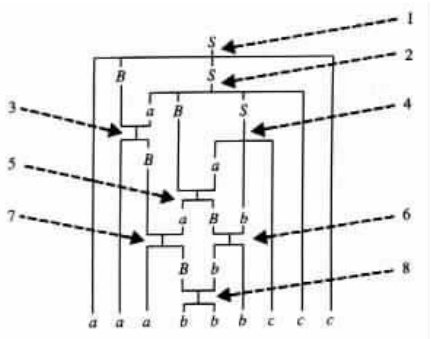
ب : اگر $\alpha \rightarrow \beta$ قاعده ی تولیدی استفاده شده در اشتقاق مستقیم $\gamma_i => \gamma_{i+1} : 0 \leq i < n_0$ و $\gamma_0 = S$ باشد آنگاه گراف اشتقاق D_{i+1} که مطابق است با $\gamma_0 \Rightarrow \ldots \Rightarrow \gamma_{i+1}$ از D_i بدست می آید و دارای حداکثر $\max(|\beta|, 1)$ گره های جدید است.

گره های جدید بوسیله ی کاراکترهای β برچسب گذاری میشوند و به عنوان جانشین هر گره ی Di که برابر یک کاراکتر در α است اختصاص می یابند . در نتیجه برگهای اشتقاق گراف D_{i+1} با γ_{i+1} برچسب گذاری میشوند.گراف های اشتقاق زمانیکه گرافهای مستقیم درخت هستند درختهای اشتقاق یا درختهای تجزیه نامیده میشوند

مثال ۱-۲-۱۳ :شکل ۱-۲-۱ (a) نمونه ای از درخت های اشتقاق برای اشتقاق های گرامر مثال ۱-۲-۷ را نشان میدهد. شکل ۱-۲-۱ (b) گراف های اشتقاق برای اشتقاق های گرامر مثال ۱-۲-۸ را نشان میدهد



شکل ۱-۲-۱ (a) درختهای اشتقاق . (b) گراف های اشتقاق .



شکل ۱-۲-۲ درخت اشتقاق با ترتیب معمول قواعد تولید که با پیکانها نمایش داده شده است .

چپ ترین اشتقاق :

یک اشتقاق $\gamma_0 \Rightarrow \ldots \Rightarrow \gamma_n$ چپ ترین اشتقاق نامیده می شود اگر α_1 در اشتقاق قبل از α_2 قرار بگیرد. و وقتی که دو شرط زیر رخ دهد :

الف : α_1 در γ_i سمت چپ γ_i ظاهر شود. $(0 \le i < n)$

ب : α_1 و α_2 به ترتیب در طی اشتقاق برطبق بعضی قواعد دستوری به فرم $\alpha_1 \to \beta_1$ و $\alpha_2 \to \beta_2$ جایگزین می شوند .

مثال ۱-۲-۱٤ : گراف اشتقاق در شکل ۱-۲-۲ نشان می دهد که قواعد تولیدی در اشتقاق $a^3b^3c^3$ مثال ۱-۲-۱۲ استفاده می شوند. زیر رشته ی $\alpha_1=aB$ که در هفتمین مرحله ی اشتقاق جایگزین می شود شبیه فرم جمله ای زیر رشته ی $\alpha_2=Bb$ است که در ششمین مرحله ی اشتقاق جایگزین می شود. این اشتقاق یک اشتقاق چپ نیست زیرا α_1 باید سمت چپ α_2 ظاهر شود در حالیکه بعد از α_2 جایگزین شده .

$$
\begin{aligned}
\underline{S} &\Rightarrow \overline{ab\underline{Sc}} \\
&\Rightarrow \underline{aB\overline{a}}\overline{B\underline{S}cc} \\
&\Rightarrow a\overline{aB}B\underline{S}cc \\
&\Rightarrow aaB\underline{B\overline{a}}\overline{bccc} \\
&\Rightarrow aa\overline{aB}Bbccc \\
&\Rightarrow aaa\underline{B\overline{b}}\overline{bccc}
\end{aligned}
$$

$$\Rightarrow aaa\overline{bb}bccc$$

از سوی دیگر اشتقاق زیر یک اشتقاق چپ برای $a^3b^3c^3$ در G است زیرا قواعد تولیدی شبیه شکل نشان داده شده در ۱-۲-۲ استفاده می شود. تنها تفاوت در شاخص هـای ٦ و۷ است که باید تغییر کنند.

$$\underline{S} \Rightarrow \overline{ab\underline{sc}}$$
$$\Rightarrow \underline{aB\overline{a}}\overline{Bscc}$$
$$\Rightarrow a\overline{aB}B\underline{S}cc$$
$$\Rightarrow aaB\underline{B}\overline{\underline{a}bccc}$$
$$\Rightarrow aa\overline{aB}Bbccc$$
$$\Rightarrow aaa\underline{B}\overline{\underline{b}}bccc$$
$$\Rightarrow aaa\overline{bb}bccc$$

سلسله مراتب گرامرها :

کلاسهای زیر از گرامرها بوسیله ی افـزایش تـدریجی محـدودیت کـه قواعـد تولیـدی مجبورند آنها را اجرا کنند بدست آورده شده اند.

یک گرامر نوع ۱ یک گرامر نوع صفر<N, Σ, P, S> است که در دو شـرط زیـر صـدق می کند :

الف : هر قاعده ی تولیـدی $\alpha \to \beta$ در P اگـر بـه فـرم $S \to \varepsilon$ نباشـد بـه صـورت $|\alpha| \le |\beta|$است.

ب : اگر $S \to \varepsilon$ از P آنگاه S درسمت چپ هیچ قاعده ی تولیدی ظاهر نمی شود.

یک زبان ، زبان نوع ۱ نامیده می شود اگر یک گرامر نوع ۱ وجود داشته باشد کـه زبـان را بوجود بیاورد.

مثال ۱-۲-۱۵ : گرامر مثال ۱-۲-۸ یک گرامر نوع ۱ نیست زیرا شرط b را ارضا نمـی کند. گرامر می تواند با جایگزینی قواعد تولیدی اش با یکی از قواعد زیر به گرامر نـوع ۱ تبدیل شود. E یک نماد غیر پایانی فرض می شود.

$S \to E$
$\to \varepsilon$
$E \to aBEC$
$\to abc$
$Ba \to aB$
$Bb \to bb$

بعلاوه تبدیل گرامر یک قاعده ی تولید به فرم $Bb \to b$ یک گرامر نوع ۱ نخواهد بود چون شرط a را نقض می کند.

یک گرامر نوع ۲ در هر قاعده ی تولید $\alpha \to \beta$ که در شرط $|\alpha|=1$ صدق می کند یک گرامر نوع ۱ است. یعنی α یک نماد غیر پایانه است. یک زبان ، زبان نوع ۲ نامیده می شود اگر گرامر نوع دو وجود داشته باشد که آن را بوجود بیاورد.

مثال ۱-۲-۱۶ : گرامر مثال ۱-۲-۷ یک گرامر نوع ۱ نیست و بنابراین یک گرامر نوع ۲ هم نیست. این گرامر می تواند با جایگزینی قواعد تولیدی اش با یکی از قواعد زیر به گرامر نوع ۲ تبدیل شود. E یک نماد غیرپایانه فرض می شود.

$S \to \varepsilon$
$\to E$
$E \to aEb$
$\to ab$

بعلاوه یک قاعده ی تولیدی به فرم $aE \to EaE$ یک گرامر نوع ۲ را نتیجه نخواهد داد.یک گرامر نوع ۳ یک گرامر نوع ۲ $< N, \Sigma, p, s >$ است در هریک از قواعد تولیدی $\alpha \to \beta$ که به فرم $S \to \varepsilon$ نیست و در یکی از شرایط زیر صدق می کند.

الف : β یک نماد پایانه است.

ب : β یک نماد پایانه است که یک نماد غیر پایانه از آن پیروی می کند.

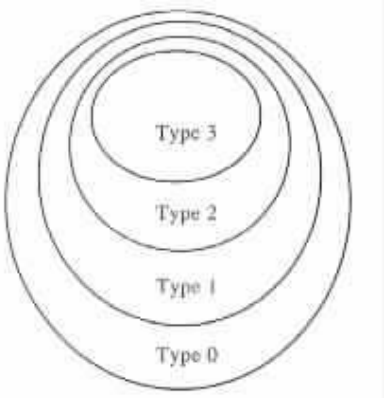
یک زبان نوع ۳ نامیده می شود اگر یک گرامر نوع ۳ که آن را بوجود می آورد وجود داشته باشد.

مثال ۱-۲-۱۷ : گرامر $< N, \Sigma, p, s >$ که قواعد دستوری زیر را دارد یک گرامر نوع۳ است .

$S \to \varepsilon$
$\to aA$
$\to bB$
$\to b$
$A \to bB$
$\to b$
$B \to aA$
$\to bB$
$\to b$

بعلاوه یک قاعده ی تولید به فرم $A \to Ba$ یا به فرم $B \to bb$ یک گرامر نوع ۳ را نتیجه می دهد.

شکل ۱-۲-۳ سلسله مراتب انواع مختلف گرامرها را نشان می دهد.



شکل ۱-۲-۳ : سسلسله مراتب گرامرها

۱-۳ برنامه ها

وابستگی عمیق ما در پردازش اطلاعات ، درباره قرارگیری برنامه ها در یک صف افزایشی از کاربردها آورده شده است . برنامه ها میتوانند در خانه ، کار ، تجارت ، بیمارستان و مدرسه ها ایجاد شوند . آنها برای یاد گرفتن و بازی کردن بازیها و حروفچینی و مرتب کردن تماسهای تلفنی و مهیا کردن خدمات تشخیصی و پیش بینی کردن آب و هوا و پرواز هواپیماها و بسیاری اهداف دیگر استفاده می شوند.

برای آسانتر کردن وظیفه نوشتن برنامه ها برای بسیاری از کاربردهای مختلف، زبانهای برنامه نویسی بسیاری توسعه داده شده اند . تنوع زبانهای برنامه نویسی تفاسیر متفاوتی را که میتوانند بر حسب اطلاعات داده شوند منعکس می کند . به هر حال از لحاظ قدرتشان در بیان نتایج اندکی میانشان تفاوت وجود دارد . در نتیجه ، در مطالعه برنامه ها میتوان از زبانهای برنامه نویسی متفاوت استفاده نمود.

حتی اگر از زبانهای برنامه نویسی ثابتی استفاده کنیم مطالعه برنامه ها مفید خواهند بود . این مطالعه بحث راجع به برنامه ها را امکان پذیر میکند . انتخاب باید برای یک زبان که به اندازه کافی کلی و برای همه برنامه ها مناسب است باشد ، اما برای ساده کردن بحث باید به کاملا پایه ای و مقدماتی باشد.

انتخاب یک زبان برنامه نویسی

در اینجا یک برنامه بعنوان یک دنباله متناهی از دستورات روی چندین دامنه D تعریف شده است دامنه D دامنه متغییرها نامیده می شود که فرض می شود مجموعه ای از عناصر است با یک عنصر برجسته که اولین مقدار متغییرها نامیده می شود . هر یک از عناصر D فرض می شود یک انتقال ممکن از مقدار به متغییرهای برنامه باشد.

الف : فرم دستورات خواندن جاییکه X متغییر است .

read x

ب : فرم دستورات نوشتن جاییکه X متغییراست

write x

ج : شکل دستورات انتقال قطعی جاییکه x1,….,xm و y متغییر ها هستند و f یک تابع از D به توان M به D است.

$y := f(x_1, \ldots, x_m)$

د : شکل دستورات If شرطی وقتیکه I یک دستور و x1,....xm متغییرها هستند و q یک گزاره از D به توان m به {false,true} است .

if $Q(x_1, \ldots, x_m)$ then I

e. ه : شکل دستورات حلقه زنی قطعی جاییکه آلفا یک دنباله ناتهی از دستورات است و x1,....xm متغییرها هستند و q یک گزاره از D به توان m به {false,true} است .

do

α

until $Q(x_1, \ldots, x_m)$

و : شکل پذیرش دستورات شرطی

if eof then accept

ز : شکل دستورات عدم پذیرش

Reject

ک : شکل دستورات انتقال غیر قطعی وقتی که X متغیر است .

x := ?

ل : شکل دستورات حلقه زنی غیر قطعی وقتیکه2=<Kوهر یک ازالفاها دنباله ناتهی از دستورات هستند و x1,....xm متغییرها هستند و q یک گزاره از D به توان m به {false,true} است .

do
a_1
or
a_2
or
⋮
or
a_k

until $Q(x_1, \ldots, x_m)$

در هر برنامه دامنه D از متغیرها فرض می شود که ۱ نماینده روی چندین الفبا دارد . برای مثال D میتواند مجموعه ای از اعداد طبیعی ، اعداد صحیح و هرمجموعه متناهی از عناصر باشد . به نظر می رسد تابع f و گزاره Q از یک مجموعه پیش ساخته از توابع وگزاره های قابل محاسبه باشند.

در ادامه مطلب ، دامنه متغیرها وقتیکه به طور طبیعی اهمیت کمی دارد صریحا آشکار نخواهد بود . به علاوه عبارات در نمادهای میانوندی برای مشخص کردن توابع و گزاره ها استفاده خواهند شد.

برنامه هایی که فاقد دستورالعملهای غیرقطعی هستند برنامه های قطعی نامیده می شوند و برنامه هایی که از دستورالعمل های غیرقطعی استفاده میکنند برنامه های غیر قطعی نامیده می شوند.

مثال ۱-۳-۱ : برنامه P1 در شکل ۱-۳-۱ را در نظر بگیرید. (a) یک مثال از برنامه قطعی است و برنامه P۲ یک مثال از برنامه غیر قطعی است . مجموعه اعداد طبیعی با صفر به عنوان مقدار برجسته برای دامنه متغییرها فرض می شود.

read x do

```
y := 0                    read x
z := 1                  or
do                                y := ?
y := y + 1              write y
z := z + 1              until y = x
until z = x             if eof then accept
read y
if eof then accept
reject
```

(a) (b)

شکل ۱-۳-۱

(b)یک برنامه غیر قطعی (a)یک برنامه قطعی

برنامه P1 از سه متغییر به نامهای Z،Y،X استفاده می کند . دو تابع در این برنامه وجود دارند . تابع ثابت f1 ()=0 و تابع یگانی f2(n)=n+1 بوسیله یک افزایش می یابد . دستورالعملهای حلقه در برابری شرطهای دودویی استفاده می شوند.

برنامه P2 از دو دستور غیر قطعی استفاده میکند . یک دستور غیر قطعی ، یک دستور انتقال در فرم "y := ?" دیگری یک دستور در فرم "do...or..until..." است . یک ورودی از برنامه داده شده یک دنباله از عناصر دامنه متغییرهای برنامه است . هر عنصر ورودی برنامه ، مقدار ورودی نامیده می شود.

مثال ۱-۳-۲ : برنامه هایی از مثال ۱-۳-۱ میتواند یک ورودی که یک دنباله متناهی از اعداد طبیعی است داشته باشند . یک ورودی به فرم "۲،۳،٤،۱" شامل چهار مقدار ورودی است و هر ورودی در فرم "" شامل هیچ مقدار ورودی نیست."۱،۲،۳،.." نمی تواند ورودی برای برنامه ها باشد زیرا این دنباله متناهی نیست.

یک اجرای متوالی از برنامه معین، اجرا دستورات روی یک ورودی معین مبتنی بر معنایشان است .دستورات متوالیا اجرا میشوند . با اولین دستور شروع می شوند و متغیرهای اولیه ، مقدار نخست متغیرها را نگه می دارند.

برنامه های قطعی

یکی از خواص برنامه های قطعی این است که به ازای هر تعداد تکرار برنامه با یک سری داده های ورودی نتایج ثابت و درست در خروجی ظاهر میگردد. هر دستور از برنامه قطعی اعمالی را که انجام میشوند کاملا مشخص می کند ، در مقابل دستورهای غیر قطعی فقط جزئی از اعمال را مشخص می کنند.

یک اجرا از دستور خواندن read X ، مقدار بعدی ورودی را در X ذخیره میکند و یک اجرا از دستور نوشتن writeX مقدار X را مینویسد

دستورهای انتقال قطعی و if شرطی قاعده معنایی دارند.

یک اجرا از دستور حلقه do α until $Q(x_1,\ldots,x_m)$ عبارت است از اجرای مکرر و چک شدن مقدار $Q(x_1,\ldots,x_m)$. اجرای دستور حلقه با مشخص شدن اینکه گزاره $Q(x_1,\ldots,x_m)$ مقدار صحیح دارد خاتمه می یابد.اگر $Q(x_1,\ldots,x_m)$ شامل true است پس فقط یک بار تکرار اجرا می شود . از طرف دیگر اگر $Q(x_1,\ldots,x_m)$ شامل false بود برای همیشه حلقه ادامه می یابد ،مگر اینکه اجرا در الفا خاتمه یابد.

بعد از اینکه همه ورودیها مصرف شدند یک دستور پذیرش شرط سبب توقف اجرای میگردد ،که در این حالت شرط رسیدن به انتهای فایل است . در غیر اینصورت اجرای دستورالعملها باعث اجرای دستور العمل بعدی به ترتیب خواهد شد . البته اجرای دستورات با رسیدن به دستور reject یا رسیدن به آخرین کد

دستورالعمل یا محاسبه متغیری خارج از دامنه و محدوده یا رسیدن به انتهای فایل ورودی نیز متوقف میشود .

مثال ۱-۳-۳ : دو برنامه در شکل ۱-۳-۲ را در نظر بگیرید

```
do                          do
if eof then accept          read value
read value                  write value
write value                 until value < 0
until false                 if eof then accept
```

(a) (b)

شکل ۱-۳-۲ دو برنامه قطعی

در نظر بگیرید ورودیهای برنامه مجموعه ای از اعداد طبیعی است و با صفر به عنوان مقدار اولیه شروع می شود.

برای هر ورودی برنامه در شکل ۱-۳-۲ (a) یک روند اجرا وجود دارد در هر روند اجرا برنامه یک خروجی که برابر ورودی است را ایجاد می کند . همه روندهای اجرا از برنامه ، با اجرای دستور پذیرش شرط خاتمه می یابند.

در ورودی"۱،۲" روند اجرا مکررا برای ۳ بار بدنه دستور حلقه را اجرا می کند . در طول اولین تکرار روند اجرا مشخص می کند که گزاره eof مقدار false دارد . بنابر این دنباله اجرا دستور پذیرش را نادیده می گیرد و بوسیله خواندن مقدار ۱ و نوشتن آن در خروجی ادامه می یابد . در طول دومین تکرار ، روند اجرا مشخص می کند که هنوز انتهای ورودی نرسیده است و سپس روند اجرا مقدار ۲ را می خواند و در خروجی می نویسد . در طول سومین تکرار بعد از مشخص شدن مقدار true برای گزاره eof روند اجرا خاتمه می یابد.

روند اجرای برنامه در شکل ۱-۲-۳ (b) در دستور پذیرش شرط فقط در ورودیهایی که با یک مقدار منفی پایان می یابند متوقف می شوند و در هیچ جای دیگر مقدار منفی ندارند .(به عنوان مثال ترتیب ورودی ۱ و ۲ و ۳-) در ورودیهایی که هیچ مقدار منفی ندارند روند اجرای برنامه با سعی کردن برای خواندن دستور العمل بعد از انتهای برنامه متوقف می شود . در ورودیهایی که مقدار منفی قبل از انتهایشان دارند روند اجرا برنامه ها با انتقال کنترل به بعد از دستور العمل انتهایی برنامه متوقف می شوند.

به طور واضح یک پذیرش می تواند به عنوان دستور خاتمه بررسی شود که اجرای موفقیت آمیز اجرای برنامه را مشخص می کند ، در حالیکه پذیرش فقط می تواند بعد از آخرین ورودی که بدست آمده است اجرا شود . به طور مشابه یک عدم پذیرش می تواند به عنوان دستور خاتمه که اتمام ناموفق اجرای برنامه را مشخص می کند، بررسی شود.

نیازی را که دستورات پذیرش بعد از خواندن همه مقادیر ورودی اجرا می کنند نباید مشکلی ایجاد کنند زیرا هر برنامه میتواند برای مناسب شدن این شرایط تغییر کند . علاوه بر این چنین محدودیتی به نظر طبیعی می رسد زیرا هر برنامه مجبور است که همه مقادیر ورودی را قبل از علامتگذاری موفق بوسیله یک دستور پذیرش چک کند .به طور مشابه درخواستی را که یک روند اجرا با خواندن از میان یکی از ورودیهای آخر باید خاتمه دهد به نظر طبیعی می رسد . این مهم نیست که دستور خواندن متغیر بعدی , در حال اجرای دستور خواندن است یا چک کردن برای شرط رسیدن به انتهای فایل (eof)

باید توجه شود که گزاره $Q(x_1,\ldots,x_m)$ در دستورات شرطی if و دستورات حلقه نمی تواند یکی از اشکال eof باشد . گزاره ها فقط در رابطه با مقادیر متغیرهای X1…Xm تعریف می شوند نه در رابطه با ضوابط ورودی.

محاسبات

برنامه ها دنباله های متناهی از دستورات را برای توضیح دادن مجموعه هایی از تعداد نامتناهی محاسبات بکار می برند . توضیحات محاسبات به وسیله عدم گردش دنباله هایی از دستورات در روند اجرا بدست می آیند .در مورد برنامهای قطعی هر دنباله اجرا یک توضیح برای محاسبه فراهم میکند . از طرف دیگر در مورد برنامه های غیر قطعی برخی از دنباله های اجرا ممکن است به عنوان محاسبات مطرح شده باشند در حالیکه انواع دیگر ممکن است غیر محاسباتی مطرح شده باشند . برای مشخص کردن این تمایزها به تعاریف زیر نیاز داریم .

یک روند اجرا محاسبه پذیر نامیده میشود اگر با یک دستور پذیرش خاتمه یابد . یک روند اجرا محاسبه ناپذیر یا محاسبه مردود نامیده می شود اگر ورودیی وجود دارد که به ازای آن روند محاسبه پذیر نیست . یک روند اجرا چه محاسبه پذیر باشد که محاسبه ناپذیر محاسباتی نامیده میشود .یک محاسبه محاسبه توقفی نامیده میشود اگرروند اجرا متناهی باشد.

مثال ۱-۳-٤ : برنامه شکل ۱-۳-۳ را در نظر بگیرید

```
read value
do
write value
value := value - 2
until value = 0
if eof then accept
```

شکل ۱-۳-۳ یک برنامه معین

در نظر بگیرید ورودیهای برنامه مجموعه ای از اعداد طبیعی است و با صفر به عنوان مقدار اولیه شروع می شود.

دریک ورودی که عبارتست ازیک صحیح مثبت ، زوج و تنها , برنامه یک دنباله اجرا که محاسبه پذیر است دارد. (به عنوان مثال در ورودی"٤")
در یک ورودی که شامل بیش از یک مقدار است که با یک عدد صحیح زوج مثبت شروع میشود روند برنامه یک اجرای توقفی دارد که محاسبه ناپذیر است . (به عنوان نمونه اعداد ٤ و ٣ و ٢)
در این مثال با ورودی ٣ در دومین اجرای حلقه , به عدد ١ می رسیم که باعث منفی شدن عبارت و در نتیجه تکرار نامحدود حلقه خواهد شد بنابر این محاسبات ناپذیراست .
یک ورودی پذیرفته شده یا تشخیص داده شده به وسیله یک برنامه نامیده میشود اگر برنامه یک محاسبه پذیرفتنی در چنین ورودی داشته باشد . در غیر این صورت ورودی پذیرفته نشده یا مردود شده بوسیله برنامه نامیده میشود .

در صورتیکه یک برنامه به ازای ورودی X محاسبه پذیر بوده و خروجی Y داشته باشد گفته خروجی y در ازای ورودی x دارد . خروجیهای محاسبات ناپذیرفتنی تعریف نشده در نظر گرفته میشوند حتی اگر چنین محاسباتی قادر به اجرا کردن دستور نوشتن باشند.

مثال ١-٣-٥ : برنامه مثال ١-٣-٤ ورودیهای "٢" و "٤" و "٦" و.... را می پذیرد . در ورودی "٦" برنامه خروجیهای "٢و٤و٦" دارد و در ورودی "٢" برنامه خروجی "٢" دارد . برنامه ورودی های "١" و "٠" و "٢و٤" را نمی پذیرد . برای این ورودی ها برنامه خروجی ندارد ، خروجیها تعریف نشده هستند .

یک محاسبه غیرقطعی نامیده میشود اگر شامل اجرای یک دستور غیر قطعی است . در غیر این صورت محاسبه ، محاسبه قطعی نامیده می شود.
برنامه های غیرقطعی
اهداف متفاوت، نیاز برای دستورات غیر قطعی را در زبانهای برنامه نویسی ایجاد میکنند .یکی از اهداف اجازه دادن به برنامه ها برای سر و کار داشتن با مسائلی که بیش از یک راه حل دارند است . درچنین مواردی دستورهای غیرقطعی یک روش طبیعی

انتخاب را فراهم می کنند . هدف دیگر ساده کردن وظیفه برنامه نویسی است . هدف دیگر فراهم کردن ابزاری برای شناخت مسائل مشکل و مطالعه کردن انواع محدود شده ی برنامه ها است .

انجام ملاحظات نباید خواننده را در این موضوع به دردسر بیندازد. یک نفرمعمولا معنای زبانهای برنامه نویسی را قبل از علم آن را فرا میگیرد و اگر یک نفر همیشه پیاده سازی چنین زبانهایی را انجام میدهد. در ادامه نشان داده خواهد شدکه چطور یک برنامه غیرقطعی می تواند به یک برنامه قطعی برگردانده شود که یک تابع وابسته را حساب می کند .

دستورهای غیرقطعی ،دستورهای اساسی هستند که میتوان آنها را از بین چندین حالت انتخاب کرد. با اینکه روزانه در زندگی انتخابهایی انجام میدهیم ،اما وجود دستوراتی در زبانهای برنامه نویسی که بتوانیم بوسیله انها انتخابهایی انجام دهیم عجیب به نظر می رسد.

معنای دستورحلقه غیر قطعی ازفرم(do x_1 or x_2 or…or x_k until $Q(x_1,\ldots,x_m)$ شبیه دستورحلقه زنی قطعی ازفرم (do x until $Q(X_1,\ldots,X_3)$ است. تنها تفاوت این است که در یک نمونه قطعی یک شبه کد ثابت x در هر بار تکرار اجرا می شود درحالیکه در نمونه های غیرقطعی ،یک شبه کد اختیاری در فرم $x_1,\ldots x_k$ در هر تکرار اجرا میشود .انتخاب یک شبه کد میتواند در یک تکرار با تکرار دیگر متفاوت باشد .

```
counter := 0
/* .Choose five input values */
do
read value
or
read value
write value
counter := counter + 1
until counter = 5
/*Read the remainder of the input */
do
if eof then accept
read value
```

until false

مثال ۱-۳-۶ : برنامه شکل ۱-۳-۴ غیر قطعی است . در نظر بگیرید ورودیهای برنامه مجموعه ای از اعداد طبیعی است و با صفر به عنوان مقدار اولیه شروع می شود.

برنامه در ورودی "۱،۲،۳،٤،٥،٦" یک دنباله اجرا به فرم زیر دارد. دنباله اجرا با یک تکرار از دستور حلقه غیر قطعی درحالیکه اولین شبه کد انتخاب شده است شروع می شود. اجرای شبه کد شامل خواندن مقدار ورودی ۱ است تا زمانیکه چیزی نوشته نشده و counter با مقدار ۰- مانده است. سپس دنباله اجرا با ۵ تکرار اضافی از دستور حلقه غیرقطعی ادامه می یابد. در هر یک از تکرار های اضافی، دومین شبه کد انتخاب شده است . هر اجرا از دومین شبه کد یک مقدار را در ورودی میخواند ، خروجی های مقداری که خوانده شده است و مقدار counter بوسیله یک افزایش می یابد. وقتیکه counter به مقدار ۵ رسید ، دنباله اجرا از اولین دستورحلقه زنی خارج میشود .در طول اولین تکرار از دومین دستور حلقه ،دنباله اجرا با اجرای دستور پذیرش شرطی خاتمه می یابد . دنباله اجرا یک محاسبه پذیرفتنی با خروجی "۲،۳،٤،٥،٦, " است.
برنامه در ورودی "۱،۲،۳،٤،٥،٦," چهار دنباله اجرای اضافی شبیه بالا دارد. تنها تفاوت این است که دنباله اجراهای اضافی به جای نادیده گرفتن مقدار "۱" ، مقادیر ورودی ۲،۳،٤،٥, را به ترتیب نادیده می گیرد. یک دنباله اجرا مقدار ورودی ، بوسیله انتخاب کردن و خواندن مقدار در اولین شبه کد از دستورات حلقه زنی غیر قطعی را نادیده میگیرد (یعنی دنباله اجرا مقدار ورودی را چون حق انتخاب بین مقادیر مختلف دارد می تواند نادیده بگیرد و در هر اجرا مقدار متفاوت شود) دنباله های اجرای اضافی محاسبات پذیرفتنی به ترتیب با خروجیهای "۱،۳،٤،٥،٦" و "۱،۲،۳٤،٥ " و "۱،۲،۳،٤،٦" هستند.

برنامه در ورودی "۱،۲،۳،٤،٥،٦" نیز یک محاسبه پذیرفتنی به فرم زیر دارد. محاسبه با

۵ تکرار از اولین دستور حلقه زنی آغاز میشود .در هر یک از تکرارها دومین شبه کد دستور حلقه زنی غیر قطعی اجرا شده است .در طول هر تکرار ، یک مقدار ورودی خوانده می شود که مقدار در خروجی نوشته شده است.و مقدار counter یکی افزایش می یابد .بعد از ۵ تکرار از دستور حلقه غیرقطعی مقدار counter به ۵ می رسد و محاسبه به دستور حلقه قطعی منتقل میشود. محاسبه، مقدار ٦ را در طول اولین تکرار از دستور حلقه قطعی می خواند و در طول دومین تکرار خاتمه میابد. خروجی محاسبه "۵،٤،۳،۲،۱," است.

برنامه به تعداد ۱٤-$۲^{۷}$بار دستور با ورودی "۱،۲،۳،٤،۵،٦" اجرا میکند که محاسباتی نیستند. ۷-$۲^{٦}$تا از این دنباله های اجرا با سعی در خواندن انتهای ورودی بوسیله ی اولین دستور خواندن خاتمه می یابند. ۷-$۲^{٦}$تا از این دنباله های اجرا با سعی کردن خواندن انتهای ورودی با دومین دستور خواندن خاتمه می یابند. در هر یک از این دنباله های اجرا حداقل دو مقدار ورودی بدلیل مصرف شدن مقادیر ورودی در اولین قسمت کد از دستورحلقه غیرقطعی نادیده گرفته میشوند. تفاوت دنباله های اجرا در مقادیر ورودی است که آنها برای نادیده گرفتن انتخاب می کنند.
هیچ یک از دنباله های اجرای برنامه در ورودیهای "۱،۲،۳،٤،۵،٦" یک محاسبه ناپذیرفتنی نیست زیرا برنامه در چنین ورودی یک محاسبه پذیرفتنی دارد.
برنامه ورودی "۱،۲،۳،٤" را نمی پذیرد. در چنین ورودی برنامه $۲^{۵}$دنباله اجرا دارد که همه محاسبات ناپذیرفتنی هستند.

اولین دستور حلقه غیر قطعی برنامه برای انتخاب مقادیر خروجی از ورودی ها استفاده میشود. روی انتخاب کردن ۵ مقدار دنباله های اجرا با مصرف کردن بقیه ورودیها در دومین دستور حلقه زنی قطعی ادامه می یابند .
در ورودیهایی با کمتر از ۵ مقدار ، روند اجرا در اولین دستور حلقه غیر قطعی بخاطر سعی برای خواندن انتهای ورودی خاتمه می یابند .
متغییر counter تعداد مقادیر انتخاب شده در مراحل، در طول هر دنباله اجرا را نگه میدارد.
یک برنامه قطعی دقیقا یک دنباله اجرا روی هر ورودی دارد و هر دنباله اجرا از یک

برنامه قطعی یک محاسبه است. ازطرف دیگر آخرین مثال نشان میدهد که یک برنامه غیر قطعی ممکن است بیش از یک دنباله اجرا در یک ورودی معین داشته باشد و بعضی از دنباله های اجرا ممکن است محاسباتی از برنامه نباشند .

دستورهای حلقه غیرقطعی نشان داده شده اجازه ی انتخابها بین سگمنت کدها را میدهند. انگیزه برای مطرح کردن دستورهای انتقال غیر قطعی اجازه دادن انتخاب ها بین مقادیر است. مخصوصاً یک دستور انتقال غیر قطعی از فرم x:=? به x یک مقدار دلخواه از دامنه متغییر ها نسبت می دهد انتخاب مقدار نسبت داده شده میتواند با مواجهه دستور با یک مقدار دیگر تغییر کند.

مثال ۱-۳-۷ : برنامه در شکل ۱-۳-۵ غیر قطعی است. مجموعه اعداد طبیعی به عنوان دامنه ی متغییرها فرض شده است مقداراولیه صفر فرض میشود.

```
❊/Nondeterministically find a value that
a. appears exactly once in the input, and
b. is the last value in the input/❊                    .
last? =:
write last
❊/Read the input values, until a value
equal to the one stored in last is reached     .
/❊
do
    read value
until value = last
❊/Check for end of input/❊ .
if eof then accept
reject
```

شکل ۱-۳-۵ : یک برنامه نامعین برای تعیین نمایش آخرین مقدار ورودی

برنامه یک ورودی را می پذیرد اگر و تنها اگر آخرین مقدار ورودی در هیچ جای دیگر از ورودی از ورودی ظاهر نشده باشد، چنین مقداری همچنین خروجی یک محاسبه پذیر است. برای مثال در ورودی "۱،۲،۳" برنامه خروجی"۳ " دارد. از طرف دیگر در ورودی "۱،۲،۱," هیچ خروجی تعریف نشده است چون برنامه ورودی را نپذیرفته است.

در هر ورودی، برنامه روند اجرای نامحدود دارد. هر روند اجرا مطابق با انتقال یک

مقدار متفاوت به last از دامنه متغییرهای برنامه است.
با انتقال مقداری به last که در ورودی ظاهر شده است اجرا از دستور حلقه خارج میشود. دراین صورت یکی ازحالات زیر اتفاق می افتد :

۱)دنباله اجرا یک محاسبه پذیرفتنی است اگر مقدار انتقال یافته به last فقط در انتهای ورودی ظاهر شده باشد.

۲)دنباله اجرا یک محاسبه ناپذیرفتنی است اگر مقدار انتهایی ورودی ، بیش از یکبار در ورودی ظاهر شود.بعنوان مثال داده های ورودی ۱و۲و۱ و مقدار ۱ در last

۳) اگر نه حالت ۱ و نه حالت ۲ اتفاق نیافتدروند اجرا محاسبه پذیر نیست.به عنوان مثال نسبت دادن یک عدد طبیعی به جز ۱ و۲ و ۳ به last در صورتیکه دنباله ورودی "۱و۲و۳ " باشد.

یک انتقال به last از مقادیری که در ورودی ظاهر نشده اند دلایل یک دنباله اجرا را در صورت حلقه زنی به خاطر سعی کردن خواندن آنسوی انتهای ورودی خاتمه میدهد .با چنین انتقالی یکی از حالات زیر پیش خواهد آمد:

۱)دنباله اجرا محاسبه ناپذیرفتنی است اگر مقدار انتهای ورودی ،بیش از یکبار در ورودی ظاهر شود.

۲)دنباله اجرا محاسبه ناپذیرفتنی است اگر ورودی تهی است .

۳)دنباله اجرا محاسبه نیست اگر نه ۱ و نه۲ نگهداشته نشوند.

به طور استدلالی هر برنامه در هر ورودی دنباله های اجرای خوب و دنباله های اجرای بد را تعریف می کند. دنباله اجراهای خوب با دستورات پذیرش خاتمه می یابند و دنباله های اجرای بد با پذیرش دستورها خاتمه نمی یابند. بهترین دنباله های اجرا برای یک ورودی معین ،محاسباتی هستند که برنامه در ورودی دارد. اگر انجام دنباله های اجرای خوبی وجود دارد مجموعه محاسبات با آن مجموعه شناخته شده است .در غیر این صورت مجموعه محاسبات با مجموعه ای از دنباله اجراهای بد شناخته شده است . محاسبات برنامه در یک ورودی معین یا همه محاسبات پذیرفتنی یا همه محاسبات ناپذیرفتنی هستند. بعلاوه اغلب محاسبات ناپذیرفتنی ممکن است هیچ وقت خاتمه نیابند .در ورودیهایی که پذیرفته شده هستند ،ممکن است برنامه دنباله اجراهایی که محاسباتی نیستند داشته باشند. در غیر این صورت ورودیهایی که پذیرفته شده نیستند همه دنباله های اجرا محاسباتی هستند .

معنای هر برنامه بوسیله محاسبات برنامه مشخص می شود. در مورد برنامه های قطعی معنای یک برنامه معین مستقیماً به معنای دستوراتش وابسته است. هر اجرا از دستورات برنامه را در دوره ای از محاسبه نگه می دارد .

در مورد برنامه های غیرقطعی بین دنباله های اجرا و محاسبات تمایز وجود دارد و بنابر این معنای یک برنامه معین فقط در یک روش محدود شده به معنای دستوراتش وابسته است .اگر چه هر محاسبه میتواند بوسیله اجرای دستورات انجام شود بعضی از دنباله های اجرا با هیچ محاسبه مطابقت ندارند دلیل چنین چیزی توانایی دستورات غیر قطعی در ایجاد اختیار انتخاب ها است .

هر برنامه میتواند بعنوان دارنده یک فاعل خیالی با قدرت سحرآمیز که برنامه را اجرا میکند در نظرگرفته شود.در یک ورودی معین ،وظیفه فاعل موهومی دنبال کردن هر یک ازمحاسباتی است که برنامه روی ورودی دارد، درموردبرنامه های قطعی به عنوان یک نمونه کوچکتر و محدود شده می تواند مطرح شده باشد که فاعل با نداشتن استقلال مخالف است. نتیجه اجرای هردستور قطعی برای فاعل بواسطه معنای دستور کاملاً معین است ، از طرف دیگر وقتی یک دستور غیرقطعی اجرا میشود فاعل باید نه تنها معنای موضعی دستور را ادا کند ،بلکه همچنین هدف سراسری از دسترسی به یک دستور پذیرش هر وقت هدف سراسری قابل انجام است را نیز باید ادا کند.

به خصوص، معنای موضعی دستور حلقه زنی غیرقطعی ازفرم (do x_1 or..or x_k until b $Q(x_1,.... x_m$ نیاز دارد که در هر تکرار دقیقاً یکی از شبه کدهای $x_1...x_k$ در یک الگوی اختیاری بوسیله فاعل انتخاب شود. معنای سراسری برنامه نیاز دارد که انتخاب برای یک شبه کد صورت بگیرد که میتواند دنباله اجرا را به موعد توقفش و به یک دستور پذیرش شرط سوق دهد، هر وقت چنین چیزی ممکن است .

متشابهاً معنای موضعی ازیک دستور انتقال غیر قطعی از فرم x=? نیاز دارد که هر مقدار نسبت داده شده ی x بوسیله ی فاعل ، طبق یک فرم اختیاری از دامنه متغییرها انتخاب شود. معانی سراسری برنامه نیاز دارد که انتخاب برای یک مقدار که متوقف می کند دنباله اجرا را با یک دستور پذیرش شرط صورت بگیرد. هر وقت چنین چیزی ممکن است.

بواسطه بحث بالا فهمیده می شود که مشی"اولین حدس یک راه حل وسپس چک کردن برای صحت آن "می تواند وقتی که یک برنامه نوشته می شود استفاده شود.

چنین مشیی وظیفه برنامه نویس را آسانتر میکند زیرا چک کردن صحت یک راه حل ساده تر از حل ریشه ای مسئله است. در چنین موردی ،بار قطعیت یک حدس صحیح بر فاعل اجرای محاسبات تحمیل میشود.

روی این نکته باید تاکید شود که از دید فاعل یک حدس صحیح است، اگر و تنها اگر منجر به یک دنباله اجرا در طول محاسبه از برنامه شود. فاعل هیچ چیز راجع به مساله ای که برنامه قصد دارد حل کند نمیداند. تنها چیزی که فاعل را پیش میبرد تفاوت هدف دسترسی، اجرای دستور پذیرش شرط در انتهای ورودی است. بنابر این زود است برای برنامه نویس که کاملاً معین کند فشاری را که باید بوسیله حدس صحیح برآورده شود.

```
/* Guess the output value. */
x := ?
write x
/* Check for the correctness of the
guessed value.                    */
do
    if eof then accept
    read y
until y = x
```

شکل ۱,۳,٦ یک برنامه غیر قطعی که یک مقدار غیر ورودی را به خروجی می دهد .

مثال ۱,۳,۸ :توجه کنید به شکل ۱,۳,٦ خروجیهای یک مقدار که در ورودی ظاهر نشده است. برنامه هر محاسبه را به وسیله حدس زدن یک مقدار و نگه داشتن آن در x شروع می کند. سپس برنامه مقدار ورودی را می خواند و چک میکند که آیا مقدار ورودی متفاوت از مقدار نگهداری نشده در x است یا نه.

اطلاع از یک فاعل موهومی یک راه تقاضا برای توضیح دادن عدم قطعیت آماده میکند. معهذا از اطلاع باید با احتیاط استفاده شود و از اطلاع های غلط اجتناب شود. بویژه یک فاعل موهومی فقط در برنامه های کامل باید بکارگرفته شود .اجازه تعریف فضا (محل) برای فاعل موهومی که در یک فاعل بکار گرفته می شود وجود ندارد برای مثال یک فاعل موهومی که داده شده در برنامه p در مثال زیر نمی تواند بوسیله ی فاعلهای دیگرکه پیش میروند با پذیرش درست همان ورودی هایی که فاعل نمی پذیردبکارگرفته شود.

```
sum1 := 0
sum2 := 0
do
    if eof then accept
    do                  /* Guess where the next input
value belongs. */
        read x
        sum1 := sum1 + x
    or
        read x
        write x
        sum2 := sum2 + x
    until sum1 = sum2         /* Check for the
correctness of the
                    guesses, with respect to the portion
                    of the input consumed so far.
                */
until false
```

شکل ۱,۳,۷ یک برنامه غیر قطعی برای تقسیم بندی ورودی به مجموع عناصر .
مثال ۱,۳,۹توجه کنید به برنامه p در شکل ۱,۳,۷، در یک ورودی معین ، خروجیهای p یک انتخاب اختیاری از مقادیر ورودی است که مجموعشان برابر مجموع مقادیر ورودیهای انتخاب نشده است .
مقادیر همان ترتیب نسبی در خروجی را دارند که در ورودی هست. (یعنی ترتیب نسبی مقادیرخروجی ، مثل ترتیب نسبی مقادیر در ورودی است.)
برای مثال در ورودی "۲و٤و۳و۱و۲" خروجیهای ممکن "٤و۲ "و"۲و٤ "و"۲و۳و۱ "و "۳و۱و۲ " باشند. از طرف دیگر هیچ خروجی برای ورودی "۳و۲ " تعریف نشده است .

در هر تکرار از دستور حلقه زنی تودرتو ، برنامه حدس میزند که آیا مقدار ورودی بعدی در میان یکی از انتخاب شده ها هست یا نه.اگر انتخاب شده است قدری sum2 افزایش می یابد و از طرف دیگر sum1 هم به اندازه مقدار ورودی افزایش می یابد . برنامه بوسیله مقایسه کردن sum1,sum2 چک می کند که مجموع مقادیر ورودی انتخاب شده برابر مجموع مقادیر انتخاب نشده است یا نه.

مثال ۱,۳,۱۰: برنامه درشکل ۱,۳,۸، میانه مقادیر ورودی اش را به خروجی می دهد که [n/2] کمترین مقدار ورودی برای نمونه ای است که ورودی شامل n مقداراست. برنامه خروجی "۲ "درورودی "۳و۲و۱ "داردوورودی "۳و۳و۱و۲ "برنامه خروجی "۳ " را دارد.

```
 median := ?          /* Guess the median. */
write median
count := 0
do
    /* Find the difference between the
    number of values greater than and those
    smaller than the guessed median.
 */
    do
        read x
        if x > median then
            count := count + 1
      if x < median then
            count := count - 1
      if x = median then
            do
                count := count + 1
         or
                count := count - 1
         until true
   until 0 ≤ count ≤ 1
   /* The median is correct for the portion of the
    input consumed so far.                          */
    if eof then accept
until false
```

شکل ۱,۳,۸ یک برنامه غیر قطعی که میانه مقادیر ورودی را پیدا می کند .

برنامه هرمحاسبه با نگهداشتن یک حدس برای مقدار میانه درmedian آغاز می شود. برنامه مقادیر ورودی را می خواند و در count تفاوت بین تعداد مقادیر ورودی که بزرگتر از مقدار نگه داشته شده در median است و تعداد مقادیر ورودی که کوچکتر از مقدار نگهداری شده درmedian است را نگه میدارد.
برای آن مقدار ورودی که برابر با مقدار نگهداری شده درmedian هستند،برنامه حدس

میزند که آنهاباید به عنوان مقادیر بزرگتر یا مقادیر کوچکتر مطرح شوند.
برنامه بوسیله معلوم کردن اینکه count مقدارصفریامقداریک رانگه میداردچک میکند که حدسها درست هستند.

_روابط محاسبه شده بوسیله برنامهp،باR(P)نشان داده میشود،مجموعه {Pیک محاسبه پذیرفتنی در X با خروجی Y دارد |(x , y)}است. وقتیکه p برنامه قطعی است R(P)یک تابع است.

مثال ١,٣,١١ برنامه P در شکل١,٣,٦ فرض میشود که مجموعه ای از اعداد طبیعی برای دامنه ی متغییرها دارد .رابطه R (P) کهP محاسبه می کندبه صورت زیر است.

{ xیک دنباله ازاعدادطبیعی است و a یک عددطبیعی است که درXدیده نشده است |(x , a)}

_زبانی که برنامه p را می پذیرد بوسیله L(P) نشان داده می شود و شامل همه ورودیهاست که p می پذیرد.

پیکربندی های برنامه ها

یک اجرا از یک برنامه در یک ورودی معین یک فرایند مجزا است که ورودی مصرف شده،یک خروجی تولید شده است .متغییرها مقدارشان تغییر کرده است و برنامه دستوراتش را می پیماید. نتیجه هر مرحله در فرایند یه نتیجه مرحله قبل بستگی دارد.نتیجه در هر مرحله پیکربندی ازبرنامه است که نشان میدهد شروع دستورات رسیده است ،مقادیر درمتغییرها ذخیره شده اند، قسمتی از ورودی باقی مانده باید خوانده شود و خروجی تاکنون تولید شده است. بنابر این فرایند میتواند بوسیله یک دنباله ای از انتقالها بین پیکربندی های برنامه توصیف شود.

قطعه ای از برنامه ،قطعه دستور نامیده میشود اگر در یکی از فرمهای زیر باشد :

١)دستورخواندن
٢)دستورنوشتن
٣)دستورت انتقال
٤)قسمتی ازشرط دستور if
5)قسمتی ازیک دستورحلقه زنی do
٦)قسمتی ازدستورحلقه زنی until

۷)دستورپذیرش شرط
۸)دستورعدم پذیرش

برنامه p ,که K قطعه دستور دارد،با m متغییر و دامنه متغییرها با D نشان داده شده است را در نظر بگیرید. یک پیکربندی یاتوصیف آنی از p یک پنج تایی (i,x,u,v,w) است که 1<=i<=k و x یک دنباله از m مقدار از D،و w ,v ,u دنباله هایی از مقادیری ازD هستند .
به طور شهودی ، یک پیکربندی (i, x , u ,x ,w) میگوید که p در i امین قطعه دستور است .j امین متغییرش شامل j این مقدار از x است و u قسمتی از ورودی است که تاکنون خوانده شده است ، باقیمانده از ورودی v است و بنابر این خروجی تاکنون w است (مولفه u در تعریف پیکربندی نیاز نیست. این تعریف برای سازگاری باتعاریف آینده که نیازبه چنین مولفه ای دارندآمده است .)

مثال ۱,۳,۱۲ توجه کنید به شکل ۱,۳,۹فرض میشود

```
last := ?                          /* I1*/
write last                               /* I2*/
do                                          /* I3*/
    read value                        /* I4*/
until value = last                  /* I5*/
if eof then accept                 /* I6*/
reject
```

شکل ۱,۳,۹ برنامه ای شامل هفت قطعه دستور

مجموعه ی اعداد طبیعی برای دامنه d متغییرهاست و با صفر به عنوان مقدار اولیه .
هر خط I_i در برنامه یک قطعه دستور است برنامه k=7 قطعه دستور و m=2 متغییر دارد.
در هر پیکربندی (i ,x ,u ,x ,w) از برنامه i یک عدد طبیعی بین ۱ تا ۷ است وx یک جفت <last و value> از اعداد طبیعی است که مطابق با انتقال ممکن از last،و value در متغییرهای last و value است به طورمتشابه u ,v ,w دنباله هایی از اعداد طبیعی هستند.

پیکربندی (<> و <۳و۲و۱> و <> و<۰ و ۰ >و۱) بیان میکند که برنامه در اولین قطعه دستور است متغییرهای صفر را نگه میدارد، هیچ مقدار ورودی هنوز خوانده نشده، باقیمانده ورودیها "۳و۲و۱ " است و خروجی تهی است .

پیکربندی (<۳> و <۳ >و<۲و۱ >و< ۲و۳ >و۵) بیان میکند که برنامه در قطعه دستور پنجم است و متغییر last مقدار ۳ را نگه میدارد و متغییر value مقدار "۲ " را نگه میدارد و "۲و۱" قسمتی از ورودی مصرف شده تاکنون است. باقیمانده ورودی فقط شامل مقدار ۳ است و خروجی تاکنون فقط شامل مقدار"۳ " است .

پیکربندی (i , x , u , x , w) از p پیکربندی اولیه نامیده می شود اگر1 = i , و xیک دنباله از m مقدار اولیه و u دنباله ی تهی باشد و w نیز تهی باشد .پیکربندی ،پیکربندی پذیرفتنی است اگر i امین قطعه دستور از p یک دستور پذیرش شرط باشد وv یک دنباله تهی باشد. یک حرکت مستقیم درp از پیکربندی c_1 به پیکربندی c_2 بوسیله c_1|- pc_2 نشان داده می شود یا ساده تر c1|-c2 اگر p معین باشد . دنباله ای از تعداد نامشخص از حرکات p از پیکربندی c_1 به پیکربندی c_2 با c1...p*c2 نشان داده می شود یا ساده تر c1..*c2 اگر p معین شده باشد.

مثال ۱,۳,۱۳:توجه کنید به برنامه در شکل ۱,۳,۹ در ورودی "۳و۲و۱ "یک محاسبه پذیرفتنی دارد که از میان دنباله زیر از حرکات بین اشکال میرود . اولین شکل در دنباله شکل متمایزی از برنامه در ورودی "۳و۲و۱ " است و آخرین شکل در دنباله یک شکل پذیرفتنی از برنامه است. در هر شکل (I , x ,u , v , w) جفت <x=<last , xalue مطابق با انتقال last , value به متغییرهایشان به ترتیب است.

$$
\begin{aligned}
(1,\langle 0,0\rangle,\langle\rangle,\langle 1,2,3\rangle,\langle\rangle) &\vdash (2,\langle 3,0\rangle,\langle\rangle,\langle 1,2,3\rangle,\langle\rangle)\\
&\vdash (3,\langle 3,0\rangle,\langle\rangle,\langle 1,2,3\rangle,\langle 3\rangle)\\
&\vdash (4,\langle 3,0\rangle,\langle\rangle,\langle 1,2,3\rangle,\langle 3\rangle)\\
&\vdash (5,\langle 3,1\rangle,\langle 1\rangle,\langle 2,3\rangle,\langle 3\rangle)\\
&\vdash (3,\langle 3,1\rangle,\langle 1\rangle,\langle 2,3\rangle,\langle 3\rangle)\\
&\vdash (4,\langle 3,1\rangle,\langle 1\rangle,\langle 2,3\rangle,\langle 3\rangle)\\
&\vdash (5,\langle 3,2\rangle,\langle 1,2\rangle,\langle 3\rangle,\langle 3\rangle)\\
&\vdash (3,\langle 3,2\rangle,\langle 1,2\rangle,\langle 3\rangle,\langle 3\rangle)\\
&\vdash (4,\langle 3,2\rangle,\langle 1,2\rangle,\langle 3\rangle,\langle 3\rangle)\\
&\vdash (5,\langle 3,3\rangle,\langle 1,2,3\rangle,\langle\rangle,\langle 3\rangle)\\
&\vdash (6,\langle 3,3\rangle,\langle 1,2,3\rangle,\langle\rangle,\langle 3\rangle)
\end{aligned}
$$

این زیر محاسبه شامل صفر حرکت است :

$$(1,\langle 0,0\rangle,\langle\rangle,\langle 1,2,3\rangle,\langle\rangle)\vdash^{*}(1,\langle 0,0\rangle,\langle\rangle,\langle 1,2,3\rangle,\langle\rangle)$$

و این زیر محاسبه دارای یازده حرکت :

$$(1,\langle 0,0\rangle,\langle\rangle,\langle 1,2,3\rangle,\langle\rangle)\vdash^{*}(6,\langle 3,3\rangle,\langle 1,2,3\rangle,\langle\rangle,\langle 3\rangle)$$

۱-٤ مسائل

دو بخش اول از این فصل با جنبه های متمایزی از اطلاعات سر و کار دارند.در بخش سوم برنامه را ملاحظه کردید.هدف باقیمانده این فصل بکاربردن انگیزه ای برای برنامه های نوشته شده بواسطه دستکاری اطلاعات ،به عبارت دیگر، با مسائل است.

هرمسئله k زوجی است شامل یک مجموعه و یک سوال ، که سوال میتواند با هر عنصری از مجموعه سر و کار داشته باشد. مجموعه ، دامنه مسئله نامیده میشود و عناصرش نمونه هایی از مسئله نامیده میشوند.

مثال ۱,۱,٤ مسئله k_1 را که با دامنه وسوال ذیل مشخص شده درنظر بگیرید.

دامنه:

{ a و b اعداد طبیعی هستند |<a, b> }

سوال :

برای نمونه مفروض x=<a,b> مقدار جزء صحیح y که از تقسیم a بر b بدست می آید کدام است؟

دامنه مسئله نمونه های <۰, ۰>, <۵, ۰>, <۳, ۸>, <٦, ۲٤> و <۸, ۲۷> را شامل میشود.از سوی دیگر <۳, ۵-> نمونه ای ازمسئله نیست .

برای نمونه <۸, ۲۷> مسئله پرسیده شده این است که جز صحیح تقسیم ۲۷ بر۸ چقدر است. بطور مشابه ، برای نمونه <۰, ۰> مسئله پرسیده شده اینست که جزصحیح ازتقسیم ۰ بر ۰ کدام است.

یک پاسخ به سوالی که مسئله k برای یک نمونه مفروض مطرح میکند یک راه حل برای مسئله در نمونه مفروض گفته میشود.

مجموعه { (x, y) | x نمونه ای از مسئله است و y راه حلی برای مسئله در x است } رابطه ایجادشده بواسطه مسئله است وبا R(k) مشخص میشود. اگر برای هرنمونه مسئله راه حل به صورت yes یا no باشد مسئله یک مسئله تصمیم گیری (انتخاب) گفته میشود.

مثال ۱-٤-۲ مسئله k_1 درمثال ۱-٤-۱ را ملاحظه کنید.مسئله در نمونه <۸, ۲۷> حل ۳ ، و در نمونه <۰, ۰> یک حل تعریف نشده را میدهد. K_1 رابطه $R(K_1)$ = {(<0, 1>, 0), (<0, 2>, 0), (<1, 1>, 1), (<0, 3>, 0), (<1, 2>, 0), (<2, 1>, 2), ((<0, 3>, 0), . . . } ایجاد میکند.

مسئله k_1 یک مسئله تصمیم گیری نیست . اما مسئله k_2 کـه بوسـیله زوج ذیـل تعریـف میشود مسئله تصمیم گیری است .

دامنه :

{(a, b) | a و b اعداد طبیعی هستند}

سوال :

آیا برای نمونه مفروض <a, b>، b بر a تقسیم پذیر است؟

قابلیت حل پاره ای و قابلیت حل

برنامه P مسئله مفروض K را به طور پاره ای حل می کند اگر برای هرنمونه ازمـسئله K پاسخی داشته باشد،به عبارت دیگر،اگر R(P) = R(K). بعلاوه اگر همه محاسبات برنامه محاسبات توقف کننده باشند، آنگاه برنامه را حل کننده ی یک مسئله گویند.

مثال ١-٤-٣ برنامه P_1 در شـکل ١-٤-١ (a) را ملاحظـه کنیـد. دامنـه متغیرهـا برابـر مجموعه اعداد طبیعی گرفته شده است . برنامه مسئله K_1 از مثـال ١-٤-١را بطـور پـاره ای (ناکامل) حل میکند.

```
read a
read b
ans := 0
if a ≥ b then
   do
      a := a - b
      ans := ans + 1
   until a<b
write ans
if eof then accept
```

(a)

```
read a
read b
do
   if a = b then
      if eof then accept
   a := a - b
until false
```

(b)

Figure (a) A program that partially solves the problem of dividing

1.4.1 natural numbers. (b) A program that partially decides the problem of divisibility of natural numbers.

شکل ۱-٤-۱ (a) برنامه ای است که مسئله تقسیم اعداد طبیعی را بطور نسبی حل میکند. (b) برنامه ای است که مسئله ای از قابلیت تقسیم اعداد طبیعی را بطورنسبی معین میکند.

برای ورودی "۸,۲۷" برنامه در یک وضعیت پذیرش با پاسخ ۳ درخروجی متوقف میشود. برای ورودی "۰،۰" برنامه هرگز متوقف نمیشود و بنابر این برنامه روی یک چنین ورودی خروجی تعریف نشده است. بر روی ورودی "۲۷" و ورودی "۲۷,۸,۲" برنامه در وضعیت نپذیرفتن (ردکردن) متوقف میشود و خروجیی تعریف نمیکند.

برنامه p_1، مسئله k_1 را حل نمیکند زیرا هنگامیکه ورودی برای 0،b است متوقف نمیشود . p_1 میتواند به یک برنامه p که k_1 را بوسیله گذاشتنِ p برای جلوگیری کردن از انتساب ۰ به جای b حل میکند اصلاح شود.

یک برنامه را اگر در دو شرط ذیل صدق کند تصمیم گیرنده پاره ای یک مسئله گویند.

a . مسئله یک مسئله تصمیم گیری باشد.

b . برنامه یک ورودی مفروض را بپذیرد اگر و تنها اگر مسئله برای ورودی یک پاسخ yes داشته باشد، به عبارت دیگر، برنامه زبان { x | x نمونه ای از مسئله است و مسئله درx پاسخ yes رادارد } رامیپذیرد.

یک برنامه را تصمیم گیرنده ی یک مسئله گویند اگر آن تصمیم گیرنده ی پاره ای مسئله باشد و همه محاسباتش محاسبات توقف کننده باشند.

مثال۱-٤-٤ بحث کردن پیرامون تصمیم پذیری پاره ای یا تصمیم پذیری مسئله k_1 ازمثال ۱-٤-۱ بوسیله یک برنامه بی معنی است ، زیرا مسئله یک مسئله تصمیم نیست

.از سوی دیگر ، مسئله k_2 از مثال ١-٤-٢ یک مسئله تصمیم گیری است. مسئله اخیر بوسیله برنامه p_2 درشکل١-٤-١.(b) بطور پاره ای تصمیم پذیر است.

تفاوت اصلی بین یک برنامه p_1 که حل کننده پاره ای (تصمیم گیرنده پاره ای) یک مسئله است و یک برنامه p_2 که حل کننده (تصمیم گیرنده) همان مسئله است درآن است که p_1 میتواند یک ورودی را بوسیله یک محاسبه بدون توقف رد کند.(بیاد می آوریم که روی یک ورودی که بوسیله یک برنامه پذیرفته شده است ،برنامه فقط محاسبات پذیرفته شده دارد، و همه این محاسبات،محاسبات توقف کننده دارند . اما بر روی یک ورودی پذیرفته نشده برنامه میتواند بیش از یک محاسبه داشته باشد، از سویی ممکن است برخی هرگز متوقف نشوند .

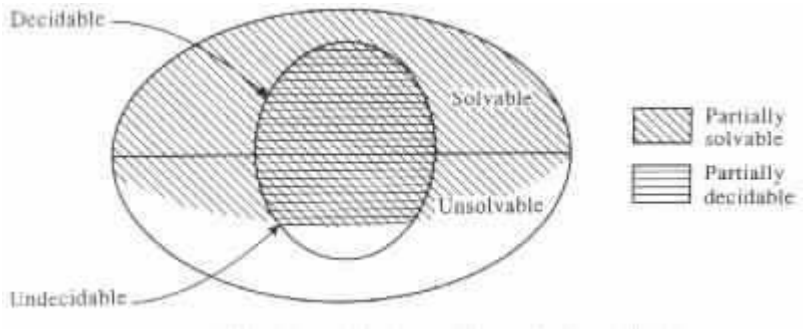
مفاهیم قابلیت حل پاره ای ، قابلیت حل ، تصمیم پذیری پاره ای وتصمیم پذیری یک مسئله بوسیله یک برنامه، میتواند مستقیما دریک روش آسان با جایگزینی توابع موثر(اکیداماشینی) بجای برنامه تعمیم یابد . بهرحال ،رسمی کردن تعمیم ها به این نیاز دارد که تصویر شهودی از تابع موثر رسمی شود. در هر صورت ، این قبیل تعمیم هایِ ذاتا بدیهیِ یک مسئله را حل پذیر بطور پاره ای ، حل پذیر ، تصمیم پذیرپاره ای و تصمیم پذیر گویند اگر آن بتواند بوسیله یک تابع موثر،به ترتیب ،بطور پاره ای حل شود، حل شود، بطور پاره ای تصمیم گرفته شود و تصمیم گرفته شود.

درادامه توابع موثر ،الگوریتم نامیده خواهند شد.

مثال ١-٤-٥ برنامه p_1 از مثال ١-٤-٣ شرح میدهد که مسئله k_1 از مثال ١-٤-١ از چه طریق میتواند حل شود. برنامه p_2 از مثال ١-٤-٤ شرح میدهد که مسئله k_2 از مثال ١-٤-٢ از چه طریق میتواند حل شود.

یک مسئله غیر قابل حل گفته میشود اگر هیچ الگوریتمی نتواند آن را حل کند. مسئله تصمیم ناپذیر گفته میشود اگر آن مسئله تصمیم گیری باشد و هیچ الگوریتمی نتواند

برای آن تصمیم بگیرد. ارتباط بین رده های مختلف مسائل در نمودار ون شکل ۱-٤-۲ توضیح داده شده است .



رده بندی مجموعه ای ازمسائل شکل ۱-٤-۲

باید در نظر داشت که یک مسئله غیر قابل حل ممکن است بوسیله یک الگوریتم که از یک جستجوی کامل برای یک راه حل بوجود آمده است بطور پاره ای حل پذیر باشد .در چنین مواردی هنگامیکه راه حل تعریف شده است سرانجام حل پیدا میشود ، اما هنگامیکه راه حل تعریف نشده است جستجو ممکن است برای همیشه ادامه پیدا کند. بطور مشابه ،یک مسئله غیر قابل حل ممکن است بوسیله یک الگوریتم که از یک جستجوی کامل برای یک راه حل بوجود آمده است بطور پاره ای تصمیم پذیر باشد. هرچند، در اینجا هر وقت که راه حل ارزش yes داشته باشد بالاخره حل پیدا میشود، اما ممکن است وقتیکه راه حل ارزش no پیدا کرد جستجو برای همیشه ادامه پیدا کند .

مثال ۱-٤-٦ مسئله عضویت عبارت تهی برای گرامرها مسئله ایست شامل دامنه وسوال زیرین.

دامنه:

{ G | یک گرامر است G}

سوال :

برای گرامر مفروض G کلمه تهی ε در L(G) هست ؟

نشان دادن مسئله ای که تصمیم ناپذیر است امکان پذیر است (e.g. رجوع کنید به قضیه ٤-٦-٢ و تمرین ٤-٥-٧). از سوی دیگر ، مسئله بطورنسبی حل پذیر است زیرا یک نمونه مفروض $G = <N,\Sigma, P,S>$ را میتواند بطور کامل برای یک اشتقاق به شکل $S = \alpha_0 \Rightarrow \alpha_1 \Rightarrow \ldots \Rightarrow \alpha_{n-1} \Rightarrow \alpha_n = \varepsilon$ با در نظر گرفتن همه اشتقاقهای با طول n برای $n = 0,1,2,\ldots$ که با چنین الگوریتمی اشتقاق میابد جستجوکند ، و چنانچه ε در L(G) باشد بالاخره پیدا خواهد شد. هر چند ، اگر ε در L(G) نباشد ،آنگاه جستجو ممکن است هرگز خاتمه نیابد .

برای گرامر $G = <N,\Sigma , P, S>$ که قوانینش در زیر فهرست شده اند، الگوریتم به روش زیرین پیش میرود.

$$\begin{aligned} S &\rightarrow aBS \\ &\rightarrow Ba \\ aB &\rightarrow SB \\ BS &\rightarrow \varepsilon \end{aligned}$$

الگوریتم با تعیین مجموعه ای ازهمه اشتقاقهای $\Psi_1 = \{S\rightarrow aBS , S\rightarrow Ba\}$ با طول n=1 شروع میشود.پس از مشخص شدنِ آنکه هیچ یک ازاشتقاقها در Ψ_1 رشته تهی ε را تولید نمیکند، الگوریتم مجموعه ای ازتمام اشتقاقهای $\Psi_2 = \{S\rightarrow aBS\rightarrow aBaBS , S\rightarrow aBS\rightarrow SBS , S\rightarrow aBS , S\rightarrow a\}$ با طول n=2 را تعیین میکند. سپس الگوریتم باتعیین مجموعه Ψ_3 از اشتقاقهای با طول ۳ ،مجموعه Ψ_4 ازتمام اشتقاقهای با طول ٤ و غیره و ادامه میابد. الگوریتم وقتی و فقط وقتی ، متوقف میشود (با پاسخ yes) که یک مجموعه Ψ_n ازاشتقاقهای با طول n پیدا شود که شامل یک اشتقاق برای ε باشد. چنین مجموعه Ψ_n ی برای n=5 بواسطه اشتقاقِ $S \Rightarrow aBS \Rightarrow SBS \Rightarrow BaBS \Rightarrow BSBS \Rightarrow \varepsilon$ وجود دارد.

از سوی دیگر، برای گرامر $G = <N,\Sigma,P,S>$ که قواعد تولیدش در زیر فهرست شده اند، الگوریتم هرگز متوقف نمیشود.

$$\begin{aligned} S &\rightarrow aSb \\ &\rightarrow aAb \\ A &\rightarrow \varepsilon \end{aligned}$$

عدم قابلیت حل یک مسئله به این معنی نیست که دربرخی نمونه هایش نتواند راه حلی پیدا کند. فقط به این معناست که هیچ الگوریتمی نمیتواند بطور یکنواخت برای هر نمونه مفروض یک راه حل پیدا کند. در نتیجه ، یک مسئله غیر قابل حل ممکن است نسخه هایی ساده شده که قابل حلند داشته باشد. ساده سازی ها میتواند درسوال درخواست شود و در دامنه مطرح شود.

مثال ۱-٤-۷ مسئله عضویت عبارت تهی برای گرامر نوع ۱ مسئله ایست شامل دامنه وسوال ذیل.

دامنه :

{ G | Gیک گرامر نوع ۱ است }

سوال :

برای گرامر مفروض G کلمه تهی ε در L(G) هست ؟

مسئله تصمیم پذیر است زیرا برای یک گرامرِ نوعِ ۱ مفروض $G = <N, \Sigma, P,S>$ ، ε در L(G) هست اگروتنها اگر $S \rightarrow \varepsilon$ یک قاعده از G باشد .

یک تابع f محاسبه پذیر (بترتیب ، محاسبه پذیرپاره ای ، محاسبه ناپذیر) گفته میشود اگر مسئله بوسیله دامنه و سوالی که حل پذیراست (بترتیب ، حل پذیرِ پاره ای ، حل ناپذیر) مشخص شود.

دامنه :

دامنه ای از f .

سوال :

مقدار(f(x در نمونه مفروض x چقدر است ؟

مسائلی درباب برنامه ها

اگرچه برنامه ها نوشته شده اند تامسائل را حل کنند، ولی مسائل جالب توجهی نیز وجود دارند که به برنامه ها مربوط میشوند. در ذیل تعدادی مثال از این قبیل مسائل تصمیم گیری (انتخابی) وجود دارد .

مسئله توقف یکنواخت برای برنامه ها

دامنه :

مجموعه ی همه برنامه ها .

سوال :

آیابرنامه مفروض روی هریک ازورودی هایش متوقف میشود ، به عبارت دیگر ، همه محاسبات برنامه ها ،متوقف می شوند ؟

مسئله توقف برای برنامه ها

دامنه :

{ (P, x) | P یک برنامه است و x یک ورودی برای P است }

سوال :

آیا برای نمونه مفروض (P,x)،P روی x متوقف میشود ، یا بعبارت دیگر ،همه محاسباتِ p روی ورودی x محاسبات توقف کننده هستند ؟

مسئله شناسایی/ پذیرش برای برنامه ها

دامنه :

{(P, x) | P یک برنامه است و x یک ورودی برای P است }

سوال :

آیا برای نمونه مفروض (P,x)، x،P را میپذیرد ؟

مسئله عضویت برای برنامه ها

دامنه:

{(P, x, y)| P یک برنامه است وx و y به ترتیب مقادیری ازدامنه ی متغیرهایِ p هستند}

سوال :

آیا(x,y) برای نمونه مفروض (P,x,y) دررابطه R(P) هست ، به عبارت دیگر، آیا P یک محاسبه پذیرفته شده روی x با خروجی y دارد ؟

مسئله پوچی برای برنامه ها

دامنه :

مجموعه ی همه برنامه ها

سوال :

آیا برنامه مفروض یک زبان تهی را میپذیرد،به عبارت دیگر، آیا برنامه ای با هیچ ورودی پذیرفته میشود ؟

مسئله ابهام برای برنامه ها

دامنه:

مجموعه ی همه برنامه ها

سوال:

آیا برنامه مفروض دو یا بیشترازدومحاسبه ی پذیرفته شده که خروجی یکسانی برای چند ورودی تعریف میکند، دارد؟

مسئله تک مقداری برای برنامه ها

دامنه:

مجموعه ی همه برنامه ها

سوال:

آیا برنامه مفروض یک تابع تعریف میکند،به عبارت دیگر، آیا برنامه مفروض برای هرورودی بیش ازیک خروجی دارد ؟

مسئله هم ارزی برای برنامه ها

دامنه :

$\{(P_1, P_2) \mid P_1$وP_2 برنامه اند$\}$

سوال :

آیا زوج مفروض(P_1, P_2) ازبرنامه ها ، رابطه یکسانی را تعریف میکند ،بعبارت دیگر، آیا $R(P_1) = R(P_2)$ ؟

مثال ١-٤-٨ دو برنامه P_1 و P ازمثال ١-٤-٣ هم ارزند ، اما فقط P_2 روی تمام ورودی ها متوقف میشود .

مسئله توقف غیر یکنواخت ،مسئله عدم ابهام ، مسئله عدم هم ارزی و قس علی هذا برای برنامه ها بطور مشابه تعریف میشوند همچنانکه بترتیب، مسئله توقف یکنواخت (یکسان)، مسئله ابهام ، مسئله هم ارزی و غیره برای برنامه ها بطور یکسانی تعریف میشوند. تنها تفاوت اینست که سوالها تحت الفظی بیان میشوند به این ترتیب بنابر مسائل جدید راه حل ها معکوس مسائل قدیمی هستند ، به عبارت دیگر ،yes میشود no و no میشود yes .

نتیجه میشود که پاسخ دادن به سوالات غیر بدیهی پیرامون برنامه ها ، اگر غیرممکن نباشند ، مشکلند . وقتیکه برنامه های تحت بررسی متناسبا محدود شوند این طبیعی است که انتظار داشته باشیم که مسائل آسانتر شوند . هر اندازه که برنامه ها محدودیت

داشته باشند ، علاوه بر کاهش قدرت بیانشان و افزایشِ منابع که انها بسبب چنین محدودیتهایی لازم دارند ، سوالهایی مجذوب کننده نیز روی خود برنامه ها وجود دارند .

مسائلی درباب گرامرها

برخی از مسائل به برنامه هایی که میتوانند به روش مشابهی تعریف شـوندوهمچنین بـه گرامرها مربوط میشوند . در ذیل چند مثال ارچنین مسائلی آمده است .

مسئله عضویت برای گرامرها

دامنه:

{(G, x)| G یک گرامر <N, Σ, P, S> است و x یک رشته در* Σ است }

سوال :

آیا برای نمونه مفروض (G, x)، x در L(G) هست ؟

مسئله پوچی برای گرامرها

دامنه :

{G| G یک گرامر است }

سوال :

آیا گرامرمفروض ،یک زبان تهی را تعریف میکند ؟

مسئله ابهام برای گرامرها

دامنه :

{G| G یک گرامر است }

سوال :

آیا گرامرمفروض G دویا بیشترازدو گراف اشتقاق برای تعدادی رشته در L(G) دارد ؟

مسئله هم ارزی برای گرامرها

دامنه :

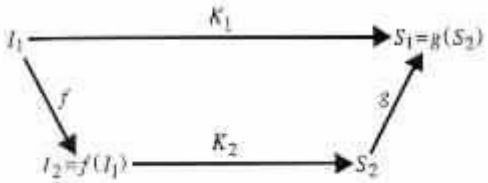
{ G1| ⟨G_1, G_2⟩} و G2 گرامرند

سوال :

آیا زوج گرامرهای مفروض، زبان یکسانی تولید میکنند ؟

۱-۵ تقلیل پذیری در میان مسائل :

یک شیوه ی رایج در حل مسائل تغییر شکل دادن آنها به مسائل مختلف و حل یکی از آنهاست. و استنتاج کردن راه حل ها برای مسائل اصلی از آنها برای مسائل جدید. این شیوه وقتی که حل مسائل جدید ساده تر است یا وقتی که الگوریتم حل آنها از قبل شناخته شده باشد سودمند است. یک چنین شیوه ای در طبقه بندی مسائل بر طبق پیچیدگی های آنها بسیار سودمند است .



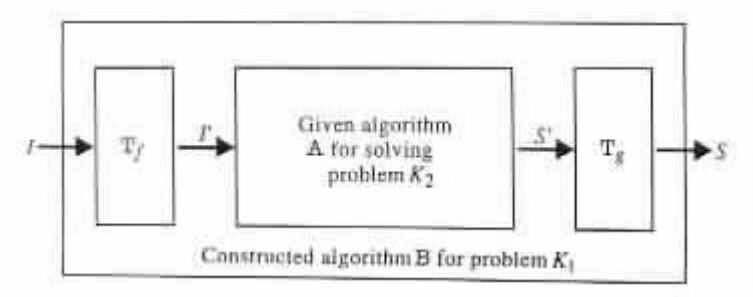
یک مسئله ی K_1 زمانیکه به مسئله ی دیگر K_2 تغییر شکل می یابد تقلیل پذیر به مسئله جدید نامیده می شود. بویژه به یک مسئله ی K_1 تقلیل پذیر به مسئله ی K_2 گفته می شود اگر توابع جمع محاسباتی f و g با خواص زیر وجود داشته باشد (به شکل ۱-۵-۵ نگاه کنید)

اگر I_1 یک نمونه از K_1 باشد آنگاه :

a : $I_2 = f(I_1)$ یک نمونه از K_2 است.

b : K_1 یک راه حل S_1 در I_1 دارد اگر و فقط اگر K_2 یک راه حل S_2 در $I_2 = f(I_1)$ داشته باشد بطوریکه $S_1 = g(s_2)$.

مثال ۱-۵-۱ ψ را مجموعه ی $\{m|m=2^i for some integer i\}$ قرار دهید . مسئله ی به توان رساندن اعداد از ψ به مسئله ی ضرب اعداد طبیعی کاهش پذیر است این کاهش یک مفهوم ضمنی از تساوی $x^y=(2^{\log x})^y=2^{y.\log x}$ است که به ترتیب انتخابی از $f(x,y)=(y,\log x)$ و $g(z)=z^2$ را برای f و g می پذیرد.



مثال ۱-۵-۲ K_ϕ و $K_\equiv$ را به ترتیب به عنوان مسائل تهی وهم ارزی برای برنامه ها در نظر بگیرید. آنگاه کاهش از K_ϕ به $K_\equiv$ بوسیله ی توابع f و g که به فرم زیر هستندانجام می شود. f یک تابع است که مقدارش در برنامه ی P جفت برنامه های (P,P_ϕ)است . P_ϕ یک برنامه است که هیچ ورودی را نمی پذیرد به عنوان مثال برنامه ای که تنها شامل دستورالعمل reject است. g یک تابع است که g(yes)= yes و g(no) = no .

اگر K_1 یک مسئله ی کاهش یافته به مسئله ی K_2 باشد. بوسیله ی توابع کامل f و g که به ترتیب به وسیله ی الگوریتم های T_F و T_g محاسبه پذیرند و K_2 قابل حل باشد بوسیله ی یک الگوریتم A آنگاه یکی از آنها می تواند یک الگوریتم B برای حل K_1 فراهم کند . (شکل ۱-۵-۲ را ببینید)

با گرفتن یک ورودی I الگوریتم B بوسیله ی اجرای T_F روی I شروع می شود. سپس B خروجی I' از T_F به A را می دهد. در نهایت B خروجی S' از A به T_g را می دهد و فرض می کند همان خروجی S از T_gبدست می آید.

فصل دوم

برنامه های حافظه متناهی

اهداف

در پایان فصل، دانشجو با مفاهیم زیر آشنا می‌شود:

مبدل های حالت متناهی
اتوماتهای حالت متناهی و زبانهای منظم.
محدودیت های برنامه های با حافظه متناهی.
خواص بستاری برای برنامه های با حافظه متناهی.
خواص تصمیم پذیری برای برنامه های با حافظه متناهی .

مقدمه

برنامه های حافظه متناهی احتمالا یکی از ساده ترین طبقات برنامه ها هستند برای اینکه مطالعه ی ما با معنا شود.اولین بخش این فصل انگیزه ی تحقیق درباره ی این طبقه است.

دومین بخش سیستم های ریاضیاتی مبدل حالت متناهی را معرفی میکند و نشان میدهد این مدل ها نتیجه ی محاسبه ی برنامه ها ی حافظه متناهی هستند. سومین بخش به تو صیف صفات اختصاصی دستوری زبانهایی می پردازد که برنامه های حافظه متناهی انها

را می پذیرند و چهارمین بخش به محدودیتهای این برنامه ها می پردازد .بخش پنجم درباره ی مهمترین خواص بستاری در این برنامه ها و کاربردشان در برنامه های حافظه متناهی بحث میکند . و اخرین بخش به خواص تصمیم پذیری برای برنامه های حافظه متناهی رسیدگی میکند.

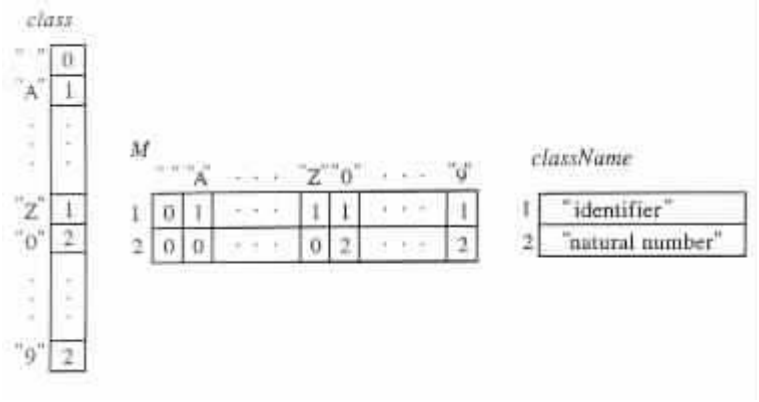
۲-۱ انگیزه

اغلب مفید است که وقتی اطلاعات را در یک زمینه جدید گسترش می دهیم بررسی را با یک نمونه ی محدود شده شروع کنیم و سپس رفته رفته به نمونه های کلی تر گسترش دهیم. چنین روشی افزایش تدریجی پیچیدگی مباحث را اجازه می دهد. بخصوص این یک استراتژی عمومی کلی است که بررسی سیستمهای نامتناهی بوسیله ی بررسی زیر سیستمهای متناهی شروع شود. ما چنین روشی را به وسیله ی برنامه های کاربردی با دامنه ی متناهی از متغییرها بکار می گیریم، که برنامه های حافظه متناهی یا برنامه های دامنه متناهی نامیده می شوند.

بهر حال، باید به این توجه شود که برنامه های حافظه متناهی به خاطر تواناییشان مهم هستند. آنها قادرند برخی انواع معمولی برنامه های کامپیوتری را طراحی و آنالیز کنند. برای مثال در کامپایلر (برنامه هایی که برنامه های نوشته شده به زبان سطح بالا را به برنامه معادل، نوشته شده در زبان ماشین ترجمه می کنند)، تحلیل گرهای لغوی به طور اساسی به عنوان برنامه های حافظه متناهی طراحی شده اند وظیفه اصلی تحلیل گر لغوی بررسی ورودی معین و تعیین مکان کردن نمادهایی است که متعلق به هر یک از نشانه ها هستند.

```
char :="??"
do
/*?.Find the first character of the next token */
```

```
if char = "??" then
do
if eof then accept
read char
"??" until char≠
/*?.Determine the class of the token */
(charClass := class(char
(write className(charClass
/*?.Determine the remaining characters of the token */
do
write char
if eof then accept
oldCharClass := charClass
read char
(charClass := M(charClass, char
until charClass≠ oldCharClass
until false
```



شکل ۲-۱- ۱ (b) جدول آنالیز لکسیکال (a) آنالیز لکسیکال

مثال ۲-۱-۱ LEXANLرا برنامه حافظه متناهی در شکل ۲-۱-۲ در نظر بگیرید . دامنه ی متغییرها مساوی با { ۲ و ۱ و ۰ و " ۰...۹" و "۲" و ... و "A" و " ? ? ? " } فرض می شود که "??" بعنوان مقدار اولیه است. نوع تابع، نام نوع و M بوسیله ی جداول

شکل (b) تعریف شده اند.

LEXANL یک تحلیل گر لغوی است که نشانه ها را در یک ورودی معین تعیین می کند و آنها را در اعداد طبیعی و شناسه ها طبقه بندی می کند. هر شناسه بوسیله ی یک حرف که بوسیله ی تعداد اختیاری از حروف و ارقام دنبال می شود، نمایش داده می شود. (شناسه با حرف شروع می شود و دنبال آن می تواند تعداد متناهی حرف یا رقم بیاید) هر عدد طبیعی بوسیله ی یک یا بیشتر رقم نمایش داده می شود. هر جفت از نشانه های متوالی (بجز برای یک عدد طبیعی) که بوسیله ی شناسه شناخته می شوند باید بوسیله ی یک یا چند جای خالی از هم جدا شوند.

LEXANL براحتی می تواندبا ایجاد تغییراتی در آن یک نوع متفاوت از نشانه ها را تشخیص دهد. فقط به وسیله ی تعریف دوباره نوع، نام نوع، و M.

پروتکل ها برای فرایندارتباط برقرار کردن نمونه هایی از سیستمهایی هستند که مکرراً به عنوان برنامه های حافظه متناهی طراحی شده اند. در چنین سیستمهایی، هر فرآیند به وسیله ی یک برنامه حافظه متناهی نمایش داده می شود. هر کانال (راه) از یک فرآیند به دیگری بوسیله ی یک صف مجازی جدا شده است یعنی بوسیله ی یک حافظه (اولین ورودی- اولین خروجی first output – first input). در هر مورد صف ،آن پیغامهایی که از طریق کانال فرستاده شده ولی هنوز دریافت نشده نگه می دارد. هر پیغام فرستاده شده بوسیله ی نوشتن پیغام در کانال مناسب نمایش داده می شود. هر پیغام دریافت شده بوسیله ی خواندن پیغام از کانال مناسب نمایش داده می شود.

در بخش ۲,٦ نشان داده می شود که برنامه های حافظه متناهی چندین خاصیت تصمیم پذیری جالب دارند. چنین خاصیت های تصمیم پذیر، برنامه های حافظه متناهی را جالب می کنند تا بعنوان ابزاری برای نشان دادن پیچیدگی برخی مسائل که به طور ظاهری ناوابسته اند به کار روند. (یعنی با استفاده از خاصیت تصمیم پذیری برنامه های حافظه متناهی، می توان پیچیدگی رابطه ی چند مساله، که به نظر رابطه ای ندارند را فهمید).

مثال ۲-۱-۲ : مساله تصمیم گیری k درباره تصمیم گیری وجود راه حل روی اعداد طبیعی برای برابری سیستمهای Diophantine خطی را در نظر بگیرید. که برای برابری سیستم هایی به فرم زیر است aijها و bi ها صحیح فرض می شوند.

$$a_{1\,1}x_1 + \cdots + a_{1\,n}x_n = b_1$$

$$\vdots$$

$$a_{m\,1}x_1 + \cdots + a_{m\,n}x_n = b_m$$

به نظر می رسد هیچ الگوریتم سرراست و مستقیمی برای تصمیم پذیری این مسئله وجود ندارد، با این حال با جستجوی کامل و جامع برای انتساب به متغیرهاکه سیستم را ارضا کند ، مسئله به طور پاره ای تصمیم پذیر می شود .

برای هر نمونه I از K، یک برنامه حافظه متناهی P_I می تواند ایجاد شود که برخی ورودیها را می پذیرد اگر و تنها اگر I یک راه حل روی اعداد طبیعی داشته باشد. در نتیجه، مساله K به مساله خالی بودن برای برنامه های حافظه متناهی- (قضیه ۲-٤-۱) ساده شدنی است. در واقع دلیل قضیه ۱-٤-۱ ایجاب می کند که سیستم یک راه حل روی اعداد طبیعی داشته باشد اگر و تنها اگر سیستم یک راه حل در مقادیر X_M و .... و X_1 دارد که بزرگتر از برخی کرانه های مربوط به فقط a_j و b_i و m و n نیستند.

برنامه های کامپیوتری که حافظه کمکی بجز برای نگهداری ورودیها و خروجیها استفاده نمی کنند بر طبق تعریف مثالها، برنامه های حافظه متناهی هستند. به هر حال

چنین برنامه هایی می توانند با دامنه های با کاردینالیتی بالا سر و کار داشته باشند. (۲^K برای کامپیوتر ها با K بیت برای هر کلمه) و به عنوان یک نتیجه، متناهی بودن دامنه ها تأثیری روی طراحی آنها ندارد. بنابراین چنین برنامه هایی نباید به طور کلی به عنوان برنامه های حافظه متناهی « طبیعی» مطرح شده باشند.

۲-۲ مبدل های حالت متناهی

- برنامه های حافظه متناهی مختصر
- مبدل های حالت متناهی
- پیکربندی انتقال در مبدل های حالت متناهی
- قطعیت یا عدم قطعیت در مبدل های حالت متناهی
- محاسبات مبدل های حالت متناهی
- روابط و زبانهای مبدل حالت متناهی
- از مبدل های حالت متناهی تا برنامه های حافظه متناهی .

در مرکز تحقیق برنامه های حافظه متناهی مشاهده شده است که مجموعه ی همه ی حالات قابل دسترس در محاسبات یک چنین برنامه ای متناهی است .
در نتیجه، محاسبات هر برنامه ی حافظه متناهی می تواند بوسیله ی یک دسته حالات متناهی و یک دسته قوانین متناهی برای گذربین آن حالات توصیف شود.
برنامه های حافظه متناهی مختصر:
P یک برنامه ی حافظه متناهی با m متغیر $x_1 \ldots x_m$ و k بخش دستورالعمل $I_1 \ldots I_K$ است مقدار اولیه متغیر P را با $\odot$ مشخص می کنیم.

هر حالت P یک m+1 تایی $\left[i, v_1, \ldots, v_m \right]$ هست که i یک عدد صحیح بین ۱ و K

است و $v_1 \ldots v_m$ مقادیری از دامنه ی متغیرهاست.

یک حالت $[i, v_1, \ldots, v_m]$ نشان می دهد که برنامه به بخش دستورالعمل I_i با مقادیر $v_1 \ldots v_m$ در متغیرهای $x_1 \ldots x_m$ به ترتیب رسیده است .

مثال ۲-۲-۱ p را به عنوان برنامه در شکل ۲-۲-۱ در نظر بگیرید. دامنه ی متغیرها به طور فرض برابر $\{0,1\}$ است و اولین مقدار به طور فرض ۰ است.

$[i, x, y]$ را مشخص کننده ی حالت p قرار دهید که با بخش دستورالعمل I_i و مقدار x در x و مقدار y در y مطابق است.

```
?
/*? x := ?                    /* I1
/*? write x                      /* I2
/*? do                           /* I3
/*? do                         /* I4
/*? read y                    /* I5
/*? until x = y                /* I6
/*? if eof then accept         /* I7
/*? do                          /* I8
/*? x := x - 1            /* I9
or
/*?y := y + 1             /* I10
/*?until x≠y                  /* I11
/*?until false                /* I12
```

شکل ۲-۲-۱ یک برنامه با حافظه محدود با دامنه ورودی (۱و۰)

?

حالت[۰, ۱،۰] نشان می دهد که برنامه به اولین بخش دستورالعمل را با مقدار x=۰ و y=۰ رسیده است .

حالت $[5,1,0]$ نشان می دهد که برنامه پنجمین بخش دستورالعمل با مقدار۱= x و ۰= y را بدست آورده .

از حالت $[5,1,0]$ برنامه می تواند به هر یک از دو حالت $[6,1,0]$ یا $[6,1,1]$ برسد.

در گذر از حالت $[5,1,0]$ و حالت $[6,1,0]$ برنامه مقدار۰ را می خواند و هیچ چیز نمی نویسد.

درگذر از حالت $[5,1,0]$ به حالت $[6,1,1]$ برنامه مقدار ۱ را می خواند و هیچ چیز نمی نویسد.

رفتار محاسباتی p می تواند بوسیله ی یک سیستم صوری $< Q, \sum, \Delta, \delta, q_0, F >$ که بوسیله ی الگوریتم زیر تعریف می شود مجزا شود.

Q : نماینده ی مجموعه ی حالاتی که P می تواند بپذیرد.

δ : نماینده ی مجموعه ی اتنقال هایی که P می تواند بین حالات انجام دهد.

$\sum$: نماینده ی مجموعه ی مقادیر ورودی که P می تواند بخواند.

Δ : نماینده ی مجموعه ی مقادیرخروجی که P می تواند بنویسد.

q_0 : نماینده ی حالت اولیه از p

F : نماینده ی دسته ی حالات پذیرنده ای که P می تواند به آنها برسد.

الگوریتم مجموعه های $\delta, \Delta, \sum$ F ,Q, را با هدایت یک جستجو برای عناصر مجموعه ها تعیین می کند.

گام ۱ : Q به مجموعه ی شامل فقط [= q_0,⊙, . . . ,,1 ⊙], و δ به یک مجموعه ی خالی مقدار دهی اولیه می شوند. q_0 حالت اولیه p و δ جدول انتقال p نامیده می شود.

گام ۲ : اضافه کردن حالت $p=[j,u_1,...,u_m]$ از p به Q اگر برای بعضی

حالات $q = \left[i, v_1, \ldots v_m \right]$ در Q شرایط زیر رخ دهد :

P می تواند بوسیله ی اجرای I_i مقادیر $v_1 \ldots v_m$ در متغیرهایش به I_j با $u_1 \ldots u_m$ در متغیرهایش برسد.

گام ۳ : اضافه کردن $(q, \alpha, (p, \rho))$ به δ اگر p با اجرای یک بخش دستورالعمل واحد بتواند از حالت q در Q به حالت P در Q برود. در حالیکه α را می خواند و ρ را می نویسد.

برای تسهیل نمادگذاری ، $(q, \alpha, (p, \rho))$ به صورت (q, α, p, ρ) نوشته خواهد شد. هر چندتایی در δ یک قاعده ی انتقال p نامیده می شود.

گام ٤ : تکرار گام های ۲ و۳ به شرطی که بیشتر حالات یا بیشتر قوانین گذر بتوانند به δ □ اضافه شوند.

گام ٥ : مقدار اولیه دادن به $\sum$ و Δ و F برای مجموعه های خالی .

گام ٦ : اگر (q, α, p, ρ) یک قاعده ی انتقال در δ و $\alpha \neq \varepsilon$ باشد آنگاه α به $\sum$ اضافه می شود متشابهاً اگر (q, α, p, ρ) یک قاعده ی انتقال در δ باشد و $\rho \neq \varepsilon$ آنگاه ρ به Δ اضافه می شود. هر α در $\sum$ یک نماد ورودی p نامیده می شود . و $\sum$ یک الفبای ورودی نامیده می شود. متشابهاً هر ρ در Δ یک نماد خروجی p و Δ یک الفبای خروجی p نامیده می شود.

گام ۷ : درج کردن هر حالت $\left[i, v_1, \ldots, v_m \right]$ به F در Q برای I_i که یک دستورالعمل پذیرش شرط است. حالات در F حالات پذیرفته شده یا حالات نهایی P نامیده می شوند.

δ یک رابطه است از $Q \times (\sum Y\{\varepsilon\})$ به $Q \times (\Delta Y\{\varepsilon\})$. بعلاوه مجموعه های Q, $\sum, \Delta, \delta$, F همه متناهی هستند. زیرا تعداد بخش های

دستورالعمل در P ، تعداد متغیرها در P و دامنه ی متغیرهای P همگی متناهی هستند.

مثال۲-۲-۲ نمادهای مثال ۲-۲-۱ را در نظر بگیرید . حالت اولیه برنامه P $[1,0,0]$ است . با اجرای اولین دستور العمل برنامه می تواند از حالت $[1,0,0]$ منتقل شود و به یکی از دو حالت $[2,0,0]$ یا $[2,1,0]$ وارد شود . در هردو وضعیت هیچ نماد ورودی خوانده نمی شود از این رو جدول انتقال δ برای P شامل قواعد انتقال $([1,0,0],\varepsilon,[2,1,0],\varepsilon)$ و $([1,0,0],\varepsilon,[2,0,0],\varepsilon)$ است .

متشابها بوسیله ی اجرای دومین دستور العمل برنامه ی P مجبور است از حالت $[2,1,0]$ به حالت $[3,1,0]$ وارد شود در حالیکه چیزی نمی خواند و ۱ می نویسد . از اینرو δ همچنین شامل قاعده ی انتقال $([2,1,0],\varepsilon,[3,1,0],1)$ است.

تعداد حالات در Qبیشتر از {۲×۲×۱۲} نیست . {۰،۱} الفبای ورودی وخروجی برنامه ی P هستند . {[۱و۱و۷]}و [۰و۰و۷] مجموعه ی حالات پذیرش برای P است .

مبدلهای حالت متناهی :

به طور کلی یک سیستم صوری M شامل یک شش تایی $<Q,\Sigma,\Delta,\delta,q_0,F>$ یک مبدل حالت متناهی نامیده می شود اگر در شرایط زیر صدق کند :

Q:یک مجموعه متناهی است که اعضایش حالات M نامیده می شوند .

Σ:یک الفبا است که الفبای ورودی M نامیده می شود . هر نماد در Σ یک نماد ورودی M نامیده می شود .

Δ: یک الفبا است که الفبای خروجی M نامیده می شود.هر نماد در Δ یک نماد خروجی M نامیده می شود .

δ:یک رابطه است از $Q\times(\Sigma U\{\varepsilon\})$ به $Q\times(\Delta U\{\varepsilon\})$ که جدول انتقال Mنامیده می شود . هر چند تایی $(q0,\alpha,(P,\rho))$ یا ساده تر $(q0,\alpha,P,\rho)$ یک قاعده انتقال در δ از M نامیده می شود.

q_0:یک حالت در Qاست که نخستین حالات Mنامیده می شود.

F:یک زیر مجموعه ازQ است که حالات آن حالات پذیرش یا حالات پایانی Mنامیده می شوند.

مثال ۲-۲-۳ چند تایی

$M=<\{q_0,q_1\},\{a,b\},\{1\},\{(q_0,a,q_1,1),(q_0,b,q_1,\varepsilon),(q_0,b,q_1,1),(q_1,a,q_0,\varepsilon)\},q_0,\{q_0\}>$

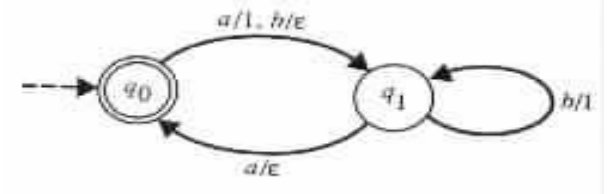
یک مبدل حالت متناهی است . مبدل حالت متناهی حالات q_1,q_0 را دارد . الفبای ورودی Mشامل دو نماد a,b است. الفبای خروجی M شامل یک نماد منفرد۱ است . مبدل حالت متناهی Mچهار قاعده ی انتقال دارد . q_0 نخستین حالت M و تنها حالت پذیرش M است.

قاعده ی انتقال $(q_0,a,q_1,1)$ازM از نماد ورودی a و نماد خروجی ۱ استفاده می کند .

قاعده ی انتقال(q_0,a,q_0,ε)از نماد ورودی a استفاده می کند وهیچ نماد خروجی ندارد.

هر مبدل حالت متناهی $<Q,\Sigma,\Delta,\delta,q_0,F>$می تواند بطور هندسی بوسیله ی یک دیا گرام انتقال به فرم زیر نمایش داده شود . هر حالت Q دیا گرام انتقال یک گره ی متناظر دارد که بوسیله یک دایره نمایش داده می شود . نخستین حالت بوسیله ی یک پیکان که از هیچ جا به گره مشابه اشاره می کندمشخص می شود . هر قاعده ی $<q,\alpha,p,\rho>$ در δبوسیله ی یک یال بر چسب دار با α/ρاز گره بر چسب دار با حالت pنمایش داده می شود. برای تسهیل نماد گذاری یال هایی که مبدا ومقصودشان یکی است با هم ادغام می شوند و بر چسب آنها با کاما از هم جدا می شوند .

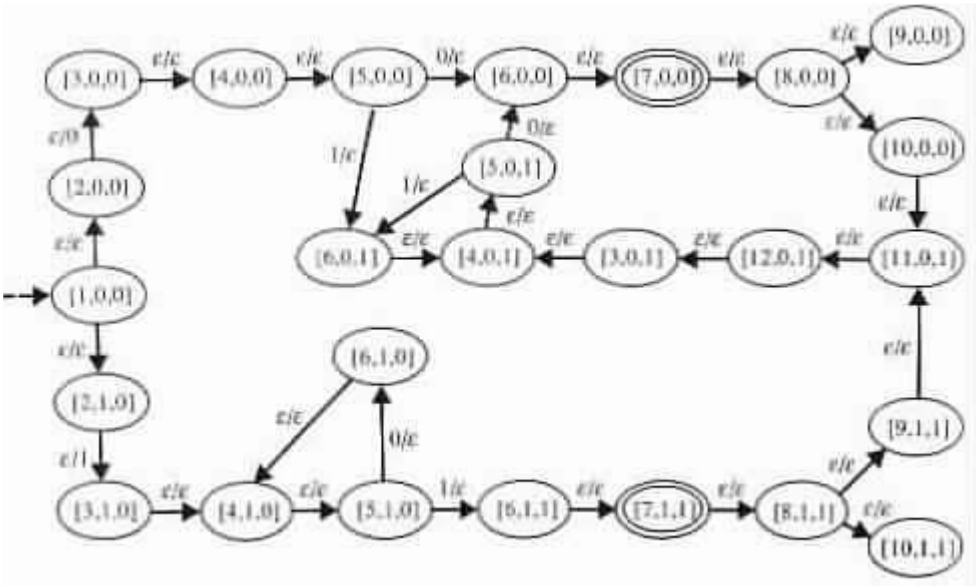
مثال ۲-۲-٤ دیا گرام گذر در شکل ۲-۲-۲



نمایش مبدل حالات متناهی Mاز مثال ۲-۲-۳. برچسب $a/1$ روی یالی از حالت به q_0حالت q_1در دیاگرام انتقال با قاعده ی انتقال (q_0,a,q_1,ε) از Mیکی است . بر چسب b/ε روی یالی از حالت q_0 به حالت q_1 با قاعده انتقال $<q_0,b,q_1,\varepsilon>$مطابق است .

بر چسب $b/1$ روی یالی از حالت q_1 به خودش با قاعده ی انتقال $(q_1,b,q_1,1)$مطابق است.

مثال ۲-۲-۵ دیاگرام گذر در شکل ۲-۲-۳



نمایش مبدل حالت متناهی که برنامه مثال ۲-۲-۱ را توصیف می کند.

شکل و انتقال مبدل های حالت متناهی:

? ?

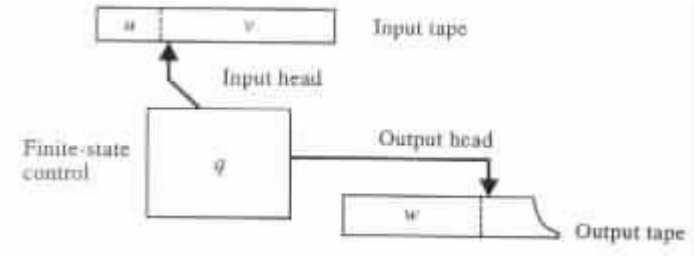


Figure A view of a finite-state transducer as an abstract computing machine. 2.2.4

به طور شهودی، یک مبدل حالت متناهی می تواند یک ماشین محاسباتی انتزاعی در نظر گرفته شود. یک ماشین محاسبه شامل یک کنترل حالت متناهی و یک نوار ورودی، یک رأس ورودی فقط خواندنی، یک نوار خروجی و یک رأس خروجی فقط نوشتنی میباشد. (به شکل ٤. ٢. ٢ نگاه کنید).

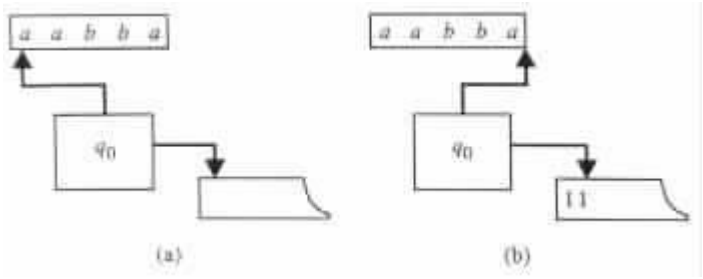
هرنوار به خانه هایی که هر کدام می توانند حاوی یک نماد باشند تقسیم میشود.

نوار ورودی برای نگهداری ورودی uv از M استفاده میشود. رأس ورودی برای دستیابی به نوار ورودی استفاده میشود. نوارخروجی برای نگهداری خروجی W ازM استفاده میشود . و رأس خروجی برای دستیابی به نوار خروجی استفاده میشود. کنترل حالت متناهی برای ثبت حالت Mبه کار می رود.

برروی هر ورودی $a_1 \ldots a_n$ از Σ^* ماشین محاسباتی M تعدادی مجموعه از پیکربندی های ممکن دارد هرپیکربندی یا توصیف لحظه ای از M یک جفت (uqv,w) است بطوریکه q یک حالت در Q است. $uv = a_1 \ldots a_n$ و w یک رشته در Δ^* است.

به طور استدلالی، یک شکل (uqv,w) به ما می گوید که M همراه ورودی uv و بعد از خواندن u به حالتq می رسد و w را نوشته است. بدون اینکه به کلیت خدشه ای وارد شود فرض میشود که Q و Σ متقابلاً گسسته هستند.

? ?



شکل ۲-۲-۵ : ترتیب مبدل حالت محدود شکل ۲-۲-۲

مثال ۲-۲-۶ Mرا مبدل حالت متناهی مثال ۲-۲-۳ قرار دهید(شکل ۲-۲-۲ را نگاه کنید). حالت $aabq_1$ (ba,1)از M می گوید که M به حالت q_1 بعد از خواندن u=aab از نوار ورودی می رسد و w=1 را درون نوار خروجی می نویسد. بعلاوه می گوید که v=ba باقیمانده ی ورودی است. (شکل ۲-۲-۵ را نگاه کنید)

پیکربندی $(q_0, aabba, \varepsilon)$ازM می گوید که M بعد از خواندن هیچ چیز(ε =i.e., u) از نوار ورودی به حالت q_0 می رسد و درنوارخروجی هیچ چیز ننوشته است. بعلاوه شکل می گوید که V=aabba ورودی است که باید به کارگرفته شود.(شکل ۵. ۲. ۲را ببنید).

پیکربندی $(aabbaq_0, 1)$ از M بعد از خواندن همه ی ورودی (ε=v, .i.e) به حالت q_0 می رسد و W=11 را می نویسد. همچنین شکل می گوید ورودی خوانده شده u=aabba است. یک پیکربندی(uqv,w) ازM یک پیکربندی ابتدایی نامیده می شود اگر $q = q_0$ و u=w=ε باشد. یک آرایش ابتدایی می گوید که رأس ورودی درابتدای (سمت چپ ترین موقعیت) ورودی قرار می گیرد.نوار خروجی خالی است و کنترل حالت منتاهی روی حالت آغازی قرارداده میشود.

به یک پیکربندی (uqv,w) ازM یک پیکربندی پذیرش گفته میشود اگر vu=w=ε و q یک حالت پذیرش درF باشد. یک پیکربندی پذیرش می گوید کهM یک حالت پذیرفته شده را بعد از خواندن همه ی ورودی ها بدست می آورد.

مثال ۲-۲-۷ مبدل حالت متناهی M از مثال ۲-۲-۳ (شکل ۲. ۲. ۲ را نگاه کنید)پیکربندی ابتدایی ($q0aabba,\varepsilon$)و پیکربندی پذیرش $(aabbaq_0,11)$ را روی ورودی aabba دارد .(به شکل ۲-۲-۵ a)) و ۲-۲-۵ (b) به ترتیب نگاه کنید.

(aabbaq0,ε) و(aabbaq0,111) همچنین پیکربندی پذیرش M روی ورودی aabba هستند. از سوی دیگر (q_0aabba,ε)فقط پیکربندی ابتدایی M روی ورودی aabba است.

قواعد انتقال M برای معین کردن حرکات ممکن M استفاده میشوند. حرکت از تعدادی قاعده ی انتقال استفاده می کند. یک حرکت روی قاعده ی انتقال (q,α,p,ρ)شامل تغییرحالت کنترل حالت متناهی از q به p است بوسیله خواندن αاز نوار ورودی و نوشتنρ روی نوار خروجی و همچنین بوسیله ی انتقال هد ورودی و خروجی وانتقال |α|و|ρ| موقعیت به سمت راست به ترتیب.

یک حرکت M از پیکربندی C_1 به پیکربندی c_2بوسیله $C_1 \vdash_M c_2$ یاساده تر $C_1 \vdash c_2$ مشخص می شود اگر مشخص شده باشد. یک رشته ی صفر یا حرکات بیشتر M از پیکربندی C_1به پیکربندی c_2 با $C_1 \vdash_M^* c_2$ یا ساده تر $C_1 \vdash^* c_2$ مشخص میشود اگر M مشخص شده باشد.

? ?

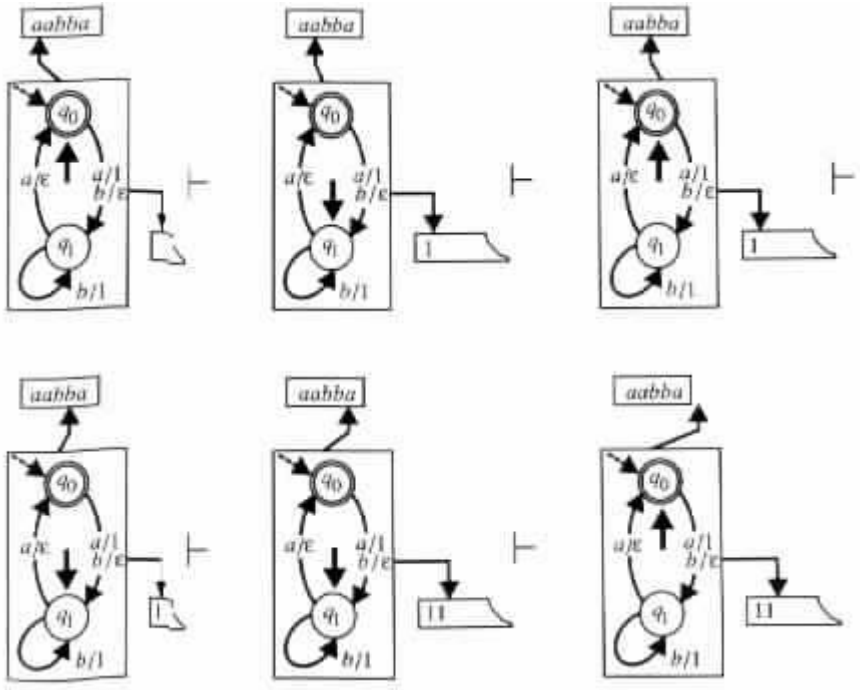


Figure 2.2.6 Sequence of moves between configurations of a finite-state transducer.

$q_0aabba, \varepsilon) \vdash (aq_1abba, 1) \vdash (aaq_0bba, 1) \vdash (aabq_1ba, 1) \vdash (aabbq_1a, 11) \vdash)$

$(aabbaq_0, 11$).

مثال ۲-۲-۸ : M را مبدل حالتهای متناهی مثال ۲-۲-۳ قرار دهید(به شکل ۲،۲،۲نگاه کنید)M روی ورودی , aabba می تواند دنباله زیر را از(۱۱$(aabbaq_0$

$(q_0aabba, \varepsilon) \vdash^*$وحرکات بین پیکربندی هاداشته باشد.(شکل ۲-۲-۶ را نگاه کنید).

$(q_0aabba, \varepsilon) \vdash (aq_1abba, 1) \vdash (aaq_0bba, 1) \vdash (aabq_1ba, 1) \vdash (aabbq_1a, 11) \vdash \vdash$
$(aabbaq_0, 11)$.

دنباله شامل ۵ حرکت است که با یک حرکت $(q_0aabba,\varepsilon) \vdash (aq_1abba, 1)$ روی نخستین قاعده انتقال $(q_0, a, q_1, 1)$از M شروع میشود. درطول حرکت، M یک انتقال از حالq_0 به حالت q_1 دارد در حالیکه a را می خواند و۱ را می نویسد.

دومین حرکت، حرکت $(aq_1abba, 1) \vdash (aaq_0bba, 1)$ روی چهارمین قاعده ی انتقال $(q_1, a, q_0, 0)$از Mاست. در طول حرکت M یک انتقال از حالت q_1به q_2دارد در حالیکه a را می خواند وهیچ چیز نمی نویسد.

دنباله بوسیله ی یک حرکت روی دومین قاعده ی انتقال $(q_0, b, q_1, \varepsilon)$ادامه می یابد و بوسیله ی یک حرکت روی سومین قاعده $(q_1, b, q_1, 1)$دنبال میشود و بعد ازیک حرکت اضافی روی چهارمین قاعده ی انتقال $(q_1, a, q_0, \varepsilon)$پایان می یابد.

دنباله ی حرکات فقط یکی است که می تواند در پیکربندی آغازی شروع شود و در یک پیکربندی پذیرش برای ورودی aabba پایان پذیرد.

طبق تعریف، درهر قاعده ی انتقال (q,a,p,ρ)، $|\alpha|=0$ یا $|\alpha|=1$ است. اگر نماد ورودی در طول حرکاتی که قاعده انتقال استفاده می کند خوانده نشود$|\alpha|=0$ و $|\alpha|=1$ است اگر دقیقاً یک نماد ورودی در طول حرکات خوانده شود، به طور متشابه $|\rho|=0$ یا $|\rho|=1$ به این وابسته است که در طول حرکات هیچ چیزنوشته نمی شود یا دقیقاًیک نماد نوشته میشود.

قطعیت و عدم قطعیت در مبدل های حالت متناهی:

یک مبدل حالت متناهی $M =< Q, \Sigma, \Delta, \delta, q_0, F >$ قطعی است اگر برای هر حالت q در Q و هرنماد ورودی a در Σ اجتماع $\delta(q,a) \cup \delta(q,\varepsilon)$ یک چند مجموعه ای شامل حداکثریک عنصرباشد.

بطوراستدلالی، M قطعی است اگرهر حالت M کاملاً معین کند که آیا یک نماد ورودی روی یک حرکت از حالت خوانده میشودیا نه. و حالت به اضافه ورودی که در یک حرکت مصرف میشود قاعده انتقال استفاده شده را کاملاً معین کند.

یک مبدل حالت متناهی غیر قطعی است اگر شرایط قبلی رخ ندهد.

مثال ۲-۲-۹ دیاگرام انتقال مبدل حالت متناهیM_1 که در شکل۲. ۲. ۲ آورده شده

است قطعی است. درهرحرکت M_1 یک نماد ورودی را مـی خوانـد. قاعـده ی انتقـال استفاده شده منحصراً بوسیله حالت معین میشود و نماد ورودی خوانده میشود.

اگر M_1 نماد ورودی a را درحرکت از حالت q_0 بخواند آنگـاه M_1 بایـد قاعـده ی انتقـال $(q_0, a, q_1, 1)$ را اسـتفاده کنـد. اگـر M_1 نمـاد ورودی b را در حرکـت از حالت q_0 بخواند آنگاه باید قاعده ی گذر $(q_0, b, q_1, \varepsilon)$ را در حرکت استفاده کند.

از سوی دیگر مبدل حالت متناهی M_2 را که به صورت زیر است در نظر بگیرید :

$$M_2 =< Q, \Sigma, \Delta, \delta, q_0, F >$$

از $Q = \{q0, q_1, q_2, q_3\}$ و $\sum$ ={a,b} و Δ={a,b}

$\delta = \{(q_0, a, q_1, a), (q_1, \varepsilon, q_2, a), (q_2, \varepsilon, q_1, b), (q_2, b, q_0, \varepsilon), (q_2, b, q_3, a)\}$ و $F = \{q_3\}$

دیاگرام گذر M_2 در شکل ۷. ۲ . ۲ آورده شده است.

? ?

? ?

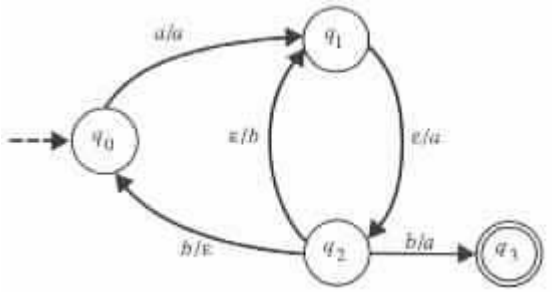


Figure 2.2.7 .A nondeterministic Turing transducer

M_2 یک مبدل حالت متناهی غیر قطعی است.

درحرکت از حالت q_0 مبدل حالت متناهی M_2 باید یک نماد ورودی را بخوانـد. در حرکت از حالت q_1 مبدل حالت متناهی M_2 یک نماد ورودی را نمـی خوانـد. قواعد انتقالی که M_2 می تواند روی حرکت از حالت q_0 و q_1 استفاده کند، منحـصراً

بوسیله ی حالات تصمیم گیری میشود از این رو این حالات منبعی برای تصمیم ناپذیری M_2 نیستند.

دلیل غیر قطعیت M_2 قواعد انتقالی است که در حالت q_2 بوجود آمده است. قواعد انتقال تصمیم نمی گیرند که آیا M_2 مجبور است یک نماد را در انتقال از حالت q_2 بخواند یا نه، آنها معین می کنند که قواعدانتقال در حرکاتی که نماد b را می خواند استفاده شده است.

محاسبات مبدل های حالت متناهی:

محاسبات مبدل های حالت متناهی در یک روش مشابه برنامه ها تعریف میشوند.یک محاسبه ی پذیرش از یک مبدل حالت متناهی M یک دنباله از حرکات M است که روی یک پیکربندی آغازی شروع می شود و دریک پیکربندی پذیرش پایان می یابد. یک محاسبه ی مردود یا رد شده از M یک دنباله از حرکات روی یک ورودی x است برای شرایطی که در زیررخ می دهد:

a . دنباله ی از پیکربندی آغازین M روی x شروع میشود.

b . اگر دنباله متناهی باشد، آنگاه دریک پیکربندی که حرکتی از آن امکان پذیر نیست پایان می یابد.

M. C محاسبه ی پذیرش روی x ندارد.

هرمحاسبه ی پذیرش و هر محاسبه ی مردود از M ، محاسبات M نامیده میشوند.

یک محاسبه یک محاسبه ی توقف است اگر شامل یک تعداد متناهی از حرکات باشد.

مثال۲-۲-۱۰ Mرا مبدل حالت متناهی مثال ۲-۲-۳ قرار دهید (شکل ۲-۲-٦ را ببنید).

روی ورودی aabba مبدل حالت متناهی M یک محاسبه دارد که بوسیله ی دنباله ای از حرکات در مثال ۲-۲-۸ آورده شده است (شکل ۲-۲-٦ را ببنید). این محاسبه یک محاسبه پذیرش است.

متناوباً روی ورودی aab مبدل حالت متناهی M دنباله ی از حرکات را دارد:

$$(q0aab,\varepsilon)\vdash(aq1ab,1)\vdash(aaq0b,1)\vdash(aabq1,1)$$

این دنباله فقط یک امکان ازپیکربندی ابتدایی M روی ورودی abb دارد. آن یک

محاسبه ی مردود است.

دو محاسبه در مثال محاسبات توقف ازM هستند.

طبق تعریف، روی ورودی هایی که بوسیله ی یک مبدل حالت متناهی پذیرفته میشوند، مبدل حالت متناهی می تواند دنباله های قابل اجرا از قواعد انتقالی که با محاسبات بررسی نشده اند داشته باشد.

مثال: ۲-۲-۱۱ بررسی مبدل حالت متناهی M که دیاگرام انتقال آن در شکل ۲-۲-۷ آورده شده است. روی ورودی M, ab محاسبه ی پذیرش دارد که در طول دنباله ی حالات (q_0,q_1,q_2,q_3)حرکت می کند. به طور متشابه M روی ورودی ab یک محاسبه ی پذیرش دارد که در طول حالات q_0,q_1,q_2,q_1,q_2,q_3 حرکت می کند. بهرحال در ورودی ab از میان حالات (q_0,q_1,q_2,q_0) دنباله ی حرکات M ها یک محاسبه از M نیست.

روی ورودی a مبدل حالات متناهی فقط یک محاسبه دارد. محاسبه یک محاسبه ی متوقف نشده است که دنباله ی حالات.... $q_0,q_1,q_2,q_1,q_2,$را ادامه می دهد. از سوی دیگر، روی ورودی aba مبدل تورینگ بی نهایت محاسبات مداوم و غیر مداوم دارد. همه ی محاسبات روی ورودی aba محاسبات مردود هستند.

محاسبات توقف M روی ورودی aba فقط پیشوندab را ازM مصرف می کنند و در بین دنباله های حالات محاسبات غیر توقف M روی ورودی aba محاسبات پذیرش هستند.

محاسبات توقف M روی ورودی aba فقط پیشوند ab را از M مصرف می کنند و در بین دنباله های حالات q_1,q_2,q_3 ..... q_0,q_1,q_2,q_1,q_2 حرکت می کنند. محاسبات غیر توقف M روی ورودی aba ورودی را مصرف می کنند تا اینکه پایان یابند

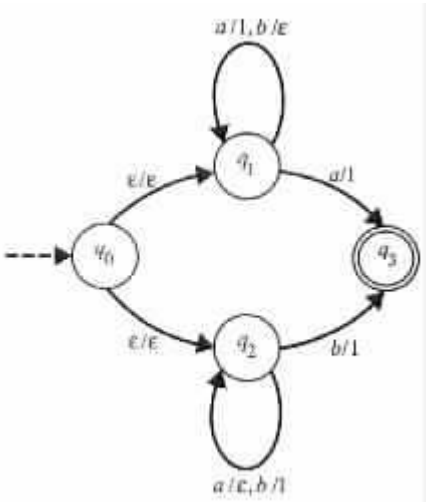
و بین دنباله هایی از حالات... و حرکت q_0,q_1,q_2,q_1,q_2 $q_1,q_2,q_0,q_1,q_2,q_1,q_2$ می کنند.

طبق تعریف هر حرکت درهرمحاسبه باید روی یک قاعده ی انتقال که سرانجام

محاسبات را می پذیرد همه ی ورودی ها را بخواند و پس از آن به یک حالت پـذیرش برسد. هرگاه بیش از یک جایگزینی در مجموعه ی قواعد انتقال وجود داشته باشـد هـر کدام از این جایگزینی ها می توانند انتخاب شوند. بهمین نحـو، هرگـاه هـیچ قاعـده ی انتقالی امکان پذیری شرایط بالا را برآورده نسازد آنگاه هـر کـدام از قواعـد انتقـال مـی توانند انتخاب شوند.این واقعیت اشاره می کند به اینکه ما محاسبات مبدل های حـالات متناهی را مانند اجرا بوسیله ی عوامل فرضی با قدرت جادویی نمایش می دهیم.

یک ورودی x پذیرفته شده یا شناسایی شده بوسیله یک مبدل حالت متنـاهی M نامیـده می شوداگر M یک محاسبه پذیرش روی X داشته باشد. یک محاسبه پـذیرش کـه در یک شکل پذیرش (xq_F, y)خاتمه می یابد یک خروجی Y دارد. فـرض مـی شـود خروجی یک محاسبه مردود تعریف نشده است.

یک مبدل حالت متنـاهی M یـک خروجـی y روی ورودی x دارد اگـر یـک محاسـبه پذیرش روی x با خروجی y داشته باشد. M توقف روی X نامیده می شـود اگـر همـه محاسبات M روی ورودی X محاسبات متوقف شده باشند.



?

Figure 2.2.8 .A nondeterministic finite-state transducer

مثال ۲-۲-۱۲ مبدل حالت های متناهی M که دیاگرام انتقال آن در شکل ۲-۲-۸ آورده شده است، روی ورودی baabb یک دنباله از حرکات دارد که بین حالت q_0 q_1, q_1, q_1, q_1, q_1 حرکت می کند و یک دنباله از حرکات بین حالات $q_0, q_2, q_2, q_2, q_2, q_2$ و یک دنباله از حرکات بین حالات $q_0, q_2, q_2, q_2, q_2, q_3$ دارد دنباله ی حرکات بین حالات $q_0, q_2, q_2, q_2, q_2, q_3$ فقط محاسبه M روی ورودی baabb است.محاسبه یک محاسبه پذیرش است که خروجی ۱۱۱ را فراهم می کند.

M همه ورودی ها را می پذیرد. هر چند مبدل حالت متناهی مثال ۲-۲-۱۱ کاملاً آن ورودی هایی را که فرم ababa.....bab را دارند می پذیرد.

چنانکه در خصوص برنامه ها، معانی مبدل های حالت متناهی به وسیله محاسباتشان توصیف شده اند. در نتیجه رفتار این مبدل ها با رجوع به محاسباتشان طبقه بندی می شود.

برای مثال به یک مبدل حالت متناهی M انتقال دهنده از شکل C_1 به شکل C_2 روی X گفته می شود اگر C_2 به دنبال C_1 در محاسبات مطرح شدهٔ M روی X بیاید. به همین نحو به M خواننده a از ورودی اش گفته می شود اگر a از ورودی در محاسبات مطرح شدهٔ M مصرف شود

مثال ۲-۲-۱۳ مبدل حالت متناهی که دیاگرام انتقال آن در شکل ۲-۲-۸ آورده شده است محاسباتش با یک حرکت روی ورودی baabb شروع می شود که M از حالت q_0 به q_2 می رود. M سپس ٤ حرکت انجام می دهد که baab را مصرف میکند و M را در حالت q_2 میگذارد . در نهایت M با خواندن b از حالت q_2 به q_3 حرکت میکندو در آنجا می ماند.

روابط و زبان های حالت متناهی:

رابطه محاسبه شده با یک مبدل حالت متناهی $M =< Q, \Sigma, \Delta, \delta, q_0, F >$ با R(M) نشان داده میشود و به صورت مجموعه $\{ (x, y) \mid (q_0x,\varepsilon) \vdash^* (xq_F, y) \text{ for } \}$

{ some q_f in F } است. یعنی رابطه محاسبه شده با M تمام جفت هـای (x,y) اسـت که M یک محاسبه پذیرش روی X با خروجیy دارد .

زبان پذیرفته شده یا شناخته شده با M با L(M) نشان داده می شود و مجموعه ی تمـام ورودی هایی است که M می پذیرد یعنی مجموعـه ی { x|(x,y) is in R(M) for some y} است. زبانی تصمیم پذیر به وسیله M گفته می شود اگر M روی تمـام ورودی هـا یعنی روی تمام xها در $\sum^*$ توقف کند .

زبان تولید شده به وسیله M مجموعه تمام خروجـی هـایی اسـت کـه M روی ورودی هایش دارد. یعنی مجموعه ی

{ y | (x, y) is in R(M) for some x }.

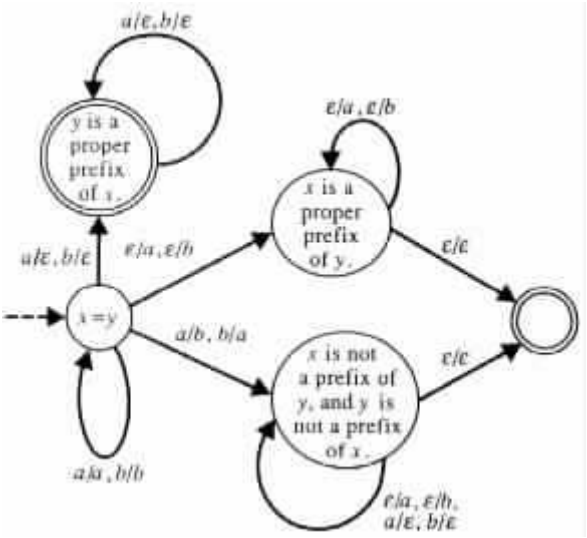
مثال ۲-۲-۱٤ مبدل حالت متناهی غیرقطعی M که دیـاگرام انتقـال آن درشـکل ۲-۲-۸ آورده شده است رابطه زیر را محاسبه می کند :

$R(M)=\{(X, 1^i) \mid x \text{ is in } \{a,b\}^*\}$,

i = تعداد aها در x اگر آخرین نماد در a, x باشد و i= تعداد bها در x اگر آخرین نماد در b ,x باشد}

$\}^*$

ماشین خودکار حالت متناهی M زبان $L(M)=\{a,b\}^*$ را می پذیرد.



?

Figure A finite-state transducer that computes the relation R(M)?=?{ (x, y) | x and y are in {a, b}*, and y≠ x }. 2.2.9

مثال ۲-۲-۱۵ مبدل حالت متناهی غیرقطعی M که دیاگرام آن در شکل ۲-۲-۹ آورده شده است رابطه $R(M)=\{(x,y) \mid x,y \text{ are in } \{a,b\}^*, y \neq x\}$ را محاسبه می کند.

تا زمانیکه M در نخستین حالت "x=y" است خروجی M مساوی بخش ورودی مصرف شده تا اینجاست.

اگر M بخواهد یک خروجی فراهم کند یعنی یک پیشوند مناسب برای ورودی اش آنگاه به محض رسیدن به پایان خروجی، M باید از حالت ابتدایی به حالت «y پیشوند مناسب x است» حرکت کند.

اگر M بخواهد ورودی اش یک پیشوند مناسب برای خروجی اش باشد آنگاه M باید به حالت x'' هست یک پیشوند مناسب y "به محض رسیدن به انتهای ورودی منتقل شود.

در غیر این صورت در بعضی نمونه های غیرقطعی انتخاب شدهٔ محاسبه، M باید به حالت x'' یک پیشوند y نیست و y یک پیشوند x نیست" و برای ساختن یک تفاوت

بین یک جفت مشابه ورودی و نمادهای خروجی منتقل شود.

از مبدل های حالت متناهی تا برنامه های حافظه متناهی:

```
state := q0
do
Accept if an accepting state of M is reached at the end of the ❊/
/❊                                                    ???.input
if F(state) then
if eof then accept
Nondeterministically find the entries of the transition rule ❊/
/❊                  ???.q,□, p, ρ) used in the next simulated move)

/❊?        do in := e or read in until true          /* in :=□
/❊? next_ state := ?                  /* next_ state := p

/❊?        out := ?                          /* out :=ρ
if not?□ (state, in, next_ state, out) then reject
/❊? .Simulate the move ❊/
if out≠e then
write out
state := next_ state
until false
```

?

Figure A table-driven finite-memory program for simulating a finite-state transducer. 2.2.10

بحث قبلی به ما نشان داد که الگوریتمی وجود دارد که هر برنامه حافظه متناهی معین را درون یک مبدل حالت متناهی معادل ترجمه می کند یعنی درون یک مبدل حالت متناهی که همان رابطه را به عنوان برنامه محاسبه میکند. به طور عکس الگوریتمی

وجود دارد که یک برنامه حافظه متناهی هم ارز را از هر مبدل حالت متناهی معین نتیجه می دهد. برنامه می تواند یک برنامه جدولی (Table-driven) باشد که به یک مبدل حالت متناهی معین $M =< Q, \Sigma, \Delta, \delta, q_0 F >$ در حالت شرح داده شده ۲,۲,۱۰ شباهت دارد.

برنامه از یک متغییر state برای ثبت حالتM دریک حرکت معین استفاده می کند همچنین از یک متغیر in برای ثبت ورودی M که در یک حرکت معین مصرف می شود و از یک متغییر next – state برای ثبت حالت M که در یک حرکت معین به دست آمده و یک متغیرout برای ثبت خروجی M که در یک حرکت معین نوشته شده نیز استفاده می کند.

برنامه یک شبیه سازی M را به وسیله مقدار دهی اولیه حالت متغیر با حالت اولیه q_0 از M آغاز می کند. سپس M داخل یک حلقه ی نامتناهی می شود.
برنامه هر تکرار حلقه را به وسیله چک کردن این که آیا یک حالت پذیرش از M در انتهای ورودی به دست آمده است آغاز می کند. اگر به چنین حالتی رسیده است برنامه در یک پیکربندی پذیرش متوقف می شود. در غیر این صورت برنامه یک حرکت منفرد را از M شبیه سازی می کندگزارهٔ F برای تصمیم گیری دربارهٔ اینکه آیا حالت رخ داده یک حالت پذیرش است استفاده می شود.
شبیه سازی هر حرکت M در یک حالت غیرقطعی انجام می شود برنامه مقداری را برای متغیر in تخمین می زند چون که باید در حرکت شبیه سازی حالتی برای next – state که M داخل می شود و مقداری برای متغیر out که برنامه می نویسد خوانده شود. سپس برنامه گزارهٔ δرا برای بازبینی این که مقادیر تخمین زده شده مناسب هستند و ادامه دادن بر طبق نتیجه بازبینی استفاده می کند.
دامنه متغیرهای برنامه برابر $Q \cup \Sigma \cup \Delta \cup \{e\}$ در نظر گرفته می شود در این جا e یک نماد جدید است که در $Q \cup \Sigma \cup \Delta$ نیست وبرای مشخص کردن رشته تهی

$\mathcal{E}$در نظر گرفته می شود .

در برنامه جدولی (table – driven)، F یک گزاره است کـه مقـدارش درسـت در نظـر گرفته می شود وقتیکه و فقط وقتیکه ورودیهای با یک قاعده انتقال M مطابق باشد.

برنامه هایی که با مبدل های حالت متناهی مختلف مطابقند در دامنه های متغیرهایشان و در درستی انتساب ها برای گزاره های F و δفرق دارند.

زمانی که مبدل حالت متناهی M قطعی باشد ،الگوریتم می تواند به راحتی به یک برنامه حافظه متناهی قطعی تبدیل شود.

? ?

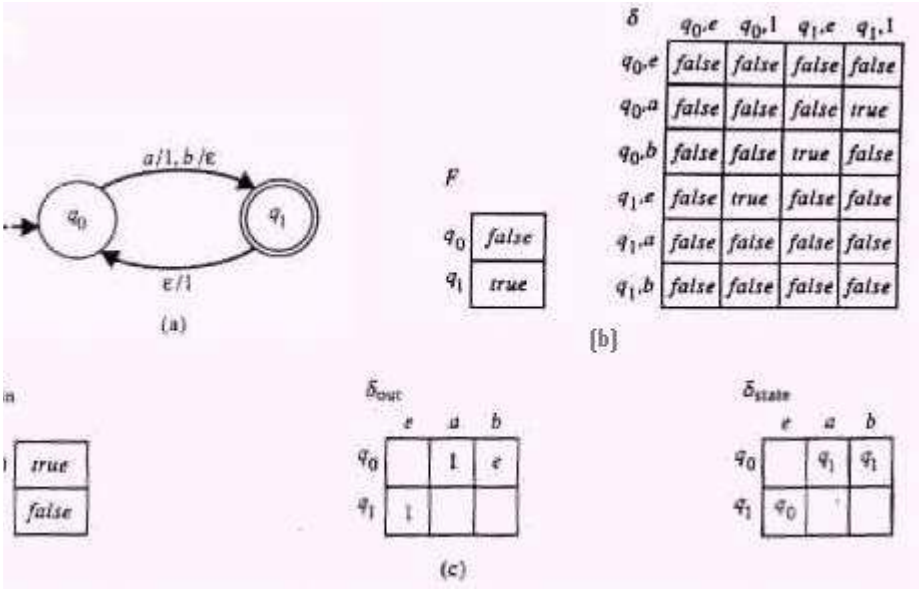


Figure 2.2.11 a) A finite-state transducer M. (b) Tables for a table-driven) program that simulates M. (c) Tables for a deterministic table-driven program that simulates M.

مثال ۲-۲-۱٦ مبدل حالت متناهی شکل ۲-۲-۱۱ (a)

شکل ۲-۲-۱۱ (a): یک مبدل حالت متناهی (b) . M) جداول برای یک برنامه جدولی که M را شبیه سازی می کند. (C) جداول برای یک برنامه جدولی قطعی که M را شبیه سازی میکند.

برنامه در شکل ۲-۲-۱۰ دامنه متغیرهای {a,b,1,q0,q1,e}را دارد. درستی مقادیر گزاره F و δبه وسیله جدولهای مطابق شکل ۲-۲-۱۱ مشخص می شوند.

برنامه همچنین می پذیرد که برای F و δ پارامترهایی وجود دارند که تفاوت آنها در جدول مشخص می شود روی چنین پارامترهایی گزاره هایی تعریف نشده در نظر گرفته می شود.

?

```
state := q0
do
if F(state) then
if eof then accept
if not?□ in(state) then
in := e
if □in(state) then
read in
(next_ state := □state(state, in

(out := □out(state, in

if out ≠e then
write out
state := next_ state
until false
```

?

Figure A table-driven, deterministic finite-memory program for simulating a deterministic finite-state transducer. 2.2.12

مبدلهای حالت متناهی همچنین می تواند به وسیله برنامه جدول قطعی شکل ۲,۲,۱۲

شبیه سازی شوند.

F یک گزاره مانند قبل در نظر گرفته می شود $\delta_{STATE}, \delta_{OUT}, \delta_{IN}$ فرض می شود که به وسیله تطابق جدول ها از شکل (2.2.11 (c مشخص شده اند.

گزاره δ_{IN} ,تعیین می کند که آیا یک نماد ورودی روی حرکت از یک حالت معین خوانده می شود. تابع δ_{OUT} خروجی خوانده شده را در هر حرکت شبیه سازی تعیین میکند و δ_{STATE} حالت به دست آمده را در دو حالت شبیه سازی شده تعیین می کند.

```
state := q0
do
if state = q0 then
do
read in
if in = a then
do
state := q1
out := 1
write out
until true
if in = b then
state := q1
until true
if state = q1 then
do
if eof then accept
state := q0
out := 1
write out
until true
until false
```

Figure 2.2.13 A non-table-driven deterministic finite-memory program that simulates the deterministic finite-state transducer of Figure? (a)2.2.11.

مبدل حالت متناهی قطعی می تواند به وسیله یک برنامه حافظه متناهی غیر جدولی

(non- table- driven) که فرم آن در شکل ۲,۲,۱۳ نشان داده شده است شبیه سازی شود.

شکل ۲,۲,۱۳: یک برنامه حافظه متناهی قطعی غیرجدولی که مبدل حالت متناهی قطعی شکل (2.2.11 (a را شبیه سازی می کند.

در چنین حالتی به خاطر دستورالعمل های if شرطی برنامه صریحاً نتایج F و $\delta_{STATE}, \delta_{OUT}, \delta_{IN}$ را ثبت می کند.

چنین برمی آید که مبدلهای حالت متناهی برنامه های حافظه متناهی را مشخص می کنند و همچنین آنها می توانند برای طراحی و تجزیه برنامه های حافظه متناهی استفاده شوند. در نتیجه مطالعه رفتار زیر برای مبدل های حالت متناهی، برای برنامه های حافظه متناهی هم انجام می شود.

فواید مبدل های حالت متناهی به شرح زیرهستند:

a. نمایش آثار گرافیکی آنها، که در خیلی موارد طبیعی تر از برنامه های حافظه متناهی است.
b. کوتاهی آنها، زیرا مبدل های حالت متناهی انتزاع هایی هستند که جزئیات آنها از مطالعه بی ربط چشم پوشی می کنند.
c. اتمام وابستگی خروجی ها روی ورودی ها.

۳-۲ ماشین های باحالات متناهی و زبانهای منظم

ماشین های باحالات متناهی

غیر قطعیت درمقابل قطعیت در ماشینهای با حالات متناهی

ماشین های باحالات متناهی وگرامرهای نوع۳

گرامرهای نوع ۳ وگرامرهای منظم

زبانهای منظم وعبارات منظم

محاسبات برنامه ها بواسطه ورودیهایشان هدایت شده اند . خروجی ها فقط نتایجی از محاسبات هستندو آنها هیچ اثری برروی مسیری که محاسبات طی میکنند ندارند. در نتیجه ، بنظرمیرسد که میتوان آنقدر درباره مبدل های باحالات متناهی ، یا بطور معادل ، درباره برنامه های با حافظه محدودمطالعه کرد درست وقتیکه خروجیهایشان نادیده گرفته میشوند.مزیت یک مطالعه هدایت شده درباره چنین مبدلهای باحالت متناهی پیاده شده دراستدلال (منظره) ساده شده ای که آنها میپذیرند است .

ماشینهای با حالات متناهی

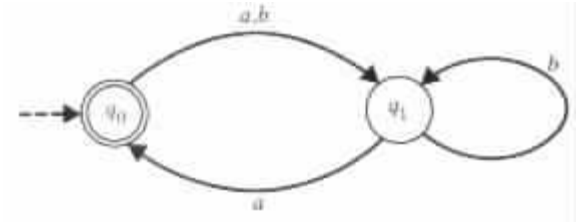
یک مبدل باحالات متناهی که اجزای خروجیش نادیده گرفته میشوند ماشین خودکار باحالت متناهی نامیده میشود. تفصیلا یک ماشین باحالت متناهی M یک چندتایی $Q, >$ $\Sigma,\delta, q_0, F>$ است ،که درآن $Q, \Sigma, q_0,$ و F باتوجه به مبدلهای باحالات متناهی تعریف میشوند تابع انتقال یک رابطه از $Q\times\{\varepsilon\}(U\Sigma()$ به Q است .

نمودارهای انتقالمیتوانند مشابه با کاربردشان برای نمایش مبدل های باحالات متناهی برای نمایش دادن ماشینهای باحالات متناهی نیزاستفاده شوند .

تنها تفاوت آن است که درموردماشینهای باحالات متناهی ، یالی که با قاعده انتقال (P,α,P) مطابق است بوسیله رشته α برچسب دار میشود .

مثال ۲-۳-۱ ماشین باحالات متناهی که بوسیله مبدل باحالات متناهی شکل ۲-۲-۲ بوجودآمده ، $Q, \Sigma,\delta,q0,F>$ که درآن $Q =\{q_0,q_1\}, \Sigma = \{a,b\}, \delta =\{(q_0, a, q_1), (q_0, b, q_1), (q_1, b, q_1), (q_1, a, q_0)\}, F = \{q_0\}$.

نمودارانتقال درشکل۲-۳-۱۱ ماشین باحالات متناهی را نمایش میدهد.



شکل ۲-۳-۱ یک ماشین باحالات متناهی که به مبدل باحالات متناهیِ شکل۲-۲-۲ مربوط میشود .

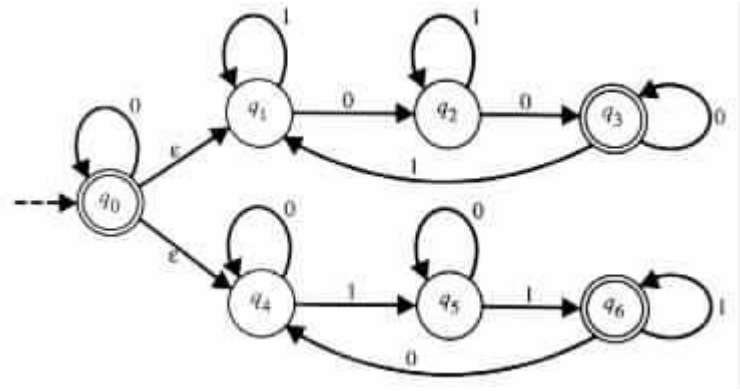
ماشین باحالات متناهیِM قطعی گفته میشود اگر ، برای هرحالت q درQ وبرای هرنماد a در Σ ، اجتماع δ (q,ε) U δ(q,a) یک چندمجموعه ای باشدکه حداکثریک عنصرراشامل شود . اگریک ماشین باحالت متناهی قطعی نباشد ماشین باحالات متناهی غیرقطعی گفته میشود.

یک قاعده انتقال(q,α,p) ازماشین باحالات متناهی یک قاعده گذارε گفته میشود اگر α = ε . یک ماشین باحالات متناهی بدون هیچ قاعده انتقالε راماشین باحالات متناهیِ مستقل ازε (بدون ε) گویند .

مثال ۲-۳-۲ ماشین باحالات متناهیِ

$M_1 = <\{q_0, \ldots, q_6\}, \{0, 1\}, \{(q_0, 0, q_0), (q_0, ε, q_1), (q_0, ε, q_4), (q_1, 0, q_2), (q_1, 1, q_1), (q_2, 0, q_3), (q_2, 1, q_2), (q_3, 0, q_3), (q_3, 1, q_1), (q_4, 0, q_4), (q_4, 1, q_5), (q_5, 0, q_5), (q_5, 1, q_6), (q_6, 1, q_6), (q_6, 0, q_4)\}, q_0, \{q_0, q_3, <\{q_6$

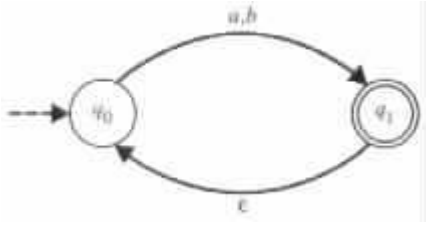
نمودارِانتقال M_1 درشکل ۲-۳-۲ نمایش داده شده است .



شکل ۲-۳-۲ یک ماشین باحالت متناهیِ غیرقطعی

M_1 بعلت قواعد انتقالی که ازحالت q_0 سرچشمه میگیرندقطعی نیست . یکی از قواعد انتقال احتیاج به خواندن یک ورودی دارد ، درحالیکه دو قاعده انتقال دیگربه خواندن هیچ ورودی نیازندارند . بعلاوه ، وقتیکه قاعده تولید ($q_0 , 0 , q_0$) نادیده گرفته شود M_1 غیرقطعی هم هست ، زیرا M_1 بطورموضعی نمیتواند تصمیم بگیرد که درانتقالات ازقواعد انتقال دیگری که ازحالت q_0 سرچشمه میگیرند پیروی کند .

ماشین باحالات متناهی M_2 درشکل ۲-۳-۲ یک ماشین باحالات متناهیِ قطعی است .



شکل ۲-۳-۲ یک ماشین باحالات متناهی قطعی

M_1 ،دو قاعده گذار ε دارد و M_2 یکی دارد .

یک پیکربندی ، یا یک توصیف آنی ، ازماشین باحالات متناهی ،uqv منحصربفردی است ، که q یک حالت درQ است وuv رشته ای درΣ* است . یک پیکربندی ،پیکربندی ابتدایی گفته میشود اگر ε=u وq حالت ابتدایی باشد . یک پیکربندی ، پذیرش یا پیکربندی نهایی گفته میشود اگر ε=v و q یک حالت پذیرش باشد. بدون هیچ لطمه ای به اصل کلی فرض شده است که Q وΣ متقابلا مجزا هستند .

تعاریف دیگر، نظیر اینها از ⊢* ,M ⊢, ⊢M و پذیرش ،شناسایی وتصمیم پذیری ازیک زبان بوسیله ماشین با حالات متناهی مشابه به تعاریف داده شده برای مبدلهای باحالات متناهی هستند .

غیر قطعیت درمقابل قطعیت درماشین های باحالات متناهی

با غیر قطعیت بواسطه قضیه زیرین ، نمیتواند به توانایی تشخیصِ ماشین های باحالات متناهی اضافه شود ، مگراینکه بتواند اختصارو کوتاهی آنها کمک کند . اثباتِ قضیه یک الگوریتم برای ایجادکردن یک ماشین باحالات متناهی قطعیِ معادل که ازهرn حالتِ معینِ ماشین باحالات متناهی ، باحداکثر 2^n حالت ، فراهم میکند .

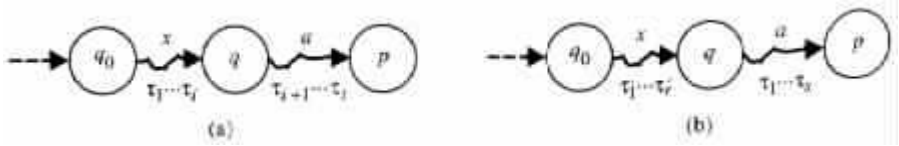
قضیه ۲-۳-۱ اگریک زبان بوسیله یک ماشین باحالات متناهی پذیرفته شود ، آنگاه آن زبان بوسیله یک ماشین باحالات متناهی قطعی که هیچ قاعده گذار ε ی ندارد پذیرفته میشود .

اثبات هرماشین باحالات متناهی به صورت $M = <Q, \Sigma, \delta, q_0, F>$ را در نظر بگیرید . A_x رامجموعه ای ازهمه ی حالاتی که M میتواند ازحالت ابتدایی q0ش ،بوسیله دنباله هایی ازانتقالات که رشته x را پیمایش میکند ،حصول شود میگیریم ، به عبارت دیگر،

$\{q \mid q_0x \vdash^* xq\}$. پس یک ورودی w بوسیله M پذیرفته میشود اگروتنها اگر A_w شامل یک حالت پذیرش(نهایی) باشد .

اثبا ت به این موضوع تاکید میکند که A_{xa} دقیقا شامل آن حالاتی است که میتوانند ازحالاتی در A_x حصول شوند ، بوسیله دنباله هایی ازقواعد انتقالی که a را پیمایش میکند ، بعبارت دیگر ، $\{ap \vdash^* qa$ و q در A_x است$\mid p\} = A_{xa}$.

بطورخاص ، اگر p یک حالت در A_{xa} باشد ، آنگاه بواسطه تعریف یک دنباله از قواعدانتقال $\tau_1, \ldots, \tau_t$ وجوددارد که M را از حالت ابتدایی q_0 به حالت p درحین پیمایش xa میبرد . این دنباله باید یک پیشوند $\tau_1, \ldots, \tau_i$ داشته باشد که M را از q_0 به حالت میانی q میبرد . (شکل ۲-۳-٤(a) رامشاهده کنید) .



شکل ۲-۳-٤ دنباله هایی ازقواعدگذار که xa را پیمایش میکند .

درنتیجه ، q در A_xاست و زیردنباله ی $\tau_{i+1}, \ldots, \tau_t$ ازقواعد انتقال M راازحالت q به حالت p درحین پیمایش a میبرد ازسوی دیگر ، اگر q در A_x باشد واگر p یک حالت است که ازحالت q بوسیله یک دنباله ی $\tau_1, \ldots, \tau_s$ ازقواعد انتقالی که a را پیمایش میکند قابل دستیابی است ، آنگاه حالت p در A_{xa}است . درچنین مواردی ، اگر $\tau'_1, \ldots, \tau'_r$ یک دنباله است از قواعد انتقال که M را از حالت ابتدایی q_0 به حالت qدرحین پیمایشx میبرد ،آنگاه M میتواند بوسیله دنباله ی $\tau'_1, \ldots, \tau'_r, \tau_1, \ldots, \tau_s$ . . .

- ازقواعد انتقالی که xa راپیمایش میکنند ازحالت q_0به حالت p برسد . (شکل ۲-۳-٤ (b) رامشاهده کنید) .

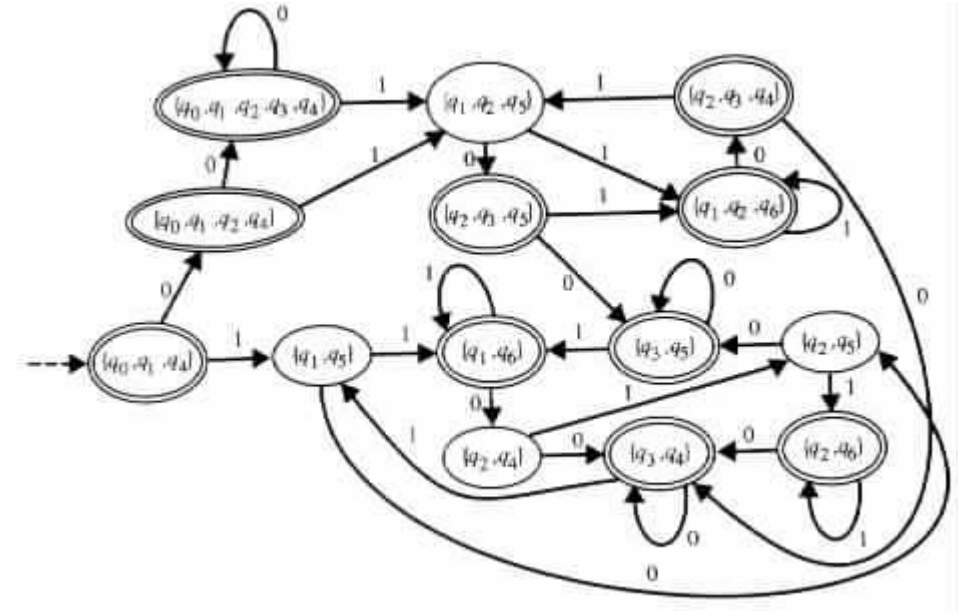
درنتیجه ، مشخص میشود که اگر $a_1...a_n$ بوسیله M پذیرفته شود ، یکی ازالزامات دنبال کردن دنباله ی A_{a1-an} , ... , A_{a1a2}, A_{a1} , A ازمجموعه ای ازحالات ، که درآن هر $Aa1-ai+1$ منحصرا ازروی ai_A_{a1} و ai+1 تعیین میشود . بنابراین ، یک ماشین باحالات متناهی قطعی M' از تعقیب شکل زبانی مشخص میشودکه بوسیله M پذیرفته شده است .

مجموعه حالاتی از M' برابراست با { A | A } زیرمجموعه ازQ است , و $A=A_x$ برای تعدادی x درΣ^* } . ازآنجاییکه Q متناهی است ، استنباط میشود که Q تنها یک تعداد متناهی اززیرمجموعه های A داردو در نتیجه M' نیزتنها تعدادی متناهی ازحالات دارد. حالت ابتدایی M' زیرمجموعه ایست ازQ که مساویست با A_ε . حالات پذیرش(نهایی) M' آن حالاتی از M'هستند که حداقل یک حالت پذیرش از M را شامل شوند. مجموعه ی زیر جدول انتقال M' است .}

{ (A', a, A) |A و A' حالاتی از M' هستند ، a درΣ است وA' مجموعه ای ازحالات است که ماشین باحالات متناهی M میتواند با پیمایش a ازرویِ آن حالاتی که در A هستند بدست آید }

برطبقِ تعریف ، M' هیچ قاعده ی انتقال ε یی ندارد . علاوه براین ، M' قطعی است زیرا ، برای هر x در Σ^* وهر a درΣ ، مجموعه ی A_{xa}منحصرا ازروی مجموعه ی A_x و نماد a تعریف میشود .

مثال ۲-۳-۳ M راماشین باحالات متناهی گرفتیم که نمودار انتقال آن درشکل ۲-۳-۲ داده شده است . نمودار انتقال درشکل ۲-۳-۵



شکل ۲-۳-۵ یک نمودارانتقال ازیک ماشین باحالات متناهیِ قطعی بدو ن ε ی که معادل است با ماشین باحالات متناهیی که نمودار انتقال آن درشکل ۲-۳-۲ داده شده است .

یک ماشین باحالات متناهی قطعی بدون ε که معادل است باMرانمایش میدهد . استفاده کردن از مجموعه اصطلاحاتی از اثبات قضیه ۲-۳-۱

$$\{A_\varepsilon = \{q_0, q_1, q_4\}, A_0 = \{q_0, q_1, q_2, q_4\}$$

$$.\{q_0, q_1, q_2, q_3, q_4\} = A_{00} = A_{000} = \ldots = A_{0\_}.$$

A_ε مجموعه ای از تمام حالاتی است که M بدو ن خواندن هیچ ورودی میتواند به آن برسد . q_0 درA_ε است زیرایک حالت اولیه ازM است. q_1 و q_4در A_ε هستندزیرا M قاعده انتقالε ی دارد که حالت اولیه q_0 را ردکرده(ترک میکند) وبه ترتیب به حالات q_1 وq_4 وارد میشود .

A_0 مجموعه ای از تمام حالاتی است که M فقط میتواند بوسیله خواندن ۰ ازآن حالاتی که در A_ε هستندحاصل شود . q_0 در A_0 است زیرا q_0 در A_ε است وM قاعده انتقال $(q_0, 0, q_0)$ را دارد. q_1 در A_0است زیرا q_0 درA_ε است و M میتواند زوج $(q_0, 0, q_0)$ و (q_0,ε, q_1) راازقواعدگذاربرای رسیدن به q_1 ازروی q_0 تنهاباخواندن ۰ بکاربرد . q_2 درA_0 است زیرا q_0 درA_ε است وM میتواندزوج (q_0,ε, q_1) و $(q_1, 0,)$ q_2) ازقواعد انتقال را برای رسیدن به q_2 ازروی q_0 تنها باخواندن ۰ استفاده کند.

نتیجه ی قضییه ی آخر نمیتواند به مبدلهای باحالات متناهی تعمیم یابد ،زیرا مبدلهای باحالات متناهی قطعی فقط میتوانند توابع رامحاسبه کنند ، درصورتیکه مبدلهای باحالات متناهی غیر قطعی روابطی که تابع نیستند رانیزمیتوانندمحاسبه کنند ، برای مثال ، رابطه $\{(a, b), (a, c)\}$. درحقیقت ، توابعی نیز وجوددارندکه میتوانندبوسیله مبدلهای باحالات متناهی غبرقطعی محاسبه شوند اما نمیتوانند بوسیله مبدلهای باحالات متناهی قطعی محاسبه شوند . $R = \{ (x0, 0^{|x|}) \mid x$ یک رشته در$\{۰،۱\}^*$ است $\}$ U $\{ (x1, 1^{|x|})$یک رشته در $\{۰،۱\}^*$ است $\}$ یک مثال ازچنین تابعی است . تابع نمیتواند بوسیله یک مبدل باحالت متناهی قطعی محاسبه شود زیرا هرمبدل باحالت متناهی قطعی M شرط ذیل را که تابع ازآن بهره ای نبرده است ارضاءمیکند : اگر x_1 یک پیشوند از x_2 است و M، x_1و x_2را میپذیرد ، آنگاه خروجیی ازM برروی ورودی x_1 یک پیشوند ازخروجیی از M برروی ورودی x_2 است . (تمرین ۲,۲,۵)

ماشین باحالات متناهی وگرامرهای نوع ۳

دونتیجه زیرین براین دلالت دارد که یک زبان بوسیله یک ماشین باحالت متناهی پذیرفته میشود اگروتنها اگر یک زبان نوع ۳ باشد . اثبات اولین نتیجه نشان میدهد که چطورگرامرهای نوع ۳ میتوانند محاسباتی ازماشینهای باحالات متناهی را شبیه سازی کنند .

قضیه ۲-۳-۲ ماشین باحالات متناهی فقط زبانهای نوع ۳ را میپذیرد .

اثبات :هرماشین باحالات متناهی بصورت $M = <Q,\Sigma,\delta, q_0, F>$ رادرنظر بگیرید . ازقضیه ۲-۳-۱میتوان برداشت کرد که M یک ماشین باحالات متناهی بدو نε است . بدون هیچ لطمه ای به اصل کلی ، میتواند برداشت شود که هیچ قاعده انتقالی، M راوقتیکه حالت یک حالت پذیرش است به حالت اولیه اش نمیبرد . (اگرموردی چنین نباشد ، آنگاه میتوان یک حالت جدید q'_0 به Q اضافه کرد ، حالت جدید q'_0 هردو حالت اولیه و پذیرش رابوجودمی آورد ، و یک قاعده انتقال جدید(q'_0,α, q) رابرای هرقاعده انتقال به شکل (q_0,α, q) که درδ است به δاضافه میکند .)

$G = <N,\Sigma, P, [q_0]>$یک گرامرنوع ۳ گفته میشود اگر N یک نمادغیرپایانی [q] برای هرحالت q درQ دارد و P قواعدتولید زیررا دارد.

a .قاعده تولید به شکل $[q] \rightarrow a[p]$ برای هر قاعده انتقال(q, a, p) درجدول انتقالδ

b . قاعده تولید به شکل $[q] \rightarrow a$ برای هر قاعده انتقال (q, a, p) درδ چنانکه P یک حالت پذیرش درF باشد .

c .قاعده تولیدی به شکل $[q_0] \rightarrow \varepsilon$ است اگرحالت اولیه ی q_0 یک حالت پذیرش در F باشد .

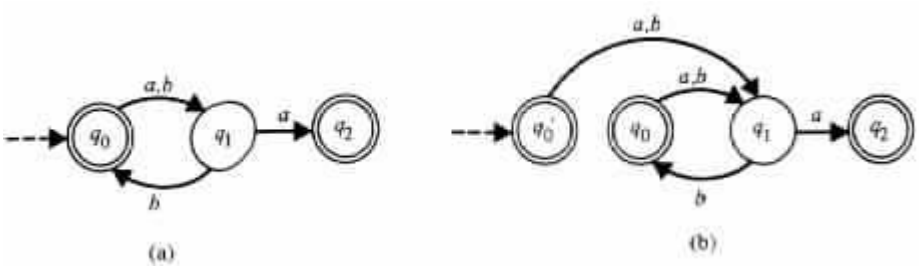
گرامر Gبرشبیه سازیِ محاسباتی ازماشین باحالت متناهی M بنا شده است . G ازمیان نمادهای غیرپایانی حالاتی ازM راثبت میکند . بویژه ، G نمادشروع $[q_0]$ش را بکارمیبرد تایک شبیه سازی ازM درحالت q_0 راآغازکند . G یک قاعده تولید به شکل $[q] \rightarrow a[p]$ رابنابر شبیه سازی کردن یک حرکت از M ازروی حالت q به حالت p بکار میبرد . درچنین کاربردی ازقاعده تولید ، G نماد a را که M درحرکت متناظرمیخواند تولیدمیکند. G یک قاعده تولید به شکل $[q] \rightarrow a$ رابه جای قاعده تولیدی به شکل [q]

$a[p \rightarrow$] بکارمیبرد، وقتیکه آن میخواهدیک شبیه سازی را دریک حالت پذیرش p پایان دهد .

بااستقراء برروی n ، یک رشته $a_n ... a_1a_2$ راکه اشتقاقی به شکل $a_n... a_1a_2 a_{n-1}[q_{n-1}] \Rightarrow ... \Rightarrow [a_1a_2 a_1[q_1] \Rightarrow a_1a_2[q_2$

$[q] \Rightarrow$درG داردرادنبال میکند اگروتنهااگرM دنباله ای ازحرکات به شکل $qa_1a_2... a_n \vdash a_1q_1a_2... a_n \vdash a_1a_2q_2a_3...a_n \vdash a_1... a_{n-1}q_{n-1}a_n \vdash a_1a_2... a_nq_n$ را برای برخی حالات پذیرش q_n دارد . تناظربالا بخصوص برای $q = q_0$ برقراراست . بنابراین $L(G) = L(M)$.

مثال ۲-۳-٤ ماشین باحالات متناهی M_1 که نمودار انتقال آن درشکل ۲-۳-٦ (b) داده شده است .



شکل ۲-۳-٦ دوماشین باحالات متناهیِ معادل

ماشین باحالات متناهیِ قطعی بدون ε است . M_1 برای یک شبیه سازی مستقیم بوسیله یک گرامر نوع ۳ مناسب نیست زیرا حالت اولیه q_0ش هم یک حالت پذیرش وهم یک حالت نهایی ازیک قاعده انتقال است . بدون اصلاحاتی در M_1 الگوریتمی که گرامر Gتولیدخواهدکرد قاعده تولید $[q_0] \varepsilon \rightarrow$ راایجاد میکند زیرا q_0 یک حالت پذیرش است وقاعده تولید $[q_1] \rightarrow b[q_0]$ بواسطه قاعده انتقال (q_1, b, q_0). یک چنین زوجی ازقواعد

تولید نمیتوانند دریک گرامر نوع ۳ باهم وجودداشته باشند .

M_1معادل است با ماشین باحالات متناهیM_2 ،که نمودار انتقال آن درشکل ۲-۳-٦ (a) داده شده است . گرامر نوع ۳ $G = <N, \Sigma, P, [q'_0]>$ زبان $L(M_2)$ را تولید میکند ، اگر $\Sigma = \{a, b\}$, $N = \{[q'_0], [q_0], [q_1], [q_2]\}$ و Pشامل قواعد تولید زیر باشد .

$$
\begin{array}{lcl}
[q'_0] & \rightarrow & \epsilon \\
 & \rightarrow & a[q_1] \\
 & \rightarrow & b[q_1] \\
[q_0] & \rightarrow & a[q_1] \\
 & \rightarrow & b[q_1] \\
[q_1] & \rightarrow & b[q_0] \\
 & \rightarrow & b \\
 & \rightarrow & a[q_2] \\
 & \rightarrow & a
\end{array}
$$

محاسبات پذیرش $q'_0abaa \vdash aq_1baa \vdash abq_0aa \vdash abaq_1a$ $abaaq_2$ از M_2روی ورودی abaaبوسیله اشتقاق

$q'_0] \Rightarrow a[q_1] \Rightarrow ab[q_0] \Rightarrow aba[q_1] \Rightarrow abaa]$ ازگرامر شبیه سازی میشود .

قاعده تولید $[q_1] \rightarrow a[q_2]$ میتواند بدون تغییردادن زبان تولیدشده از گرامرحذف شود .

قضیه بعدی نشان میدهد که عکس قضیه ۲-۳-۲ نیزبرقراراست . اثبات نشان میدهد که چطورماشین باحالات متناهی میتواند اشتقاقهایی ازگرامرنوع ۳ را دنبال کند .

قضیه ۲-۳-۲ هرزبان نوع۳ بوسیله یک ماشین باحالات متناهی پذیرفته میشود .

اثبات هرگرامرنوع ۳ به صورت $G = <N, \Sigma, P, S>$ را درنظربگیرید . ماشین باحالات متناهی $M = <Q, \Sigma, \delta, q_S, F>$ زبانی که G تولیدمیکند رامیپذیرد اگر δ , q_S , Q و F بصورت زیرتعریف شوند .

M یک حالت q_A درQ برای هر نمادغیرپایانی A درN دارد . بعلاوه ، Q یک حالت متمایز که q_f نامیده شده نیزدارد . حالت q_S ازM که به نمادشروعS مربوط است ، به عنوان حالت اولیه ای از M معین شده است . حالت q_f ازM تنها حالت پذیرش ازM گرفته شده است ، بعبارت دیگر ، $F = \{q_f\}$.

M یک قاعده انتقال درδ دارد اگروتنها اگر قاعده انتقالی را به یک قاعده تولید ازG مربوط کند. هرقاعده انتقال به شکل(q_A, a, q_B) در δ به یک قاعده تولید به شکل $A \to aB$ درG مربوط میشود . هرقاعده انتقال به شکل (q_A, a, q_f) رادرδ به یک قاعده تولید به شکل $A \to a$ در G مربوط میکند . هرقاعده انتقال به شکل (q_S, ε, q_f) درδ به یک قاعده تولید به شکل $S \to \varepsilon$ درGمربوط میشود .

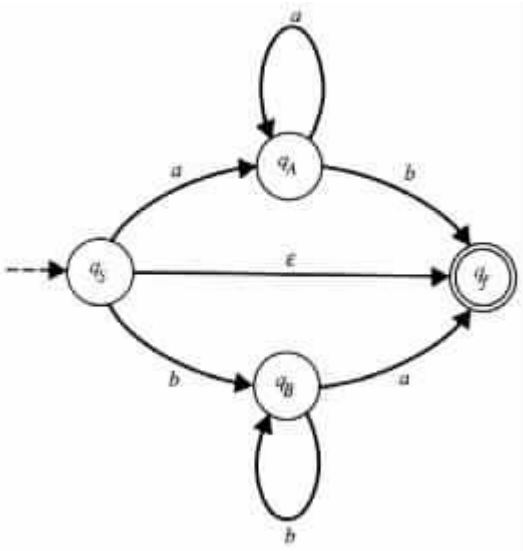
ماشین باحالات متناهی M برای دنبال کردن اشتقاقهایی ازگرامرG درمحاسباتش ساخته شده است . M حالاتش را برای حفظ توالیِ نمادهای غیرپایانیِ مورداستفاده در فرمهای جمله ایی ازG بکارمیبرد . M قواعد انتقالش را برای پیمایش کردن نمادهای ورودیی استفاده میکند کهG دراشتقاقهای مستقیمی که قواعدتولید متناظری را بکار میبرند تولید میکند .

ازاستقراء برروی n ، ماشین با حالت متناهی M که یک دنباله $q_{A0}x \vdash u_1q_{A1}v_1 \vdash u_2q_{A2}v_2 \vdash \ldots \vdash u_{n-1}q_{An-1}v_{n-1} \vdash xq_{An}$ از n حرکت دارد ساخته میشود اگروتنهااگر گرامر G یک اشتقاق باطول n به شکل $A_0 \Rightarrow u_1A_1 \Rightarrow u_2A_2 \Rightarrow \ldots \Rightarrow u_{n-1}A_{n-1} \Rightarrow x$ داشته باشد . چنین تناظری به خصوص برای $A_0 = S$ برقرار است . در نتیجه ، x در $L(M)$ است اگروتنهااگر در $L(G)$باشد .

مثال ۲-۳-۵ گرامر نوع ۳ $G = <\{S, A, B\}, \{a, b\}, P, S>$ راملاحظه کنید ، که درآن P مرکب است از قواعد انتقال زیر .

$$
\begin{aligned}
S &\rightarrow \epsilon \\
&\rightarrow aA \\
&\rightarrow bB \\
A &\rightarrow aA \\
&\rightarrow b \\
B &\rightarrow bB \\
&\rightarrow a
\end{aligned}
$$

نمودارگذاردرشکل ۲-۳-۷



شکل ۲-۳-۷ یک ماشین باحالات متناهی که L(G) رامیپذیرد که درآن G گرامری ازمثال ۲-۳-۵ است .

یک ماشین باحالات متناهی که زبان L(G) را میپذیرد نشان میدهد . اشتقاق $S \Rightarrow aA \Rightarrow aaA \Rightarrow aab$ درG بوسیله محاسبه ی $q_S aab \vdash aq_A ab \vdash aaq_A b \vdash aabq_f$ ازM دنبال میشود.

نتیجه اینکه ماشین های باحالات متناهی وگرامرهای نوع ۳ کاملا شبیه سیستمهای ریاضی هستند . حالات درماشینها نقشی همانند نقش نمادهای غیرپایانی درگرامرها

ایفامیکنند و قواعد انتقال درماشینها نقشی مشابه با نقش قواعد تولید درگرامرها ایفا میکنند.

۲-۳ گرامرهای نوع ۳ و گرامرهای منظم

گرامرهای نوع ۳ کوچکتر از آن هستند که با قرار دادن محدودیت های معنی داربرروی آنها ،گرامرهایی نتیجه شود که نمیتوانند تمام زبانهای نوع ۳ راتولید کنند . به بیان دیگر ، برخی ازمحدودیت ها که برروی گرامرهای نوع ۳ گذاشته شده دسته ای اززبانهایی که آنها میتوانند تولیدکنند ،افزایش نمی یابند .

بویژه ، یک گرامر $G = <N,\Sigma, P, S>$ گرامر راست خطی گفته میشود اگرهریک ازقواعدتولیدش به هریک ازدوشکل $A \rightarrow xB$ یا $A \rightarrow x$ باشد ، که در آن A وB نشانه های غیرپایانی در N هستند و x یک رشته ازعلامات پایانی در Σ^* است .

یک گرامر ، گرامر چپ خطی گفته میشود اگرهریک ازقواعدتولیدش به هریک ازدو فرم $A \rightarrow Bx$ یا $A \rightarrow x$باشد ، که درآن A وB نشانه های غیرپایانی درN هستند و x یک رشته ازنشانه های پایانی در Σ^* است .

یک گرامر ، گرامر منظم گفته میشود اگر به صورت هریک ازگرامرهای راست خطی یا چپ خطی باشد .

ازتمرین ۲,۳,۵ یک زبان ، زبان نوع ۳ است اگروتنها اگر منظم باشد .

زبانهای منظم و عبارات منظم

زبانهای منظم همچنین میتوانند ازروی مجموعه تهی وازروی تعدادمتناهی ازمجموعه های منحصربفرد ، بوسیله عملیات اجتماع ، الحاق و Kleene closure تعریف شوند .

هرالفبای Σ را در نظر بگیرید . آنگاه یک مجموعه منظم روی Σ به روش ذیل تعریف میشود :

a . مجموعه تهی Ø ، (مجموعه {}ε فقط مشمول رشته تهی)، و مجموعه {a} برای هرعلامت a درΣ ، مجموعه های منظم هستند .

b . اگر L_1 و L_2 مجموعه های منظم باشند ، آنگاه اجتماع $L_1 \cup L_2$ ، الحاق $L_1 L_2$ و L_1 kleen clouser* نیز مجموعه های منظمند .

c . هیچ مجموعه دیگری منظم (باقاعده) نیست .

ازتمرین ۲,۳,٦ توصیف صفات زیرین برقراراست.

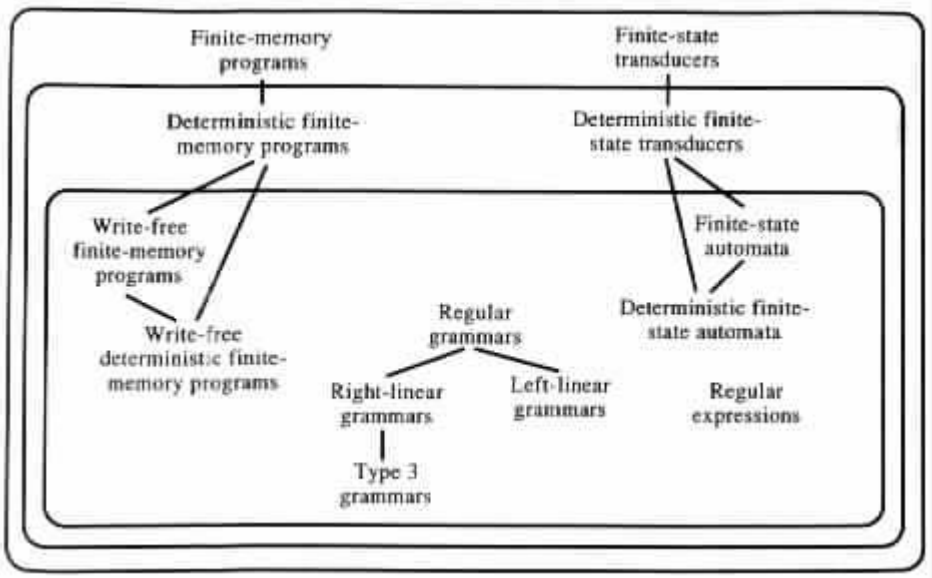
قضیه ۲-۳-٤ یک مجموعه ، یک مجموعه منظم است اگروتنها اگر بوسیله یک ماشین باحالات متناهی پذیرفته شود .

مجموعه های منظمی به شکل L_a* و $L_a L$, $L_a \cup L_b$, {a} , , {ε} Ø اکثرا بوسیله عبارات بترتیب * (a) و (a)(b) , (a)+(b) , a , ε , Ø تعریف میشوند . فرض شده a وb عباراتی باشند که دریک طریق مشابه بترتیب با L_a و L_b مشخص شوند . فرض شده a یک نماد ازالفباباشد . عباراتی که مجموعه های منظم دراین روش مشخص میکنند عبارات منظم نامیده میشوند .

اگر یک رابطه تقدم میان عملیاتهای Kleene closure ، الحاق واجتماع درترتیب معلوم فرض شود برخی پرانتزها میتوانند ازعبارات منظم حذف شوند . حذف پرانتزهادر عبارات منظم شبیه به حذف پرانتزها درعبارات حسابی است ، که درآن closure ، الحاق واجتماع درعبارات منظم نقشی مشابه به ، توان رسانی ، ضرب و جمع درعبارات حسابی ایفا میکنند .

عبارت منظم ٠*١٠*١١*(٠٠*١٠*١١*)* + ٠*(١*٠١*٠٠*(١١*٠١*٠٠*)* زبانی را مشخص میکند که بوسیله ماشین باحالات متناهیی شناخته میشود که نمودار انتقال آن درشکل ۲-۳-۲ داده شده است . عبارت نشان میدهد که هررشته با تعداد ا دلخواه از ٠ ها شروع میشود . سپس رشته با یک رشته در١*٠١*٠٠*(١١*٠١*٠٠*)* یا بایک رشته در١٠*١١*(٠٠*١٠*١١*)* ادامه میابد . درمورد اول ، رشته با یک تعداددلخواهی از یک ها ،به دنبال آن ٠ ، به دنبال آن تعدادی دلخواه از یک ها ، به دنبال آن یکی یا بیشتر ٠ ، به دنبال آن تعداد دلخواهی ازرشته هادر١١*٠١*٠٠* ادامه میابد .

برطبق بحث قبلی ، ماشین های با حالات متناهی غیرقطعی ، ماشین های باحالات متناهی قطعی ، گرامرهای منظم وعبارات منظم همه توصیف صفات اختصاصی زبان ها که برنامه هایی با حافظه متناهی میپذیرند، هستند. بعلاوه رویه های موثری برای حرکت میان توصیف صفات مختلف وجوددارند . این رویه ها پایه ای برای سیستم های بسیاری فراهم میکنند که برنامه های مبنی بر حافظه های متناهی رابواسطه توصیف صفاتی ازخاصیت قبلی تولید میکنند . برای نمونه ، یکی ازبهترین سیستم های شناخته شده ، LEX نامیده میشود ، ورودی هایی که عبارات منظم را عمومیت میدهند میگیرند و خروجی هایی که پوینده اندرا تولید میکنند . فایده چنین سیستم هایی بوضوح درتلاش تقلیل داده شده ای است که آنها برای بدست آوردنِ برنامه های خواسته شده (درخواستی) نیازدارند .شکل ۲-۳-۸



شکل ۲-۳-۸ ارتباطات ساختمانی و تابعی میان برخی سیستمهای توصیفی

نشان دادن سلسله مراتب ساختمانی و تابعی برای برخی سیستم های توصیفی . سلسله مراتب ساختمانی بوسیله گرافهای بدون دور جهتدار نشان داده شده اند . سلسله مراتب تابعی بوسیله نمودارون نشان داده شده اند .

۲-٤: محدودیت برنامه های حافظه متناهی

به طور مستقیم می توان گفت که محاسباتی هستند که برنامه های حافظه متناهی نمی توانند نتیجه دهند، به دلیل اینکه محدودیتهای روی مقداری از حافظه که برنامه ها استفاده می کنند تحمیل می شود. برای مثال می تواند گفته شود که $\{a^n\ b^n | n \geq o\}$ قابل تشخیص به وسیله ی هر برنامه حافظه متناهی نیست. استدلال در اینجا این است که بمجرد دسترسی به اولین b در داده ورودی، برنامه باید به خاطر بیاورد که چند تا a خوانده است علاوه بر این استدلال ادامه می دهد که هر برنامه حافظه متناهی مرز بالای روی تعدادی از مقادیر دارد در حالیکه چنین مرزی در تعداد a ها وجود ندارد که بتواند شامل ورودیها شود. به عنوان نتیجه، هر برنامه حافظه متناهی فقط یک تعداد از

رشته های مجموعه $\{a^n\ b^n | n \geq o\}$ را می شناسد.

اهداف این بخش نشان دادن محاسباتی هستند که نمی توانند بوسیله ی برنامه های حافظه متناهی نتیجه شوند و آماده کردن ابزار رسمی برای شناسایی چنین محاسباتی است. دلایل به طور مستقیم به انتزاع بحث بالا تکیه می کنند. بهر حال، قابل ذکر است که مساله تصمیم گیری برای هر زبان معین، که آیا زبان به وسیله ی یک برنامه حافظه متناهی قابل شناسایی هست، می تواند نشان داده شود که تصمیم ناپذیر است (یک زمان معین از لحاظ ذکر شده غیر قابل تصمیم گیری است). (نگاه کنید به قضیه ٤-٥-٦)

لم فشار برای زبانهای منظم

قضیه زیر شرایط لازم را برای اینکه یک زبان به وسیله ی برنامه حافظه متناهی قابل تصمیم گیری باشد، آماده می کند. دلیل قضیه بر ملاحظه اینکه برنامه های حافظه متناهی باید یک حالت را در ورودیهای طولانی تکرار کنند تکیه می کند و زیر محاسبات بین تکرارهای حالات می توانند فشرده شوند.

قضیه ۲-٤-۱ (قضیه فشار برای زبانهای منظم): هر زبان منظم L یک عدد m دارد بطوریکه در شرایط زیر صدق کند. اگر ω در L هست و $|\omega| \geq m$، پس ω می تواند به صورت xyz نوشته شود در جائیکه $xy^K z$ برای هر $K \geq o$ در L هست. علاوه بر این $|y| \rangle o$ و $|xy| \leq m$.

اثبات : هر زبان منظم L را در نظر بگیرید. اجازه دهید M یک ماشین حالت متناهی باشد که L را می پذیرد. بوسیله ی قضیه ی ۲-۳-۱ می تواند فرض شود که M قواعد تغییر حالت ندارد. بوسیله ی m تعدادی از حالات M نمایش داده می شود.

در ورودی $\omega = a_1,...,a_n$ از L، ماشین حالت متناهی M محاسبه به فرم زیر دارد:

$$p_0 a_1 \cdots a_n \vdash a_1 p_1 a_2 \cdots a_n \vdash \cdots \vdash a_1 \cdots a_i p_i a_{i+1} \cdots a_n \vdash \cdots \vdash$$
$$a_1 \cdots a_j p_j a_{j+1} \cdots a_n \vdash \cdots \vdash a_1 \cdots a_n p_n$$

محاسبه از میان بعضی از دنباله ی $P_o, P_1, ... P_n$ از $n+1$ حالت پیش می رود. وقتیکه P_o حالت اولیهM هست و P_n یک حالت پذیرفتنی از M هست. در هر حرکت از محاسبه درست یک نشانه ورودی موجود خوانده می شود. اگر طول n از ورودی مساوی کوچکترین تعداد m حالت از M هست ، پس محاسبه شامل m یا بیشتر حرکت است و بعضی از حالات q باید در مدت اولین m حرکت تکرار شده باشند. یعنی اگر $n \geq m$ پس $P_i = P_j$ برای برخی i و j بطوریکه $o \leq i \langle j \leq m$. در چنین موردی $x = a_1...a_i$ و $g = a_{i+1}...a_j$ و $z = a_{j+1}...a_n$

بگیرد.

با چنین تجزیه xyz از ω، محاسبه بالا از M فرم زیر را می گیرد:

$$p_0xyz \vdash^* xqyz \vdash^* xyqz \vdash^* xyzp_n$$

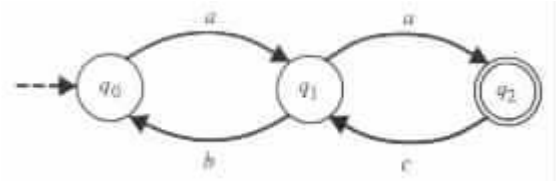
در طول محاسبه ، حالت $q = p_i = p_j$ از M تکرار شده است. رشته x قبل از رسیدن حالت q که تکرار شده است مصرف شده است. رشته y در طول تکرار حالت q مصرف شده است. رشته z بعد از تکرار حالت q مصرف شده است.

بنابراین M یک محاسبه پذیرفتنی به فرم زیر دارد:

$$p_0xy^kz \vdash^* xqy^kz \vdash^* xyqy^{k-1}z \vdash^* \cdots \vdash^* xy^kqz \vdash^* xy^kzp_n$$

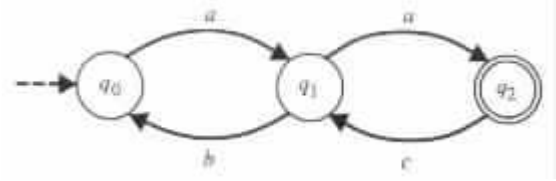
برای هر $K \geq o$. یعنی M یک محاسبه پذیرفتنی روی xy^Kz برای هر $K \geq o$ دارد در جائیکه M شروع می شود و پایان می یابد با مصرف هر y در حالت q .

زیر رشته y که بین تکرار حالت q مصرف شده است تهی نیست، زیرا به وسیله ی فرض M قاعده تغییر حالت ندارد.



شکل ۲-٤-۱ اتوماتای حالت محدود

مثال ۲-٤-۱ فرض کنید L یک زبان منظم باشد که توسط ماشین حالت متناهی شکل زیر پذیرفته شده است.



استفاده کردن از مجموعه اصطلاحات در قضیه فشار (قضیه ۲-٤-۱) و L ثابت m=۳ دارد.

در ورودی $\omega = ababaa$ ماشین حالت متناهی از بین دنباله حالت q_2 و q_1 و q_0 و q_1 و q_1 و q_0 پیش می رود. برای چنین ورودی قضیه فشار $x = \varepsilon$ و y=ab و $z = abaa$ و تجزیه ی x=a و y=ba و z=baa را فراهم می کند. موعد اولین تجزیه در اولین تکرار از حالت q_0 است و دومین یک نتیجه از اولین تکرار حالت q_1 است.

برای هر رشته ی ω از یک طول مینیمم ۳، قضیه فشار یک تجزیه xyz ایجاب می کند ، در این هم رشته y باید هر یک از دوتای ab یا ba یا ac باشد. اگر y=ab پس $x = \varepsilon$ و تکرار از q_0 فرض

می شود. اگرy=ba پس x=a و تکرار از q فرض می شود هست، اگر y=ac پس x=a و فرض می شود تکرار از q_1 هست.

کاربردهای لم فشار:

برای اثبات اینکه یک زبان معین L منظم نیست ، کاربردلم فشار با روش زیر تناقض را نتیجه می دهد:

(a) برای هدف اثبات، فرض می شود که L یک زبان منظم است.

(b) اجازه دهید M ثابت ضمنی برای L بوسیله ی لم فشار باشد، تحت این فرض در a، که L منظم است.

(c) یک رشته ω در L پیدا کنید که طول آن در کمترین مقدار، m است. لازم است که ω به یک K اشاره کند برای هر تجزیه xyz از ω، چنانکه xyz در L نیست. یعنی یک ω پیدا کنید که بوسیله ی کاربرد قضیه فشار برساند که یک رشته در L نبایست، در واقع در آنجا باشد.

(d) تناقض در (c) را برای نتیجه گرفتن اینکه قضیه فشار برای L بکار گرفته نمی شود استفاده کنید.

(e) از نتیجه در d که دلالت می کند که فرض در a، که L منظم هست و غلط است استفاده کنید.

این باید تأکید شود در روش ذکر شده ، قضیه فشار فقط وجود یک مقدار m برای زبان منظم فرض شده و وجود تجزیه xyz برای رشته ی انتخاب شده ی ωرا ایجاب می کند . این قضیه هیچ اطلاعی راجع به مقادیر m و y و x و z فراهم نمی کند و علاوه بر این محدودیتی را که آنها ارضا می شوند شرایط $|g|\rangle o$ و $|xy| \le m$ است.

اهمیت برای شما به سبب شرط $|xy| \le m$ تکذیب می شود در پذیرش برخی محدودیت روی تجزیه ی ممکن که برای ω انتخاب شده مطرح هستند. اهمیت محدودیت $|y|\rangle o$ در توانایی یک تغییرمناسب در رشته های فشرده شده است.

مثال ۲-٤-۲ : به زبان نامنظم L $L=\{ 0^n1^n \mid n>=0\}$ را در نظر بگیرید .برای اثبات اینکه L نامنظم است برعکس آن یعنی اینکه منظم است فرض می شود. بااین فرض که L منظم است وجود یک m که شرایط قضیه فشار را برای L برآورده کند، نتیجه گرفته می شود. رشته $w=0^m1^m$ را L انتخاب کنید. بوسیله ی قضیه فشار $w=0^m1^m$ یک تجزیه از Z , y , x دارد، در جائیکه $|xy| \le m$ و $|y|\rangle o$ و $xy^K z$ برای هر $K \ge o$ در L هست. یعنی تجزیه باید از فرم $y = o^j$ و $x = o^i$ و $z = o^{m-i-j}$ برای چندین i و j باشد که $j\rangle o$. (توجه: مقادیر i و j و m نمی توانند اختیاری انتخاب شده باشند) علاوه بر این باید $xy^o z$ در L باشد. هر چند $xy^o z = o^{m-j-m}$ نمی تواند در L باشد زیرا $j\rangle o$ است. این فهمیده می شود که قضیه فشار برای L بکار گرفته نمی شود. بنابراین تناقض فرض اینکه L نامنظم است.

انتخابهای دیگر از ω می توانند بکار گرفته شوند که نشان دهند L نامنظم است. بنابراین آنها ممکن است در چندین مجموعه تحلیل نتیجه گرفته شوند. برای مثال، برای $w = 0^{m-1}1^{m-1}$ سه شکل ممکن از تجزیه را فراهم می کند:

در هر مورد هر یک از سه فرم تجزیه باید نامناسب بودن را نشان دهند. نتیجه اینکه قضیه فشار برای

ω بکار

گرفته نمی شود برای (a) انتخاب k=o که $xy^oz = o^{m-i-j}\Big|^{m-1}$ که در L نیست. برای (b) انتخاب k=2 که در $xy^2z = o^{m-1}\Big|o^j\Big|^{m-1}$ نیست برای (c) انتخاب c=o $xy^oz = o^{m-1}\Big|^{m-2}$ که در L نیست.

مثال ۲-٤-۳: زبان نامنظم $L=\{\alpha\alpha^{rev} \mid \alpha \text{ is in } \{a,b\}^*\}$ را در نظر بگیرید. برای اثبات اینکه L منظم نیست و خلاف آن یعنی که منظم است فرض می شود. سپس وجود یک m ثابت پایدار که شرایط قضیه فشار از L برآورده کند استنباط می کنیم.

$\omega = a^m bba^m$ را در L انتخاب کنید. بوسیله ی قضیه ی فشار برای $a^m bba^m = xyz$ برخی x , y , z چنانکه $|y| > o$ و $|xy| \le m$, برای هر xyz در L هست. یعنی $x = a^i$ و $y = a^j$ و $z = a^{m-i-j}bba^m$ برای برخی i و j چنان که $j \rangle o$. بعلاوه $xyz = a^{m-j}bba^m$ در L نیست. بنابراین تناقض فرض یعنی L منظم است.

این باید مورد ملاحظه باشد که هر انتخاب ω خواستن تناقض را ایجاب نمی کند. برای مثال به انتخاب a^{2m} برای ω توجه کنید. بوسیله ی قضیه فشار، a^{2m} یک تجزیه xyz در $x = a^i$ و $y = a^j$ و $z = a^{2m-i-j}$ برای برخی i و j بطوریکه $j \rangle o$ دارد. با چنین تجزیه $xy^Kz = a^{2m=(k-1)j}$ در L نیست اگر و تنها اگر $2m + (k-1)j$ یک صحیح فرد است. از طرف دیگر $2m + (k-1)j$ یک فرد صحیح است اگر و تنها اگر k یک عدد زوج است و j یک عدد فرد است. اگر چه k می تواند اختیاری انتخاب شده باشد و برابر هر مقداری باشد اما در مورد j چنین نیست. بنابراین انتخاب a^{2m} برای ω خواستن تناقض را ضمانت نمی کند.

تعمیم برای لم فشار

اثبات لم فشار مبنی بر ملاحظه اینکه یک حالت در هر محاسبه از یک ورودی طولانی پی در پی است ، با یک بخش از ورودی مصرف شده موجود در بین تکرار است . تکرار حالت ، فشار زیر مجموعه ها را بین تکرار بدست آمده از محاسبات پذیرفتنی جدید در ورودیهای متفاوت مجاز می داند . اثبات لم فشار با کمی اصلاح برای قضیه زیر نیز استفاده می شود .

قضیه ۲-٤-۲ : برای هر رابطه R که بوسیله ی مبدل حالت متناهی قابل محاسبه است ، یک ثابت m وجود دارد که در شرایط زیر صدق کند . اگر (v,w) در R است و $|v| + |w| \ge m|$ پس v می تواندبه عنوان $x_vy_vz_v$نوشته شود و w می تواند به عنوان $x_wy_wz_w$نوشته شود،در جاییکه $(x_vy_v^kz_v, x_wy_w^kz_w)$ هست برای هر K>=0 و علاوه بر این $|x_vy_v| + |x_wy_w| \le m|$ و $|y_v| + |y_w| > 0$.

یک طرح همانند این ،که می تواند لم فشار را برای زبانهای نامنظم قطعی بکار ببرد، می تواند از قضیه ۲-٤-۲ برای رابط قطعی که به وسیله ی مبدلهای حالت متناهی قابل محاسبه نیستند استفاده کند .

مثال ۲-٤-٤ : رابطه $R = \{(u, u^{rev}) \mid u \, is \, in \, \{o,1\}^*\}$ به وسیله ی یک مبدل حالت متناهی قابل محاسبه نیست اگر R بوسیله ی یک مبدل حالت متناهی قابل محاسبه باشد در جائیکه m ثابتی وجود داشته باشد که شرایط قضیه ۲-٤-۲ را برای R برآورده کند . در این مورد ، چون ($^m0^m$۱ و $^m1^m$۰)در R هست ، پس $u=0^m1^m$ می تواند به عنوان $x_vy_vz_v$ و $u^{rev} = 1^m0^m$ می تواند بعنوان $x_wy_wz_w$ نوشته شود در جائیکه $x=0^iv$ و $y=0^jv$ و $z = 0^{m-i-} v1^m$.

۲-۵ ویژگی های بستار(بستگی) برای برنامه های با حافظه متناهی

یک خط مشی مفید درساده کردن (مختصرکردن) کاربرنامه نویسی تقسیم کردن مسئله ی داده شده به زیرمسائل است ، طراحی کردن زیربرنامه هایی که زیرمسائل راحل میکنندوسپس ترکیب کردن زیربرنامه ها در یک برنامه که مسئله اصلی را حل میکند . درطراحیِ مبدل های باحالات متناهی (وبرنامه های باحافظه متناهی) راهی مشابه پذیرفته میشود، تعیین کردن آن اعمالی که مجموعه ای ازروابطی رانگه میدارندکه بوسیله مبدلهای باحالات متناهی محاسبه پذیرند مفیداست . چنین بینشی (اطلاعی)میتواند درهنگام تصمیم گرفتن درباره اینکه مسائل داده شده چگونه به زیرمسائل ساده تر تجزیه میشوندمفید باشد ، بعلاوه درآماده کردن مسائل برای بصورت خودکاردرآوردنِ ترکیبِ زیربرنامه ها دربرنامه هامفیدهستند.

در کل، یک مجموعه تحت یک عمل خاص بسته گفته میشوداگربکارگیری آن عمل برروی عناصرِمجموعه یک عنصرازهمان مجموعه را نتیجه دهد .

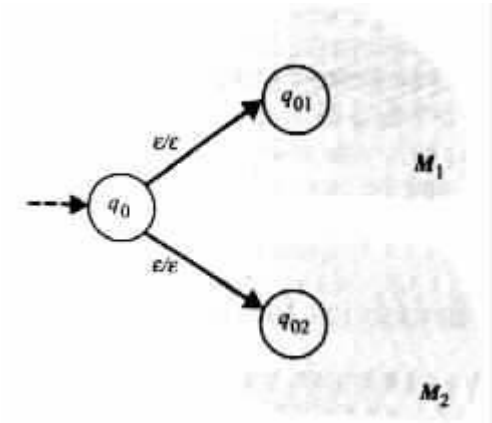
مثال ۲-۵-۱ مجموعه اعدادطبیعی روی جمع بسته است ، اما روی تفریق بسته نیست . مجموعه اعدا صحیح روی جمع و تفریق بسته است ، اماروی تقسیم بسته نیست . مجموعه $\{S$ یک مجموعه ی ۵تایی یا بیشتر ازاعداد صحیح است $\{S \mid$ تحت اشتراک یا مکمل بسته نیست . $\{S$ مجموعه ای ازبیشتراز۵عددصحیح است $\{S \mid$ تحت اشتراک بسته است اما تحت اجتماع ومکمل بسته نیست .

اولین قضیه دراین بخش به بستگی تحت عمل اجتماع مربوط میشود .

قضیه ۲-۵-۱ دسته ای ازروابطِ محاسبه پذیر بوسیله مبدلهای باحالات متناهی تحت اجتماع بسته اند .

اثبات دومبدل $M_1 = <Q_1, \Sigma_1, \Delta_1, \delta_1, q_{01}, F_1>$ و $M_2 = <Q_2, \Sigma_2, \Delta_2, \delta_2, q_{02}, F_2$ راملاحظه کنید . بدون هیچ لطمه ای به اصل کلی فرض میشود که مجموعه های q_1 وq_2 ازحالات ،متقابلا مجزا و هیچیک ازآنها شامل q_0 نمیشوند .

M3را مبدل با حالات متناهی $<Q_3\Sigma_3,\Delta_3,\delta_3, q_0, F_3>$ میگیریم ، که درآن $Q_3 = Q_1 \cup Q_2 \cup \{q_0\}$ ، $\Sigma_3=\Sigma_1\cup\Sigma_2$ ، $\delta_3=\delta_1\cup\delta_2\cup\{(q_0, \varepsilon, q_{01}), (q_0, \varepsilon, q_{02})\}$ و $F_3 = F_1 \cup F_2$ (شکل۲-۵-۱ را ببینید.)



شکل ۲-۵-۱ یک مدل از یک مبدل با حالات متناهی M_3 که $R(M_1) \cup R(M_2)$ را محاسبه میکند .ذاتا M_3 یک مبدل باحالات متناهی است که درآغازهرمحاسبه بطورغیرقطعی انتخاب میکندکه هریک ازمحاسبات M_1 یاM_2 رادنبال کند .بموجب ترکیب $R(M3) = R(M1) \cup R(M2)$

ویژگیهای بستگی درکنارفوایدشان درساده کردن کاربرنامه نویسی ، میتوانددرشناخت روابطی که نمیتوانند بوسیله مبدلهای باحالات متناهی محاسبه شوند نیز بکارروند .

مثال ۲-۵-۲ اجتماع زبانهای $L_1 = \{\varepsilon\}$ و $L_2\{=0^i1^i \mid i\geq 1\}$ برابراست بازبان $L_3 =\{ 0^i1^i \mid i\geq 0\}$. بنابرقضیه ۲-۵-۱ اجتماع $L_3=L_1 \cup L_2$ از L_1 وL_2 یک زبان منظم است .اگر L_1 و L_2 زبانهایی منظم باشند . ازآنجاییکه $L_1=\{\varepsilon\}$ یک زبان منظم است ، استنباط میشود که L_3 یک زبان منظم است اگر L_2 یک زبان منظم باشد . بهرحال ،مطابق با مثال ۲-٤-۲ زبان $L_3 =\{0^i1^i \mid i\geq 0\}$ منظم نیست. در نتیجه $L_2=\{0^i1^i \mid i\geq 1\}$ نیز منظم نیست .

روابط $R_1 =\{(0^i1^j, c^i) \mid i, j >= 1)\}$ و $R_2 = \{ (0^i1^j, c^j) \mid i ,j\geq 1\}$ بوسیله مبدلهای باحالات متناهیِ قطعی محاسبه پذیرند . زوج $(0^i1^j, c^k)$ در R_1 است اگروتنهااگر $k=i$ و آن زوج درR_2 است اگروتنهااگر $k=j$. اشتراک $R_1 \cap R_2$ شامل همه زوجهای $(0^i1^j, c^k)$ که در $k=i=j$ صدق میکند میشوند ،بعبارت دیگر،$R_1 \cap R_2$ رابطه ی $\{ (0^n1^n, c^n) \mid n \geq 1 \}$ است.

اگر $R_1 \cap R_2$ بوسیله یک مبدل باحالات متناهی محاسبه پذیرباشدآنگاه زبان $\{ 0^n1^n \mid n \geq 1 \}$ باید منظم باشد .هرچند بنابرمثال ۲,٤,۲ زبان منظم نیست . بنابراین دسته ای ازروابط که بوسیله مبدلهای باحالات متناهی محاسبه پذیرند تحت اشتراک بسته نیستند.

دسته ای از روابط که بوسیله مبدلهای باحالات متناهی محاسبه پذیرند تحت مکمل نیز بسته نیستند . یک فرض خلاف این مطلب را میرساند که زبان نامنظم $R_1 \cap R_2$ منظم است ، زیرا توسط قانون دمورگان $R_1 \cap R_2=\overline{(\overline{R_1} \cup \overline{R_2})}$. بعبارت دیگر ، یک بستگیِ انگاشته شده تحت مکمل اشاره خواهدداشت برآنکه $\overline{R_1}$ و$\overline{R_2}$ توسط مبدلهای باحالات متناهی محاسبه پذیرند . قضیه ۲-۵-۱ اشاره به این خواهد داشت که اجتماع $\overline{R_1} \cup \overline{R_2}$ بوسیله مبدلهای باحالات متناهی محاسبه پذیرند. وسرانجام ، کاربرد دیگرِ فرض این مطلب راخواهدرساند که $\overline{(\overline{R_1} \cup \overline{R_2})}= \overline{R_2} \cap \overline{R_1}$ نیز بوسیله یک مبدل

باحالات متناهی محاسبه پذیراست .

انتخابی از R_1 و R_2 نیزبرعدم بستگی ،تحت اشتراک ، بسبب دسته ای ازروابطِ محاسبه پذیر بوسیله مبدلهای باحالات متناهیِ قطعی اشاره دارد. عدم بستگی تحت اجتماع و مکمل ، بسبب (برروی) دسته ای ازروابطِ محاسبه پذیر بوسیله مبدلهای باحالات متناهیِ قطعی ،با انتخابی ازروابط {(۱, ۱)} و {(۱, ۱۱)} مقدراست .

برای زبانهای منظم قضیه ی زیر برقرار است .

قضیه ۲-۵-۲ زبانهای منظم تحت اجتماع ، اشتراک و مکمل بسته اند .

اثبات برطبقِ قانون دمورگان و بستگیِ زبانهای منظم تحت اجتماع (قضیه ۲-۵-۱ را ببینید) ،این کافیست که نشان دهیم زبانهای منظم تحت مکمل بسته اند .

برای رساندن منظوری ازاین اثبات هرماشین باحالات متناهی به شکل $M = <Q, \Sigma, \delta, q_0, F>$ رادرنظربگیرید. برطبق قضیه ۲-۳-۱ میتواند این فرض شود که M غیرقطعی است وشامل هیچ قواعدگذاری نمیشود .

M_{eof} را، M با یک مقدارتازه اضافه شده میگیریم، حالت عدم پذیرشِ "تله" ، q_{trap} وقاعده های انتقال تازه اضافه شده ی زیر را بیان میکند .

a). (q, a, q_{trap}) برای هرزوج (q, a)—ازیک حالت q درQ ویک نمادورودی a درΣ -- که هیچ حرکت تعریف شده در M ندارد . بعبارت دیگر ، برای هر (q, a) هیچ P یی درQ چنانکه (q, a, p) در δ باشد وجودندارد.

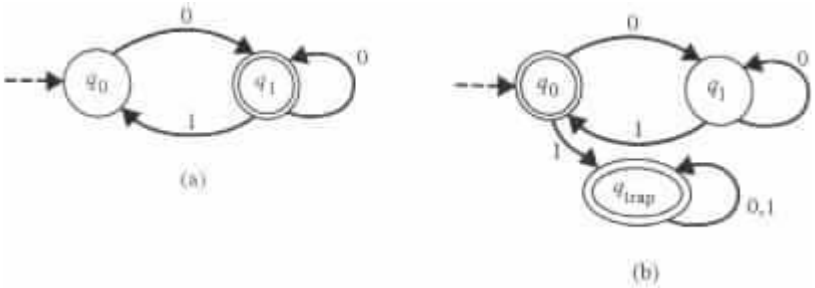
b). (q_{trap}, a, q_{trap}) برای هرنمادورودی a درΣ .

به تعبیری M_{eof} یک ماشین باحالات متناهی معادل با M است . بعلاوه M_{eof} همه ورودی ها را تا خاتمه شان دنبال میکندوهیچ قواعد انتقال ε ی ندارد .

مکملِ زبان $L(M)$ بوسیله یک ماشین باحالات متناهی $M_{complement}$ که از M_{eof} بوسیله تبادلِ نقش هایی از حالات پذیرفته شده و پذیرفته نشده بدست می آید ، پذیرفته میشود.

برای هر ورودیِ مفروض $a_1 - a_n$ ماشین باحالات متناهی $M_{complement}$ یک مسیر منحصربفرد دارد که $a_1 - a_n$ را تامنتهاالیه ش دنبال میکند . مسیر مطابق است با دنباله ای ازحرکات که M_{eof} روی چنین ورودیی طی میکند .بنابراین ، $M_{complement}$به یک حالت پذیرفته شده روی یک ورودیِ مفروض میرسد اگروتنها اگر M_{eof} به یک حالت پذیرفته شده روی ورودی نرسد .

مثال ۲-۵-۳ M را یک ماشین با حالات متناهی میگیریم که نمودار انتقال آن درشکل ۲-۵-۲ (a) داده شده است .



شکل ۲-۵-۲ ماشین باحالات متناهی در (b) میپذیرد مکمل زبانی راکه ماشین باحالات متناهی در(a) میپذیرد .

مکمل $L(M)$ بوسیله ماشین با حالات متناهی که نمودار انتقال آن درشکل ۲-۵-۲ (b) داده شده است پذیرفته میشود .بدون حالت تله q_{trap} هیچیک از M یا $M_{complement}$ توانایی پذیرفتن ورودی ۰۱۱ را نخواهندداشت ،زیرا هیچ یک ازآنها نخوهندتوانست تمام ورودی را دنبال کنند.

بدون الزام به اینکه الگوریتم فقط روی ماشین باحالات متناهی قطعی کاربردی داشته باشد $M_{complement}$میتواندمنجرشود به پذیرفتن یک ورودی که M نمیپذیرد . برای مثال بااضافه شدن قاعده انتقال $(q_1, 1, q_1)$ به M و $M_{complement}$، روی ورودی ۰۱ هریک از ماشین های باحالات متناهی میتوانند درهر یک ازحالات q_0 یا q_1 خاتمه یابند . درچنین مواردی ، M،01 را خواهد پذیرفت زیرا آن میتواند به حالت q_1 برسد و $M_{complement}$،01 را خواهدپذیرفت زیرا ان میتواند به حالت q_0 برسد .

خواص تصمیم پذیری برای برنامه های حافظه متناهی :

• مسئله تهی بودن، مسئله هم ارزی ، مسئله توقف و دیگر مسائل تصمیم گیری برای برنامه های حافظه متناهی یا به طور معادل برای مبدلهای حالت متناهی ، در یک روش یکسان برای انواع کلی از برنامه‌ها مشخص هستند .

• برای مثال مساله برابری برای مبدلهای حالت متناهی ، برای هر جفت داده از مبدلهای حافظه متناهی می پرسد که آیا مبدلها برای هر جفت داده یک رابطه را حساب می کنند یا نه .

• متشابهاً ، مساله توقف برای مبدلهای حالت متناهی برای هر جفت داده (x,M) از مبدل حالت متناهی M و از یک ورودی X برای M می پرسد که آیا M فقط محاسبات توقف در x دارد .

• در این بخش بعضی خواص مبدلهای حافظه متناهی که قابل تصمیم گیری باشند ، نشان داده شده هستند . اثبات ها در ماهیت سودمند هستند و بنابراین الگوریتم های موثر برای تعیین خواص در بحث ایجاب می کنند اولین قضیه برای کاربردهایش جالب است . این (قضیه)مربوط است به مساله تصمیم گیری در مورد اینکه ایا یک اتوماتای حالت متناهیداده شده هیچ ورودی را نمپذیرد.

قضیه ۲-۶-۱ : مساله تهی بودن برای اتوماتای حالت متناهی قابل تصمیم گیری است .

• اثبات : اتوماتای حالت متناهی M را در نظر بگیرید . M برخی از ورودی ها را می پذیرد اگر و تنها اگر یک راه در نمودار انتقال وجود دارد که این راه از گره حالت آغازی به یک گره از حالت پذیرش است موجود باشد. وجود چنین راهی می تواند به وسیله ی الگوریتم زیر مشخص شود :

• گام اول : در نمودار انتقال ، گره ای حالت آغازی از M علامت زده شود .

• گام دوم : مکرراً گره های علامت نخورده در نمودار انتقال که به وسیله ی یک یال از گره علامت خورده قابل دسترسی هستند علامت زده شوند.. وقتیکه گره های اضافی نتوانند علامت زده شوند فرآیند پایان یابد.

•گام سوم : اگر نمودار تجزیه شامل یک گره علامت خورده است که یک حالت پذیرش است پس تعیین می شود که L(M) تهی نیست . در غیر اینصورت ، تعیین می شود که که L(M) تهی است .

به وسیله تعریف،برنامه فقط ۱ محاسبه توقف روی ورودیهایی دارد که پذیرفته شده اند.

• یک تصمیم گیری کلی مهم درباره برنامه ها این است که آیا آنها روی همه ی ورودی های متوقف می شوند . اثبات قضیه زیر نشان می دهد ، چطور در مورد برنامه های حافظه متناهی ، مساله توقف یکنواخت می تواند به مساله تهی بودن تبدیل شود .

قضیه ۲-۶-۲ : مساله توقف برای اتوماتای حالت متناهی قابل تصمیم گیری است .

• اثبات : اتوماتای حالت متناهی $M = \langle Q, \Sigma, \delta, q_0, F \rangle$ را در نظر بگیرید . بدون از دست دادن کلیت ، فرض می شود که نماد C در Σ نیست و این که Q، n حالت دارد . در اضافه فرض می شود که هرحالت بواسطه این که M می تواند به یک محاسبه پذیرش بوسیله نخواندن هیچ ورودی هم برسد ، یک حالت پذیرش است . بگیرید A یک اتوماتای حالت متناهی باشد که بوسیله جایگزینی هر قاعده انتقال از فرم(q, ε, p) با یک قاعده انتقال از فرم (q,c,p) از M بدست می آید . بگیرید B یک اتوماتای حالت متناهی باشد که زبان C^n} یک زیر از x است $\{x | x \, is \, in (\Sigma \cup \{c\})^* \, and$ را می پذیرد .

M یک محاسبه توقف در یک ورودی معین دارد اگر و تنها اگر دو شرط زیر برقرار باشد .

(a) ورودی بوسیله M پذیرفته نشده است .

(b) در یک ورودی معین M می تواند به یک حالت برسد که می تواند بدون خواندن هیچ نماد وردی تکرار شده باشد .

بنابراین M یک محاسبه بی توقف دارد اگر و تنها اگر A برخی ورودی را بپذیرد که C^nرا بعنوان زیر رشته بپذیرد .

بوسیله اثبات قضیه ۲-۵-۲ یک اتوماتای حالت متناهی C می تواند پذیرفتن مکمل L(A) را تفسیر کند . به وسیله همین اثبات ، یک اتوماتای حالت متناهی D می تواند پذیرش انتقال از L(B) و L(C) را تفسیر کند .

بوسیله ی تعریف ، D اتوماتای حالت متناهی است که درست همان ورودی هایی را میپذیرد که C^n را بعنوان زیر رشته دارند و بوسیله ی A پذیرفته شده نیستند . یعنی D هیچ ورودی نمی‌پذیرد اگر و تنها اگر M روی همه ورودی ها متوقف شود . قضیه،قضیه۲-۶-۱ را پیروی میکند.

برای برنامه های حافظه متناهی که نیاز دارند روی همه ورودیها خاتمه نیابند ، نتیجه اثبات زیر یک الگوریتم تصمیم گرفتن ایجاب میکند که تعیین کند آیا آنها روی ورودیهای معین متوقف می شوند یا

نه .

قضیه ۲-٦-۳ : مساله توقف برای ماشین حالت متناهی قابل تصمیم گیری است .

اثبات : هر ماشین حالت متناهی M و هر ورودی $a_1 \ldots\ldots a_n$ برای M را بررسی کنید . در اثبات قضیه ۲-۳-۱ یکی می تواند برای هر $i=1, \ldots\ldots,n$ مجموعه $Aa_1\text{-}a_i$ از همه ی حالات را که می تواند بوسیله ی مصرف $a_1 \ldots\ldots a_i$ بدست بیاید ، نتیجه بگیرد . پس M مصمم است که توقف کند اگر و تنها اگر دو شرط زیر برقرار باشد :

a) $Aa_1 \ldots a_n$ شامل یک حرکت پذیرش باشد .

b) برای هیچ عدد صحیح i که $1 \leq i \leq n$ و مجموعه $Aa_i \ldots a_i$ شامل یک حالت که می تواند از خودش بوسیله ی یک یا بیشتر حرکت با قواعد تجزیه بدست آید ، نباشد .

• در اینجا جزئیات بسیار دیگری هستند که برای برنامه های حافظه متناهی قابل تصمیم گیری هستند . این بخش یا قضیه زیر پایان می یابد .

• قضیه۲-٦-٤ : مساله هم ارزی برای اتوماتای حالت متناهی قابل تصمیم گیری است .

• اثبات : دو اتوماتای حالت متناهی M_1 , M_2 برابرند اگر و تنها اگر (($L(M_1) \cap \overline{L(M_2)}$) ∪ ($L(M_2) \cap \overline{L(M_1)}$)) =□ برقرار باشد ، جائیکه($L(M_i)$ متمم گیری از $L(M_i)$ برای $i = 1,2$ است .

نتیجه می تواند از اثبات قضیه ی ۲-٦-۱ و قضیه ی ۲-۵-۲ استفاده کند .

• نتیجه ی قضیه ی ۲-٦-٤ نیز می تواند برای مبدلهای حالت متناهی قطعی نیز استفاده شود. بهرحال برای نوع کلی مبدلهای حالت متناهی مساله هم ارزی غیر قابل تصمیم گیری است

فصل سوم

برنامه های بازگشتی با دامنه متناهی

اهداف

در پایان فصل، دانشجو با مفاهیم زیر آشنا می‌شود:

بازگشت

مبدل پشته

زبانهای مستقل از متن

محدودیت برنامه های بازگشتی با دامنه متناهی

خاصیت های بستاری برای برنامه های بازگشتی با دامنه متناهی

خاصیت تصمیم پذیری برای برنامه های بازگشتی با دامنه متناهی

مقدمه

بازگشت یکی از ابزار مهم برنامه نویسی است که به جاست در جای خود مورد بررسی قراربگیرد. در هر صورت اینجا با فراهم کردن یک کلاس برنامه ی میانه بین کلاس محدود شده ی برنامه های با حافظه متناهی و کلاس اصلی برنامه ها اهمیت بیشتری نیز پیدا میکند، این کلاس میانه با معرفی کردن بازگشت در محدوده ی برنامه های با دامنه ی متناهی بدست می آید.

بخش اول فصل نظریه ی بازگشت در برنامه ها را مورد رسیدگی قرار میدهد. بخش دوم نشان میدهد که برنامه های بادامنه متناهی با مبدل حالت متناهی که با حافظه پشته توسعه یافته است کاراکتر بندی میشوند. توصیف دستوری برنامه های بازگشتی با دامنه

متناهی در بخش سوم گرد آمده است. بخش چهارم محدودیتهای برنامه های بازگشتی با دامنه متناهی را بررسی میکند و بخش های پنجم و ششم خواص بستار بازگشت و تصمیم پذیری برنامه های بازگشتی بادامنه متناهی را به ترتیب مورد توجه قرار می دهد.

۳-۱ بازگشت

برنامه نویسی در خیلی از موارد زمانی که از بازگشت استفاده میکند آسان تر میشود. اگر چه بازگشت توابعی را که برنامه می تواند محاسبه کند افزایش نمی دهد ، اما در موارد خاصی از برنامه های با دامنه متناهی چنین افزایشی نیز انجام میشود.

بازگشت در برنامه ها به صورت زیر معرفی می شود:

الف) شبه دستورات در نمونه زیر آمده است . این دستورات برای تعریف کردن رویه ها استفاده می شود. لیست پارامترهای اصلی شامل نام متغیرهای مجزاست و بدنه هر رویه شامل ردیفی از دستورات اختیاری است.

```
Procedure <procedure name> (<list of formal parameters>)
  <Procedure body>
End
```

ب) فراخوانی دستورات به صورت زیر است. این دستورات برای فعال کردن اجرای رویه ها استفاده می شود . لیست پارامترهای واقعی از نظر اندازه با لیست پارامترهای رسمی متناظر با آن برابر است و شامل نام متغیرهای مجزاست.

```
Call <procedure name> (<list of actual parameters>)
```

ج) دستورات بازگشت در نمونه زیر آمده است . این دستورات برای غیرفعال کردن اجرای رویه ها استفاده میشود که تنها در بدنه برنامه باید ظاهر شوند.

```
Return
```

برنامه ی با دامنه متناهی که در آنها بازگشت استفاده می شود برنامه های بازگشتی با دامنه ی متناهی نامیده می شوند.

اجرای یک دستور فراخوانی اجرای رویه ی فراخوانده شده را فعال می کند. فعال کردن شامل کپی کردن مقدار متغیرها در لیست پارامترهای واقعی متناظر با متغیرهای لیست پارامترهای رسمی است و کنترل را به اولین دستور بدنه ی رویه منتقل می کند.

اجرای دستور بازگشت موجب غیرفعال شدن آخرین رویه های فعال تحت تأثیر می شود. غیرفعال کردن باعث انتقال کنترل به دستور فراخوانی می شود که مسئول آخرین فعال سازی است. با انتقال کنترل، مقدار متغیرهای لیست پارامترهای رسمی در متغیرهای متناظر در لیست پارامترهای واقعی کپی میشود. به علاوه متغیرهایی که در لیست پارامترهای واقعی ظاهر نمی شوند درست قبل از انجام شدن دستور فراخوانی مقادیرشان را برمی گردانند.

فرض می شود همه متغیرهای یک برنامه در تمام حوزه ی برنامه شناخته شده است و هرکدام از آنها اجازه دارند در یک تعداد دلخواه از لیست های متغیرهای رسمی و واقعی ظاهر شوند.

هر نوع تلاش برای وارد شدن به یا خارج شدن از یک رویه بدون استفاده از دستور فراخوانی یا دستور بازگشت باعث میشود که برنامه یک اجرای بی نتیجه داشته باشد.

در ادامه هر دستور فراخوانی و هردستور بازگشت به عنوان یک قطعه دستور مورد بررسی قرار میگیرد.

مثال ۳-۱-۱. درشکل ۳-۱-۱ P یک برنامه بازگشتی با دامنه متناهی دیده می شود ، متغیرها با دامنه {۰و ۱ } و صفر به عنوان مقدار اولیه فرض شده اند ، برنامه P ورودی هایی را می پذیرد که تعداد صفر ها و یک ها برابر باشد. با چنین ورودی برنامه تعداد صفرها و یک ها را به عنوان خروجی درنظر میگیرد.

```
do                          /* I1 */
  if eof then accept        /* I2 */
  read x                    /* I3 */
  write x                   /* I4 */
```

```
   call RP(x)                    /* I5  */
until false                  /* I6  */
procedure RP(y)
   do                            /* I7  */
      read z                     /* I8  */
      if z y then             /* I9  */
         return                  /* I10 */
      call RP(z)                 /* I11 */
   until false                /* I12 */
end
```

شکل ۳-۱-۱. یک برنامه بازگشتی با دامنه متناهی

با ورودی ۰۰۱۱۱۰۰۱ برنامه با خواندن اولین ۰ در I_3 ، شروع می شود ، ۰ را در I_4 می نویسد و کنترل برنامه را به RP در I_5 منتقل می کند. با ورود شدن به RP، x=y=z=0 در RP برنامه از قطعه دستور در I_8 برای خواندن مقدار ۰ به عنوان دومین ورودی استفاده می کند، و سپس در I_{11} به صورت بازگشتی RP را فراخوانی میکند.

باشروع فعالیت RP اولین ۱ در ورودی خوانده و سپس دستور بازگشت اجرا میشود، اجرای اولین فعالیت RP در I_{12} با x=y=z=0 ادامه می یابد. رویه با خواندن دومین ۱ از ورودی در z دنبال می شود سپس به اجرای برنامه اصلی در I_6 با x=y=z=0 بر می گردد. برنامه اصلی ۱ را در x می خواند و مقدار را در خروجی چاپ کرده و RP را فرا میخواند.

به محض وارد شدن به RP، x=y = 1 و z=0 می باشد. رویه ۰ را می خواند و سپس کنترل را به برنامه اصلی باز میگرداند. برنامه اصلی آخرین ۰ از ورودی را در x خوانده

و مقدار را د رخروجی چاپ کرده و دوباره RP را را می خواند. RP آخرین مقدار ورودی را خوانده و کنترل را به برنامه اصلی باز میگرداند جایی که محاسبه در I_2 به پایان می رسد.

جدول شکل ۳-۱-۲ جریان اطلاعات را در زمان فعال و غیرفعال شدن RP نشان میدهد

.

Values of x y z	Comments	Values of x y z	Comments
0 0 0	Initial values	1 0 0	Call at I_5
0 0 0	Call at I_5	1 1 0	Continue at I_7
0 0 0	Continue at I_7	1 1 0	Return
0 0 0	Call at I_{11}	1 0 0	Continue at I_6
0 0 0	Continue at I_7	0 0 0	Call at I_5
0 0 1	Return	0 0 0	Continue at I_7
0 0 0	Continue at I_{12}	0 0 1	Return
0 0 1	Return	0 0 0	Continue at I_6
0 0 0	Continue at I_6		

شکل ۳-۱-۲. جریان اطلاعات در برنامه شکل ۳-۱-۱ با ورودی ۰۰۱۱۱۰۰۱

```
call RP(parity)
if parity = 0 then
   if eof then accept
reject
```

```
procedure RP(parity)
  do      /* Process the next symbol in w. */
    read x
    write x
    parity := 1 - parity
    call RP(parity)
  or              /* Leave w and go to w^rev. */
    return
  until true
 /* Process the next symbol in w^rev. */
  read y
 if y x then reject
  return
end
```

شکل ۳-۱-۳. یک برنامه بازگشتی با دامنه متناهی

تعریفی که اینجا برای بازگشت ارائه شده است استاندارد نیست اما می توان با تعریف استاندارد معادل در نظر گرفت. دلیل انتخاب تعریف غیر استاندارد این است که تصور حالات برنامه های بازگشتی را ساده می کند. قرارداد قابل شناسایی بودن متغیرهای یک برنامه در تمام حوزه آن، امکان تعریف حالات به طور یکسان را فراهم میکند. قرارداد -- به محض اجرای دستور بازگشت متغیرهایی که در لیست پارامترهای واقعی ظاهر نشده اند قبل از اجرای دستورات فراخوانی متناظر با آن مقادیر

، مقادیر خود را بر میگردانند – نشان می دهد که متغیرهای محلی در رویه یکسان به نظر می آیند.

مثال ۳–۱–۲. برنامه بازگشتی با دامنه متناهی در شکل ۳–۱–۳ رابطه $\{ w$ رشته ای با طول زوج در$^* \{۰, ۱\} \mid (ww^{rev},w)\}$ را محاسبه می کند. دامنه متغیرها با $\{۰, ۱\}$, و صفر به عنوان مقدار اولیه فرض شده است. باورودی ۰۰۱۱۱۱۰۰ برنامه با یک محاسبه خروجی ۰۰۱۱ را تولید می کند. برنامه برای خواندن ۰۰۱۱ پنج بار رویه RP را فراخوانی می کند. سپس خواندن ۰۰۱۱ تا پنج بازگشت دنبال می شود.

این نتیجه می دهد که شباهت دسترسی به یک چیز برای مطالعه برنامه های با حافظه متناهی استفاده می شود هم چنین می تواند برای مطالعه برنامه های بازگشتی با دامنه متناهی استفاده میشود. مهم ترین تفاوت بین این دو مورد در پیچیدگی روش استدلال است.

به علاوه در مورد برنامه های با حافظه متناهی باید تأکید سود برنامه های بازگشتی با دامنه متناهی فقط به عنوان وسیله ای برای بررسی کردن کلاس اصلی برنامه ها مهم نیستند بلکه به خاطر اهمیت خودشان است. به عنوان مثال در بسیاری از کامپایلرها ، تحلیل گر نحوی از پایه به صورت برنامه های بازگشتی با دامنه متناهی طراحی شده اند.(فعالیت اصلی تحلیلگر نحوی دسته بندی علامت های برنامه ی در حال کامپایل طبق برخی قانونهای گرامری است. چنین دسته هایی توانایی دریافت دستورالعمل برنامه را به کامپایلر می دهند و بنابراین کد اصلی را تولید می کنند.

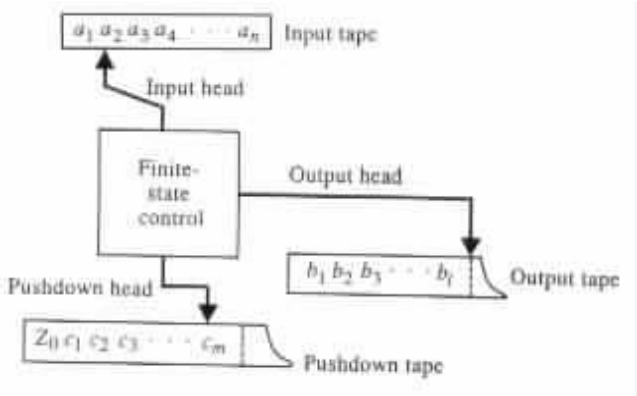
۳–۲ مبدل های پشته ای

عموما بازگشت در برنامه ها بوسیله پشته انجام می شود که یک حافظه(LIFO last in-first out) است پس طبیعی است پنداشته شود که بازگشت در برنامه هایی با دامنه متناهی می تواند به طور ضمنی از حافظه کمکی استفاده کند. به علاوه مشاهدات باعث شده است تعجبی نداشته باشد که محاسبات برنامه هایی با دامنه متناهی می تواند

با مبدل حالت متناهی که بایک پشته گسترش یافته است مشخص شوند چنین مبدل هایی ، مبدل های پشته نامیده می شوند.

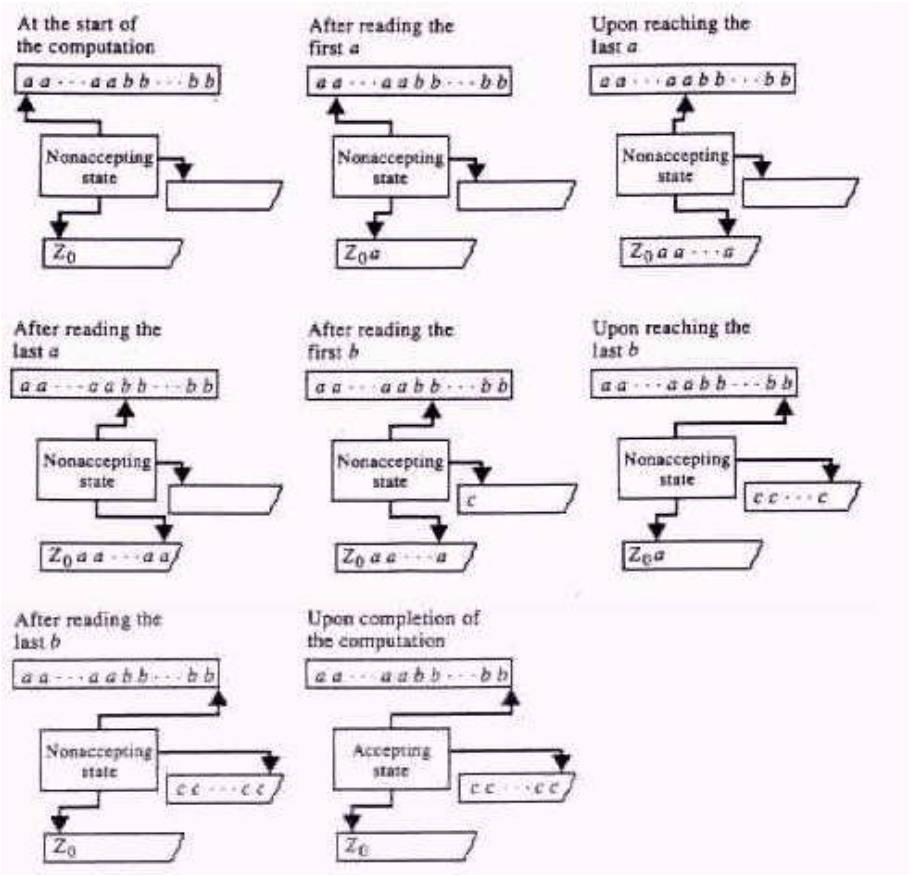
مبدل های پشته ای

هر مبدل پشته ی M می تواند به عنوان یک ماشین محاسبه گر خلاصه شده در نظر گرفته شود که شامل یک کنترل حالت متناهی ، یک نوار ورودی ، یک سر ورودی فقط خواندنی، یک نوار پشته و یک پشته ، یک سر پشته ی خواندنی و نوشتنی، یک نوار خروجی و یک سر خروجی فقط نوشتنی می شود.(شکل ۳-۲-۱ را ببینید). هر حرکت M با حالت M تعیین می شود، ورودی استفاده شده و محتوای آن برروی پشته قرارا میگیرد. هر حرکت M شامل تغییر حالت M می باشد، حداکثر یک نشانه ی ورودی را می خواند، محتوای بالای پشته را تغییر داده و حداکثر یک نشانه را رد خروجی مینویسید.



شکل ۳-۲-۱. تصویر یک مبدل پشته ای

مثال ۳-۲-۱. مبدل پشته ای M می تواند رابطه $\{ (a^i b^i, c^i) \mid i \geq 1 \}$ را با چک کردن اینکه هر ورودی به صورت a...ab...b باشد با تعداد مساوی از a,b و نوشتن همان تعداد c محاسبه کند . محاسبات M می تواند به صورت زیر باشد. (شکل ۳-۲-۲ را ببینید.)



شکل ۳-۲-۲. توصیفی از چگونگی محاسبه رابطه $\{ (a^i b^i, c^i) \mid i \geq 1 \}$ توسط مبدل پشته ای

در ابتدا فرض می شود که پشته شامل یک نشانه Z_0 که برای نشان دادن انتهای پشته استفاده می شود می باشد .M هر محاسبه را با خواندن a ها ازنوار ورودی شروع می کند و آنها را در بالای پشته قرار می دهد. در هر زمان یک نشانه از ورودی خوانده می شود.

زمانی که M،خواندن a هارا از ورودی تمام می کند، شروع به خواندن b ها میکند. زمانی که M، b ها رااز ورودی می خواند. برای هر نشانه b که از ورودی خوانده می شود یک a از روی پشته برداشته می شود. به علاوه M یک c برای هرb که

از ورودی خوانده می شود ، در خروجی می نویسد.

M ورودی را می پذیرد اگر و تنها اگر درست زمانی به انتهای ورودی برسد که به نشانه Z_0 در پشته برسد.M ورودی را رد می کنداگر قبل از رسیدن به انتهای ورودی به نشانه Z_0 برسد زیرا در این مورد تعداد بیشتری b نسبت به a دارد. M ورودی را رد می کند اگر قبل از رسیدن به نشانه Z_0 در پشته به انتهای ورودی برسد، زیرا در این مورد ورودی شامل تعداد بیشتری a نسبت به b است.

معمولا سیستم ریاضی M بر پایه یک هشت-تایی , $<Q, \Sigma, \Gamma, \Delta, \delta, q_0, Z_0, F>$ بنا میشود که یک مبدل پشته ای نامید میشود اگر شرایط زیر را دارا باشد :

Q : یک مجموعه متناهی است که عناصر Q حالات M نامیده می شوند.

Σ, Γ و Δ : الفبا هستند. Σ الفبای ورودی M نامیده می شود و عناصر آن نشانه های ورودی M نامیده میشوند. , Γ الفبای پشته ای M نامیده میشوند و عناصر آن به عنوان نشانه های پشته ای M شناخته می شوند. Δ الفبای خروجی M نامیده میشوند و عناصر آن نشانه های خروجی M نامیده می شوند .

δ : یک رابطه است از $Q\times\Gamma^*\times(\Delta U\{\varepsilon\})$ $Q\times(\sum U\{\varepsilon\})\times(\Gamma U\{\varepsilon\})$. δ تابع انتقال M نام دارد و عناصر آن قوانین انتقال هستند.

q_0 : عنصری است در Q با نام حالت اولیه M

Z_0 : عنصری است در Γ با نام انتهای پشته ی M.

F : زیر مجموعه ای از Q است . حالت های زیر مجموعه F حالات پذیرش یا حالات نهایی M نامیده می شوند.

با توجه به آنچه در ادامه آمده است ، هر قانون انتقال$(q,\alpha,\beta,(p,\gamma,\phi))$ از یک مبدل پشته به صورت

$(q,\alpha,\beta,p,\gamma,\phi)$ نوشته میشود.

مثال ۳-۲-۲ $M=<Q,\Sigma,\Gamma,\Delta,\delta,q_0,Z_0,F>$ یک مبدل پشته ای است اگر $Q=\{q_0,q_1,q_2\}$ و $\Sigma=\{a,b\}$ و $\Gamma=\{Z_0, c\}$ و , $\delta=\{(q_0,a,\varepsilon,q_0,c,\varepsilon)$ $(q_0,b,\varepsilon,q_0,c,\varepsilon)$, $(q_0,\varepsilon,\varepsilon,q_1,\varepsilon,\varepsilon)$, $(q_1,a,c,q_1,\varepsilon,a)$, $(q_1,b,c,q_1,\varepsilon,b)$,

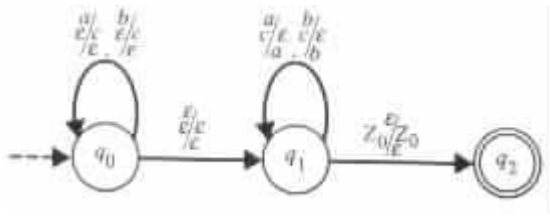
$(q_1,\varepsilon,Z_0,q_2,Z_0,\varepsilon)\}$ و $F=\{q_2\}$ باشد .

بر طبق تعریف ، در هر قانون انتقال $(q,\alpha,\beta,p,\gamma,\rho)$ ورودي p,q حالت هايي در Q هستند α يک نماد ورودي يا يک رشته خالي است ، β يک نماد پشته يا يک رشته خالي است ، γ يک رشته از نمادهاي پشته و ρ يک نماد خروجي يا يک رشته خالي است.

هر مبدل پشته ای , $M =<Q, \Sigma, \Gamma, \Delta, \delta, q_0, Z_0, F>$ می تواند به صورت زیر با وسیله نمودار انتقال نشان داده شود. برای هر حالت در Q در نمودار انتقال متناظر با هر گره یک دایره ترسیم میشود. حالت اولیه با یک فلش به گره مربوطه نشان داده می شود. حالت پذیرش با دو دایره مشخص میشود. هر قانون انتقال (q,ρ) با یک یال از گره متناظر با حالت q به گره متناظر با حالت p نشان داده می شود. به علاوه یال ها برچسب خورده می شوند با $\alpha/\beta/\gamma$.

برای نشانه گذاری مناسب، یال هایی که مبدا و مقصدشان معادل است ادغام میشوند و برچسب های آنها بوسیله کاما از هم جدا میشوند.

مثال ۳-۲-۳.نمودار انتقال مبدل پشته ای شکل ۳-۲-۲ مثال ۳-۲-۲ را نشان می دهد



شکل ۳-۲-۳. یک نمودار انتقال یک مبدل پشته ای

برچسب $\alpha/\beta/\gamma$ روی یالی که از حالت q_0 شروع می شود و درهمان حالت به پایان می

رسد ، مطابق است با قانون انتقال $(q_0, a, \varepsilon, q_0, \varepsilon, \varepsilon)$.برچسب روی یالی که از حالت q_0 شروع می شود و در حالت q_1 خاتمه می یابد مطابق است با قانون انتقال$(q_0,\varepsilon,\varepsilon,q_1,\varepsilon,\varepsilon)$.

ردیف بالا "α" در برچسب مطابق است با نوار ورودی. ردیف وسط "β" مطابق است با نوار پشته و ردیف پایین "ρ" مطابق با نوار خروجی است.

درتمام متن ،قراردادهای زیر برای هر قاعده تولید $(q,\alpha,\beta,p,\gamma,\rho)$از یک مبدل پشته ای فرض می شوند, قراردادها تأثیری بر قدرت مبدل های پشته ای ندارند و آنها برای ساده کردن بررسی مبدل های پشته ای استفاده می شوند.

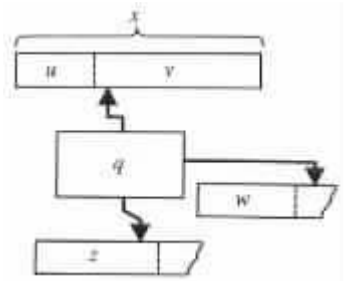
الف) اگر $Z_0=$ باشد ،آنگاه Z_0 پیشوندی از γ است .

ب) یک رشته با طول حداکثر ۲ است.

ج) اگر یک رشته به طول ۲ باشد، برابر با نماد اول است.

پیکربندی ها و حرکات مبدل های پشته ای

روی هر ورودی x از* مبدل پشته ای M چند سری از پیکربندی های ممکن را دارد. (شکل ۳-۲-٤ رابببنید.) هر پیکربندی یا توصیف آنی M، یک مجموع سه بخشی (uqv, z, w) است جایی که q یک حالت از M است ، uv=x یک ورودی M است، z یک رشته از *Γ ، از نماد های پشته است و w یک رشته از * Δ از نمادهای خروجی است. در واقع یک پیکربندی (uqv, z, w) میگوید که M روی ورودی x میتواند در داخل پشته بعد از خواندن u و نوشتن wبه حالت q با z برسد . با از دست ندادن عمومیت فرض میشود که Σ و Q از هم جدا هستند.



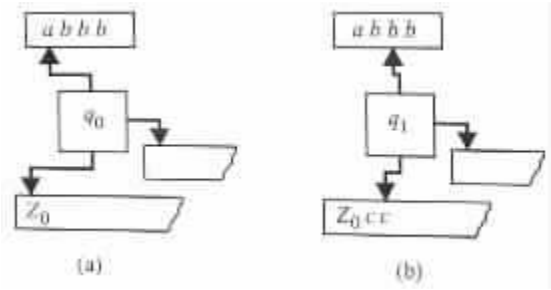
شکل ۳-۲-٤. یک وضعیت از مبدل پشته ای

به این پیکربندی یک پیکربندی اولیه گفته می شود اگر $q=q_0$، $w=u=\varepsilon$ و $z=Z_0$ باشد. چنین پیکربندی اولیه ای ، میگوید که M در حالت اولیه q_0 است با نداشتن نمادهای ورودی که خوانده شود (i.e. ,$u=\varepsilon$)، با خروجی که هنوز خالی ست(i.e.,$w=\varepsilon$)، و پشته ای که هنوز در حالت اصلی است (i.e., $z = Z_0$). به علاوه این پیکربندی می گوید که v برای M یک ورودی است.

پیکربندی ، یک پیکربندی پذیرش گفته میشود اگر $v=\varepsilon$ و q یک حالت پذیرش باشد. چنین پیکربندی پذیرشی، میگوید که ، M بعد ازخواندن همه ورودی ها به یک حالت پذیرش میرسد(i.e.,$v=\varepsilon$) و w را می نویسد. به علاوه این پیکربندی میگویدکه ورودی که M مصرف میکند برابر با v است.

مثال ۳-۲-٤. مبدل پشته ای M که نمودار انتقال آن در شکل ۳-۲-۳ داده شده است را را در نظر بگیرید.

$(q_0abbb, Z_0,\varepsilon)$,یک پیکربندی اولیه ی M روی ورودی abbb است. این پیکربندی(ε $(abq_1bb, Z_0cc,$ از M ، میگوید که M پیش از این u=ab از ورودی مصرف کرده و باقی مانده ورودی v=bb است. M با رشته Z_0cc در پشته به حالت q_1 میرسد و خروجی تا کنون خالی است. این وضعیت به ترتیب در شکل ۳-۲-۵(a) و ۳-۲-۵(b) توضیح داده شده است.



شکل ۳-۲-۵. پیکربندی های مبدل پشته ای شکل ۳-۲-۳

$(abbbq_2, Z_0, bb)$ و $(abq_2bb, Z_0cc, \epsilon)$ نیز پیکربندی های M_0 هستند. اولین پیکربندی پذیرفته می شود. دومین پیکربندی علی رغم اینکه در یک حالت پذیرفته شده قرار دارد، پذیرفته شده نیست، زیرا ورودی تا انتهای آن مصرف نشده است.

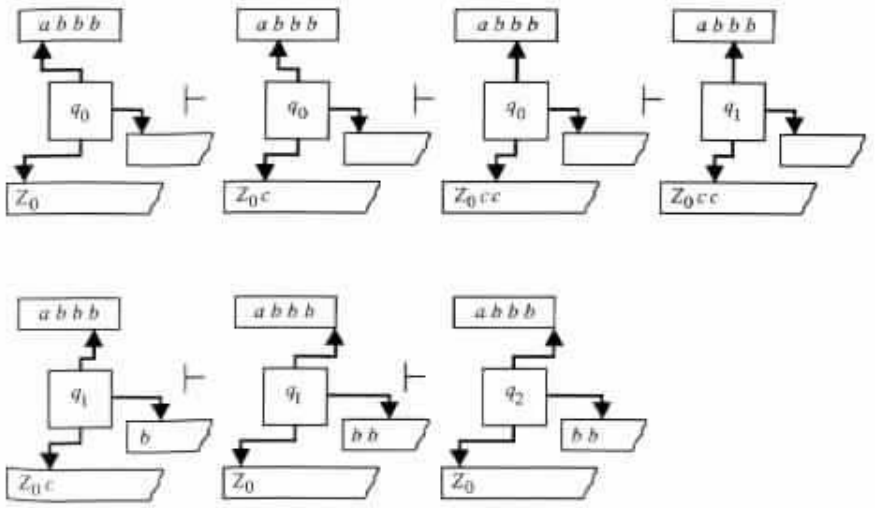
قوانین انتقال M برای تعریف کردن حرکت های ممکن M استفاده میشوند. هر حرکت مطابق با یک قانون انتقال می باشد، یک حرکت روی قانون انتقال(q,a,β,p,γ,ρ)، حالت کنترل حالت-متناهی را از q به p تغییر می دهد; a را از نوار ورودی میخواند،سر ورودی به اندازه|a| موقعیت به راست می برد،ρ را در نوار خروجی می نویسد، سر خروجی به اندازه|ρ| موقعیت به راست می برد، و در بالای (top) پشته رشته β را با رشته ی γ جایگزین می کند. (i.e از مکان سرپشته به چپ آن)، سرپشته به اندازه|β|-|γ| ، به راست حرکت می دهد. این حرکت می گوید که اگر |β|<.|γ|، POP میکند. این حرکت می گوید که اگر |γ|<.|β| ، push میکند.نماد زیر سر پشته ،نماد بالای پشته نامیده می شود.

حرکت M از پیکربندی C_1 به پیکربندی C_2 با $C_1 \vdash_M C_2$ نشان می داده می شود یا به طور ساده تر $C_1 \vdash C_2$ ،اگر M مشخص شده باشد. دنباله ای از صفر یا بیشتر حرکت های M از پیکربندی C_1 به پیکربندی C_2 با $C_1 \vdash_M^* C_2$ نشان داده می شود یا به طور ساده تر $C_1 \vdash^* C_2$ ، اگر M مشخص شده باشد.

مثال ۳-۲-۵. مبدل پشته ای که نمودار انتقال آن در شکل ۳-۲-۳ داده شده ، دنباله ای از حرکت ها روی ورودی abbb دارد که به وسیله دنباله ای از پیکربندی ها نشان داده شده است :

$(q_0abbb,Z_0,\varepsilon) \vdash (aq_0bbb,Z_0c,\varepsilon) \vdash (abq_0bb,Z_0cc,\varepsilon) \vdash (abq_1bb,Z_0cc,\varepsilon) \vdash (abbq_1b,Z_0c,b) \vdash (abbbq_1,Z_0,bb) \vdash (abbbq_2,z_0,bb).$

این دنباله فقط میتواند از یک پیکربندی اولیه شروع شده و در یک پیکربندی پذیرش برای ورودی abbb خاتمه یابد. این دنباله از پیکربندی ها درشکل ۳-۲-٦ ترسیم شده است.



شکل ۳-۲-٦. انتقال بین پیکربندی های مبدل پشته ای شکل ۳-۲-۳.

همه حرکت های M روی قوانین انتقال که در حالت q_0 شروع می شوند و خاتمه می یابند ، روی پشته push می شوند. همه حرکت های M روی قوانین انتقال، که در حالت q_1 شروع می شوند و خاتمه می یابند از روی پشته pop می شوند.

یک رشته داخل پشته از نماد آخر شروع می شود و در نماد بالایی پایان می یابد، به جز نماد آخر ، بقیه، محتویات پشته نامیده می شود. می گوییم پشته خالی ست اگر محتوای آن خالی باشد.

مثال ۳-۲-٦. M مبدل پشته ای شکل ۳-۲-۳ است . محاسبات M را روی ورودی abbb در نظر بگیرید(شکل ۳-۲-٦ را ببینید.). M با پشته خالی شروع می کند ، c را در اولین حرکت به پشته اضافه می کند. بعد از دومین حرکت محتوای پشته cc می باشد. محتوای پشته طی سومین حرکت تغییر نمیکند.

قطعیت و غیر قطعیت در مبدل های پشته ای

تعریف قطعیت و غیر قطعیت در مبدل های پشته ای ، عملا شبیه آنچه برای مبدل های حالت–متناهی گفته شد می باشد. تفاوتی که وجود دارد در جزئیات است.

به یک مبدل پشته ای , $M =, <Q\Sigma, \Gamma, \Delta, \delta, q_0, Z_0, F>$, قطعی گفته می شود اگر برای هر حالت q در Q ; هر نماد ورودی a در Σ و هرنماد پشته ای z در Γ $\delta(q, \epsilon, \epsilon) \cup$ یک مجموعه باشد که حداکثر شامل یک عنصر $\delta(q, a, Z) \cup \delta(q, a, \epsilon) \cup \delta(q, \epsilon, Z)$ است.

درواقع M قطعی است اگر حالت و نماد بالای پشته برای تصمیم گیری اینکه نماد از ورودی خوانده نشده، کافی باشند و نیز حالت و نماد بالای پشته وورودی خوانده شده برای تعیین قانون انتقالی که استفاده شده کافی باشند.

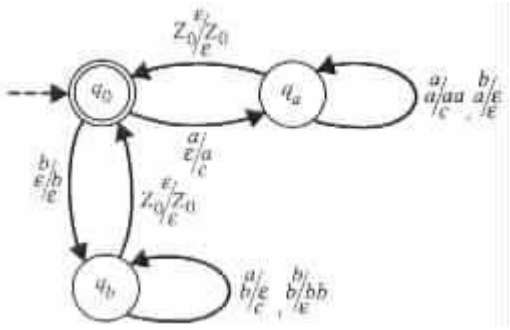
یک مبدل پشته ای را غیر قطعی میگویند اگر یک مبدل پشته ای قطعی نباشد.

مثال ۳–۲–۷. M_1 یک مبدل پشته ای است که نمودار انتقال آن در شکل ۳–۲–۳ داده شده است.

در حرکت از حالت q_1 مبدل پشته ای M_1 یک نماد ورودی را میخواند اگر و تنها اگر بالاترین نماد پشته Z_0 نباشد. اگر نماد Z_0 نباشد ، نماد بعدی به وسیله قانون انتقالی که در حرکت استفاده شده در ورودی منحصر به فرد، تعیین میشود ، اگر بالاترین نماد پشته Z_0باشد ، M_1 باید قانون انتقالی که منجر به q_2 می شود را استفاده کند. بنابراین حرکت هایی که از حالت q_1سرچشمه می گیرند می توانند به طور موضعی قطعی باشند.

از طرف دیگر ، حرکت ها از حالت q_0نمی توانند به طور موضعی قطعی باشند ، زیرا بالاترین نماد پشته اگر یک نماد ورودی خوانده شده در حرکت باشد برای تصمیم گیری کافی نیست.

M_1 یک مبدل پشته ای غیر قطعی است. در حالی که مبدل پشته ای M_2 که نمودار انتقال آن در شکل ۳–۲–۷ داده شده ، قطعی است.



شکل ۳-۲-۷. یک مبدل پشته ای قطعی

در حرکت از حالت q_0 مبدل پشته ای M_2، یک نماد ورودی را میخواند. اگر نماد خوانده شده a باشد، M_2به حالت q_a می رود. اگر نماد خوانده شده b باشد M_2 به حالت q_b می رود.

بالاترین نماد در پشته تعیین میکند M_2 روی یک حرکت ناشی شده درحالت q_a باید به حالت q_0 وارد شود یا به حالت q_a . اگر بالاترین نماد در پشته Z_0باشد ، M_2 به حالت q_0حرکت میکند. اگر بالاترین نماد a باشد M به حالت q_aحرکت میکند. درمورد قبلی، اگر نماد ورودی خوانده شده a باشد ،M از قانون انتقال (q_a, a, a, q_a, aa, c) استفاده می کند و اگر نماد ورودی خوانده شده b باشد، از قانون انتقال $(q_a, b, a, q_a, \epsilon, \epsilon)$ استفاده می کند.

محاسبات مبدل های پشته ای

محاسبات مبدل های پشته ای نیز مانند مبدل های حالت- متناهی تعریف می شود. یک محاسبه ی پذیرفته شده از مبدل پشته ای M ، دنباله ای از حرکت های M است که ازیک پیکربندی اولیه شروع میشود و در پذیرش آن خاتمه می یابد. یک محاسبه پذیرفته نشده یا رد شده از M ، دنباله ای از حرکت های روی یک ورودی x است برای حصول شرایط زیر :

الف) دنباله از پیکربندی اولیه ی M روی ورودی x شروع می شود.

ب) اگر دنباله متناهی باشد ، در پیکربندی که حرکت در آن امکان ندارد، خاتمه می یابد.
ج) M محاسبه ی پذیرفته شده روی x ندارد.

به هر محاسبه پذیرفته شده و هر محاسبه پذیرفته نشده ی M یک محاسبه ی M گفته می شود.

به یک محاسبه ، محاسبه توقف گفته می شود اگر شامل تعداد متناهی حرکت باشد.

مثال ۳-۲-۸. مبدل پشته ای M که در شکل ۳-۲-۷ نمودار انتقال آن داده شده را در نظر بگیرید.

مبدل پشته ای فقط برای ورودی هایی که تعداد a و b یکسان دارند محاسبات پذیرفته شده دارد. روی هر ورودی w که مبدل پشته ای یک محاسبه پذیرفته شده دارد ، جایی که تعداد b = تعداد i = a ، باشد ، رشته ی c^i را در نوار خروجی می نویسد.

مبدل پشته ای وارد حالت q_0 می شود تا جایی که قسمتی از ورودی خوانده شده شامل تعداد a و b یکسان باشد. مبدل پشته ای وارد حالت q_a می شود تا جایی که در ورودی خوانده شده، تعداد a بیشتر از b باشد. به همان صورت ، مبدل پشته ای وارد حالت q_b می شود تا جایی که آن قسمت از ورودی خوانده شده دارای تعداد b بیشتر از a باشد.

روی ورودی aabbba ، مبدل پشته ای Mتنها یک محاسبه دارد.M محاسبات را با حرکت از حالت q_0به حالت q_a شروع میکند ، تا زمانی که aرا میخواند ، c را می نویسد و a را روی پشته میگذارد.در حرکت دوم M، a را میخواند ، c را می نویسد و a را روی پشته میگذارد و به حالت q_a برمیگردد.

در سومین و چهارمین حرکت M، b را میخواند و a را از روی پشته بر میدارد و به حالت q_a باز میگردد.در حرکت پنجم M بدون خواندن و نوشتن یا تغییر دادن محتوای پشته به حالت q_0 میرود.در حرکت ششم M، b را میخواند ، b را روی پشته میگذارد و به حالت q_b میرود.در حرکت هفتم M، a را میخواند ، b را از روی پشته برمیدارد ، c را مینویسد وبه حالت q_b بازمیگردد.محاسبه در یک پیکربندی پذیرش با یک حرکت از حالت q_bبه حالت q_0بدون خواندن ورودی ، نوشتن خروجی و تغییر در محتوای

پشته نهایی میشود.

طبق تعریف هر حرکت در هر محاسبه باید طبق قانون انتقال باشد که محاسبات را در یک سیر نگه میدارد که سرانجام باعث میشود محاسبات تمام ورودی را خوانده و در حالت پذیرش توقف کند؛تا زمانی که بیشتر از یک چنین گزینه هایی در مجموعه قوانین انتقال ممکن وجود دارد ، هر کدام از این گزینه ها میتوانند انتخاب شوند.به همین صورت ، جایی که قوانین انتقال ممکن وجود نداردو شرایط بالا برقرار نیست ، پس هر کدام از قواعد تولید میتوانند انتخاب شوند.این مشاهده کمک میکند که ما ببینیم محاسبات مبدل های پشته ای به وسیله ی عامل های فرضی با قدرت استثنایی انجام میشود.

به یک ورودی x پذیرفته شده یا شناخته شده به وسیله ی مبدل پشته ای M گفته میشود اگر M یک محاسبه ی پذیرش روی x داشته باشد.زمانی که یک محاسبه پذیرش روی x با پیکربندی به فرم (xq_f, z, w) نهایی میشود میگوییم یک خروجی w داریم.خروجی یک محاسبه ی پذیرفته نشده تعریف نشده فرض میشود .

مثال ۳-۲-۹.مبدل پشته ای M که نمودار آن در شکل ۳-۲-۳ داده شده است را در نظر بگیرید.

مبدل پشته ای دقیقا ورودی که طول زوج دارد را میپذیرد.در هر محاسبه ی پذیرش مبدل پشته ای دومین نیمه ی ورودی را به عنوان خروجی میدهد.

تا زمانی که مبدل پشته ای در حالت q_0 است ، مکررا یک نماد ورودی را میخواند و c را روی پشته میگذارد.متناوباً تا زمانی که مبدل پشته ای در حالت q_1 مکرراً یک نماد ورودی را میخواند و c را از روی پشته برمیدارد.

به محض رسیدن به پشته ی خالی ، مبدل پشته ای برای تایید اینکه به انتهای ورودی رسیده است از حالت q_1 به حالت q_2 منتقل میشود.بنابراین در محاسبه ی پذیرش ، مبدل پشته ای به محض رسیدن به میانه ی ورودی باید از حالت q_0به حالت q_1 منتقل شود.

در ورودی abbb مبدل پشته ای با دو حرکت محاسبات را شروع میکند.دو نماد ورودی اول را میخواند دو c روی پشته گذاشته و به حالت q_0 باز میکردد.در سومین حرکت مبدل پشته ای از حالت q_0 به حالت q_1 میرود.

مبدل پشته ای با دوحرکت ادامه میدهد ، دو نماد انتهای ورودی را خوانده ، دو c از روی پشته برمیدارد و ورودی خوانده شده را روی نوار خارجی کپی میکند.پشته محاسبات را روی ورودی abbb با حرکت از حالت q_1 به حالت q_2 خاتمه میدهد.

اگر M روی ورودی abbb بیش از دو نماد ورودی را در حرکت از حالت q_0 بخواند به دلیل افزونی نماد ها در پشته در حالت q_1 متوقف می شود.اگر Mروی ورودی abbb کمتر از دو نماد ورودی را در حرکت از حالت q_1بخواند ، به دلیل فقدان نمادها در پشته در حالت q_1متوقف میشود.در هر مورد دنباله حرکات محاسبات M تعریف نشده است.

این مثال نشان میدهد که روی ورودی پذیرفته شده بوسیله ی مبدل پشته ای ، ممکن است مبدل دارای دنباله اجرای قوانین انتقال باشد که به عنوان محاسبه آنها را در نظر نمی گیرد.

سایر تعاریف مانندروابط محاسبه شده بوسیله ی مبدل های پشته ای ، زبانهای پذیرفته شده توسط مبدل های پشته ای ،و زبانها ی تصمیم گیری شده با مبدل های پشته ای ، شبیه مبدل های حالت متناهی داده شده در بخش ۲-۲ میباشد.

مثال۳-۲-۱۰.مبدل پشته ای M_1 که نمودار انتقال آن در شکل ۳-۲-۳ داده شده است ، روابط $\{ (xy, y) \mid xy \text{ is in } \{a, b\}^*, \text{ and } |x| = |y|) \}$. را محاسبه میکند.

مبدل پشته ای M_2 که نمودار آن در شکل ۳-۲-۶ داده شده است روابط $\{ (x, c^i) \mid) \}$ $\{ (x \text{ is in } \{a, b\}^*, \text{ and } i = (\text{number of a's in x}) = (\text{number of b's in x}$ را محاسبه میکند.

از برنامه های بازگشتی با دامنه ی متناهی تا مبدل های پشته ای

شبیه سازی برنامه های بازگشتی با دامنه ی متناهی بوسیله ی مبدل های پشته ای ، مشابه شبیه سازی برنامه های با حافظه ی متناهی بوسیله ی مبدل های حالت–متناهی میباشد. ، تا زمانیکه با فراخوانی و دستورات بازگشت مواجه نباشیم.در چنین مواردی مبدل های پشته ای فقط تمام حالت برنامه را بدون استفاده از پشته دنبال میکند.

به محض رسیدن به دستورات فراخوانی ، مبدل های پشته ای از جایی که فراخوانی ها سرچشمه میگیرند از پشته هایشان برای ثبت حالات استفاده میکنند.به محض اینکه به دستورات بازگشت می رسد مبدل های پشته ای از پشته برای به دست آوردن فراخوانی مطابق ، از حالاتی که فعال هستند و از این اطلاعات برای شبیه سازی کردن دستورات بازگشت استفاده میکند.

به طور اختصاصی هر برنامه ی بازگشتی دامنه ی متناهی P را در نظر بگیرید.فرض میشود که P، m متغیر به صورت $x_1, \ldots, x_m$ و k قطعه دستور به صورت $I_1, \ldots, I_k$ دارد.مقدار اولیه با ⊙ در دامنه ی متغیر های P نشان داده میشود.یک حالت P یک $(m+1)$تایی است به صورت $[i, v_1, \ldots, v_m]$ است، که i یک عدد صحیح مثبت است که از c بزرگتر نیست و $v_1, \ldots, v_m$ مقادیری از دامنه ی P هستند.

رفتار محاسباتی P میتواند توسط مبدل های پشته ای $M=<Q,\Sigma, \Gamma, \Delta, \delta, q_0, Z_0, F>$ ساخته شود که حالات آن برای ثبت حالات P استفاده میشود ، و پشته های آن برای ثبت حالاتی از P که اجرای رویه ی آنها فعال است و هنوز غیر فعال نشده است استفاده میشود.

$M=<Q,\Sigma, \Gamma, \Delta, \delta, q_0, Z_0, F>$ به صورت زیر تعریف میشود:

Q : مجموعه ای است شامل تمام حالاتی که P بدست می آورد.

Σ : مجموعه ای است شامل تمام مقادیر ورودی که P میتواند بخواند.

Γ : مجموعه ای است شامل Z_0و حالات فراخوانی در Q Z_0 .یک عنصر جدید که در Q نیست فرض میشود و یک حالت فراخوانی حالتی فرض میشود که مطابق با یک دستور فراخوانی است.

Δ : مجموعه ای است شامل تمام مقادیر خروجی که P میتواند بنویسد.

q_0 : به عنوان حالت $[0, \ldots, 0]$, از P مشخص میشود.

F : به عنوان مجموعه ی حالاتی در Q مشخص میشود که مطابق با دستوری به صورت if *eof* then accept است.

δ : محتوی یک قاعده ی انتقال به صورت $(q,\alpha,\beta,p,\gamma,\rho)$ است اگر و تنها اگر $q = [i, u_1, \ldots, u_m]$ و $p = [j, v_1, \ldots, v_m]$ حالاتی در Q است که شروط زیر را ارضا کند:

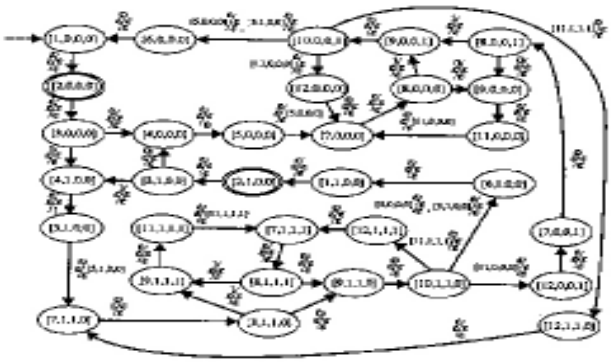
الف) با اجرای قطعه دستور I_i ، برنامه ی P (با مقدار $u_1, \ldots, u_m$ به ترتیب داخل متغیر های $x_1, \ldots, x_m$) میتواند α را بخواند ، β را بنویسد و به قطعه دستور I_j با مقادیر $v_1, \ldots, v_m$ داخل متغیر ها برسد.

ب) اگر I_i دستور فراخوانی و بازگشت نباشد ، آنگاه $\gamma = \lambda$.که پشته نادیده گرفته می شود.

ج)اگر I_i یک دستور فراخوانی باشد آنگاه $\gamma = \rho$ و $q = q_0$. به صورتی که حالت q از P قبل از درخواست رویه روی پشته گذاشته میشود.حالتی که ثبت می شود اجازه میدهد شبیه سازی دستور بازگشتی انجام شود که رویه ای را که بوسیله ی I_i فعال شده است ، غیر فعال میکند.

د) اگر I_i یک دستور بازگشت باشد ، آنگاه β حالتی از P فرض میشود و انتقال از حالت q به حالت P به صورتی انجام میشود که فراخوانی ساخته شده در حالت β را غیر فعال میکند.همچنین در نمونه ی $\gamma = \lambda$.

مثال ۳-۲-۱۱.بررسی برنامه ی بازگشتی با دامنه ی متناهی P در شکل ۳-۱-۱ با $\{0, 1\}$ به عنوان دامنه ی متغیر هایش.برنامه با مبدل پشته ای خلاصه میشود که نمودار آن در شکل ۳-۲-۸ داده شده است.



شکل ۳-۲-۸.نمودار انتقال یک مبدل پشته ای که برنامه بازگشتی با دامنه ی متناهی شکل ۳-۲-۸ را کاراکتر بندی میکند.

در نمودار انتقال ، حالت [i, x, y, z] با قطعه دستور I_i با مقادیر x, y, z در متغیر های x, y, z به ترتیب ، متناظر است.

در حرکت از حالت [۳, ۰, ۰, ۰] به حالت [٤, ۰, ۰, ۰] مبدل پشته ای مقدار ۰ را داخل x میخواند.در حرکت از حالت [۳, ۰, ۰, ۰] به حالت [٤, ۱, ۰, ۰] مبدل پشته ای مقدار ۱ را داخل x میخواند.

هر حرکت از حالت [۵, ۱, ۰, ۰] به حالت [۷, ۱, ۱, ۰] معادل یک فراخوانی تابع است و هر چنین حرکتی حالت [۵, ۱, ۰, ۰] را در پشته ذخیره می کند. در هر چنین حرکتی ، مقدار y در حالت [۷, ۱, ۱, ۰] بویسله مقدار x هدر حالت [۵, ۱, ۰, ۰] تعیین می شود و مقدار x و z در [۷, ۱, ۱, ۰] بوسیله مقدار x و z در حالت [۵, ۱, ۰, ۰] تعیین می شود.

هر حرکت از حالت [۱۰, ۱, ۱, ۰] به حالت [٦, ۱, ۰, ۰] که از قانون انتقال(, , [5, 1, 0, 0], [6, 1, 0, 0], , [10, 1, 1, 0]) , استفاده میکند با یک اجرای دستور بازگشت برای یک فراخوانی که از حالت [۵, ۱, ۰, ۰] به دست آمده است مشابه است. مقدار x در حالت [٦, ۱, ۰, ۰] به وسیله ی مقدار y در حالت [۱۰, ۱, ۱, ۰] تعیین میگردد.مقادیر y و z در حالت [٦, ۱, ۰, ۰] با مقادیر y و z در حالت [۵, ۱, ۰, ۰] تعیین میشود.

مبدل پشته ای محاسبات زیر را روی ورودی۰۰۱۱ انجام میدهد.

$([1,0,0,0]0011, Z_0, \epsilon) \vdash ([2,0,0,0]0011, Z_0, \epsilon)$
$\vdash ([3,0,0,0]0011, Z_0, \epsilon)$
$\vdash (0[5,0,0,0]011, Z_0, 0)$
$\vdash (0[7,0,0,0]011, Z_0[5,0,0,0], 0)$
$\vdash (0[8,0,0,0]011, Z_0[5,0,0,0], 0)$
$\vdash (00[9,0,0,0]11, Z_0[5,0,0,0], 0)$
$\vdash (00[11,0,0,0]11, Z_0[5,0,0,0], 0)$
$\vdash (00[7,0,0,0]11, Z_0[5,0,0,0][11,0,0,0], 0)$
$\vdash (00[8,0,0,0]11, Z_0[5,0,0,0][11,0,0,0], 0)$
$\vdash (001[9,0,0,1]1, Z_0[5,0,0,0][11,0,0,0], 0)$
$\vdash (001[10,0,0,1]1, Z_0[5,0,0,0][11,0,0,0], 0)$
$\vdash (001[12,0,0,0]1, Z_0[5,0,0,0], 0)$
$\vdash (001[7,0,0,0]1, Z_0[5,0,0,0], 0)$
$\vdash (001[8,0,0,0]1, Z_0[5,0,0,0], 0)$
$\vdash (0011[9,0,0,1], Z_0[5,0,0,0], 0)$
$\vdash (0011[10,0,0,1], Z_0[5,0,0,0], 0)$
$\vdash (0011[6,0,0,0], Z_0, 0)$
$\vdash (0011[1,0,0,0], Z_0, 0)$

از مبدل پشته ای تا برنامه های بازگشتی با دامنه ی متناهی:

با استفاده از بحث قبلی ، نتیجه میگیریم که الگوریتمی وجود دارد که هر برنامه ی بازگشتی با دامنه ی متناهی مفروض را در مبدل پشته ای معادل با آن تفسیر میکند.از طرف دیگر ، الگوریتمی موجود است که یک برنامه ی بازگشتی با دامنه ی متناهی معادل برای هر مبدل پشته ای مفروض $M=<Q, \Sigma, \Gamma, \Delta, \delta, q_0, Z_0, F>$, استخراج میکند. برنامه ی بازگشتی با دامنه ی متناهی میتواند یک برنامه ی table-driven باشد به صورتی که در شکل ۳-۲-۹ نشان داده شده است.برنامه ی مبدل پشته ای را به روشی مانند شبیه سازی یک مبدل با حالت متناهی به وسیله ی یک برنامه با حافظه ی متناهی به صورتی که در قسمت ۲-۲ نشان داده شد شبیه سازی میشود .تفاوت اصلی در شبیه سازی اثر پشته میباشد.

```
state := q0
do
   top := Z0
   call RP(top)                    /* Record the bottom pushdown symbol
Z0. */
until false
procedure RP(top)
```

```
   do
 /* Accept if an accepting state of M is reached at the end of the
input. */
      if F(state) then
         if eof then accept
/* Nondeterministically find the entries of the transition rule (q, a, b, p,
g, d) that M uses in the next simulated move.  */
      do in := e or read in until true      /*      in := a */
      do pop := e or pop := top until true      /*     pop := b */
      next_ state := ?           /* next_ state := p */
      push := ?                  /*       push := g */
      out := ?                   /*          out := d */
      if not d (state,in,pop,next_ state,push,out) then reject
      /* Simulate the next move of M. */
      state := next_ state
      if out≠ e then write out
      if pop≠ e then return
      if push≠ e then call RP(push)
   until false
end
```

شکل ۳-۲-۹.یک برنامه ی بازگشتی با دامنه ی متناهی table-driven برای شبیه سازی مبدل پشته ای.

برنامه از متغیر state برای ذخیره کردن حالاتی که M در حرکتهایش دارند ، استفاده می کند. متغیر top برای ذخیره کردن بالاترین نماد روی پشته است، متغیر in برای ذخیره کردن ورودی هایی است که M در حرکت هایش از آنها استفاده می کند ، متغیر next-state برای ذخیره کردن حالاتی است که حرکت هایش به آن ها وارد می شود ، متغیر pop برای ذخیره کردن زیررشته هایی که بر روی پشته قرار می گیرند ،متغیر push برای ذخیره کردن تغییراتی که باید روی پشته اعمال شود، و یک متغیر out برای ذخیره کردن خروجی که باید در حرکت های M نوشته شود.

محتوی پشته در طول بازگشت به طور غیر مستقیم ذخیره می شود. گذاشتن هر نماد بر روی یک پشته با یک فراخوانی بازگشتی شبیه سازی می شود و برداشتن هر نماد از روی پشته با یک بازگشت شبیه سازی می شود .

برنامه اصلی به متغیر state مقدار اولیه q_0 می دهد و برای ذخیره کردن پشته ای که فقط محتوی Z_0 است RP را فراخوانی می کند .

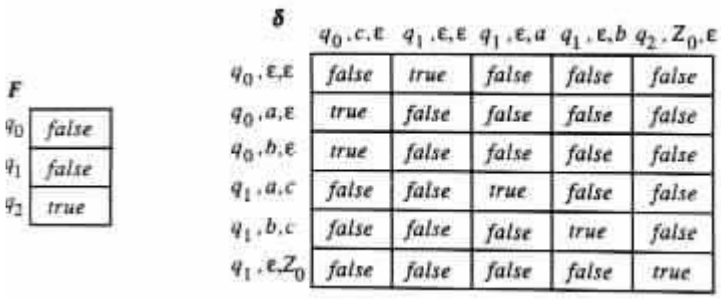
بدنه ی رویه بازگشتی RP شامل یک حلقه ی نامتناهی است . تکرار هر حلقه با بررسی اینکه یک حالت پذیرش از M در انتهای ورودی دریافت شده , آغاز می شود . اگر با چنین موردی مواجه شویم , برنامه در یک پیکربندی پذیرش متوقف می شود . در غیر این صورت , برنامه تنها یک حرکت M را شبیه سازی می کند.گزاره F برای معین کردن اینکه آیا state در یک حالت پذیرش است یا خیر استفاده می شود.

شبیه سازی هر حرکت M بوسیله یک روش غیر قطعی انجام می شود . برنامه حدس می زند که ورودی ای که باید خوانده شود در قسمت بالای پشته جایگذاری می شود , حالتی که به دست می آید , با بالای پشته تعویض می شود و خروجی نوشته می شود. سپس برنامه از گزاره ی δ برای تعیین کردن مقدار حدس زده شده ی مناسب استفاده می کند . اگر مشخص شود که حدس مناسب نیست , برنامه شبیه سازی را رد می کند . در غیر این صورت , برنامه تغییرات اعمال شده را به عنوان نتیجه حدس زده شده ی قاعده انتقال ذخیره می کند .

متغیرهای برنامه با دامنه ی { QUINTUPLE } فرض می شود .با e به عنوان نماد جدید بعلاوه بدون از دست دادن عمومیت فرض می شودکه هر قاعده ی انتقال$(q,\alpha,\beta,p,\gamma,\rho)$ از M هر رابطه ی $|\beta|+|\gamma|=1$ یا $\beta=Z_0=\gamma$ را ارضا می کند . آخرین فرض مجبور می کند از موقعیتی که در آن برداشتن از و اضافه کردن یک نماد به پشته , با حرکت یکسان از M شبیه سازی شود .

مثال ۳-۲-۱۲ برای مبدل پشته شکل ۳,۲,۳ برنامه ی table – driven دامنه ای از متغیرهای مساوی با

$\{a, b, Z_0, c, q_0, q_1, q_2, e\}$دارد . مقدار حقیقی گزاره ی F و δ با جدول مطابق شکل ۱۰,۲,۳ تعریف شده است .



F	
q_0	false
q_1	false
q_2	true

δ	q_0,c,ε	$q_1,\varepsilon,\varepsilon$	q_1,ε,a	q_1,ε,b	q_2,Z_0,ε
$q_0,\varepsilon,\varepsilon$	false	true	false	false	false
q_0,a,ε	true	false	false	false	false
q_0,b,ε	true	false	false	false	false
q_1,a,c	false	false	true	false	false
q_1,b,c	false	false	false	true	false
q_1,ε,Z_0	false	false	false	false	true

شکل ۳-۲-۱۰. جدول هایی برای یک برنامه ی بازگشتی با دامنه متناهی table-driven که مبدل پشته شکل ۳-۲-۳ را شبیه سازی میکند.

(F و δ با مقدار False برای مباحثی که که در جدول نشان داده نشده است، فرض می شود.)

مبدل پشته هم چنین می تواند به وسیله ی برنامه ی nontable-driven شکل ۳-۲-۱۱ شبیه سازی شود.

```
state := q0
next_ top := Z0
call RP(next_ top(

procedure RP(top(

   do
      if state = q0 then
         do
            read in
            if (in≠ a) and (in≠ b) then reject
            next_ top := c
         call RP(next_ top(

         or
            state := q1
        until true
    if state = q1 then
         do
            if top = Z0 then state := q2
           if top = c then
                do
                   read in
                   if (in≠ a) and (in≠ b) then reject
                   write in
```

```
                return
            until true
        until false
    if state = q2 then
        if eof then accept
    until false
end
```

شکل ۳-۲-۱۱. یک برنامه ی بازگشتی با دامنه متناهی nontable-driven برای شبیه سازی مبدل پشته ای شکل ۳-۲-۳ .

مانند آنچه در بخش ۲-۲ برای مبدل های حالت متناهی بحث شد، برنامه های بازگشتی با حافظه ی متناهی می توانند به صورت غیر قطعی اصلاح شوند هرگاه مبدل پشته ای استفاده شده قطعی باشد.

فرمول بندی بحث قبل در قضیه ی زیر به دست می آید.

قضیه ۳-۲-۱.رابطه ای ، با یک برنامه بازگشتی با دامنه ی متناهی غیر قطعی (قطعی) محاسبه پذیراست اگر و تنها اگر با یک مبدل پشته ای غیر قطعی (قطعی) قابل محاسبه باشد.

اتوماتهای پشته ای

مبدل های پشته ای ای که خروجی های ترکیبی را نمی پذیرند ماشین های پشته ای نامیده می شوند. معمولاً ماشین پشته ای یک چندتایی,<Q, Σ, Γ, δ ,q_0, Z_0,F> است جایی که Q ,Σ, Γ,Z_0, F, q_0 به عنوان مبدل های پشته ای تعریف شده باشند و δ رابطه ای است از $Q\times(\Sigma \cup \{\varepsilon\})\times(\Gamma \cup \{\varepsilon\})$ به $Q\times\Gamma^*$.

در مبدل های پشته ای ، شروط زیر برای هر قانون انتقال (q,a, β, p, γ,ρ) از یک ماشین پشته ای فرض میشود.

الف) اگر Z_0=β باشد آنگاه Z_0 یک پیشوند برای γ است.

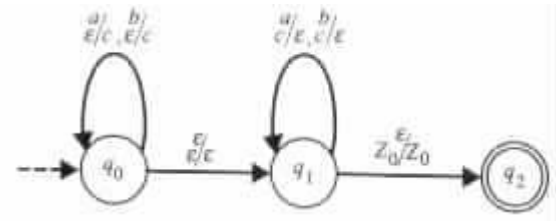
ب) γ رشته ای با طول حداکثر ۲ است.

ج) اگر γ رشته ای با طول ۲ باشد، آنگاه β با اولین نماد در γ برابر است.

نمودارهای انتقالی شبیه به آنچه برای نشان دادن مبدل های پشته ای استفاده شده، می تواند برای نشان دادن اتوماتهای پشته ای استفاده شود. تنها تفاوتی که وجود دارد این است که برچسب های یال ها شامل خروجی ها نمی شود.

مثال ۳-۲-۱۳. ماشین پشته ای M که از مبدل پشته ای شکل ۳-۲-۳ بدست می آید شامل $<Q,\Sigma, \Gamma, \delta,q_0, Z_0, F>$, است که در آن $Q = \{q_0, q_1, q_2\},\Sigma = \{a, b\}$, $\Gamma\{Z_0, c\}$,

$\delta = \{ (q_0,a,\varepsilon,q_0,c) , (q_0,b,\varepsilon,q_0,c) , (q_0,\varepsilon,\varepsilon,q_1,\varepsilon), (q_1,a,c,q_1,\varepsilon) , (q_1, b,c,q_1,\varepsilon) ,(q_1,\varepsilon,Z_0,q_2,Z_0)\}$ و $F = \{q_2\}$ می باشد. ماشین پشته ای در نمودار شکل ۳-۲-۱۲ نشان داده شده است.

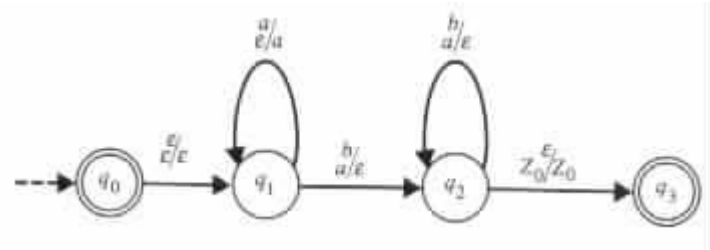


شکل ۳-۲-۱۲. نمودار انتقال یک اتوماتای پشته ای

یک ماشین پشته ای قطعی است که برای هر حالت q در Q، هر نماد a در Σ و هر نماد پشته ای z در Γ، مجموعه ی $\delta(q, a, \epsilon) \cup \delta(q,\epsilon, Z) \cup \delta(q, \epsilon, \epsilon)$ چندتایی شامل حداکثر یک عنصر باشد. یک ماشین پشته ای غیر قطعی نامیده می شود اگر ماشین پشته ای قطعی نباشد.

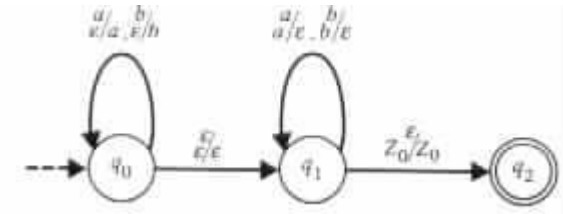
یک پیکربندی یا توصیف آنی از ماشین پشته ای جفت (uqv,z) است که q یک حالت در Q، uv رشته ای در Σ^* و z رشته ای در Γ^* است. باقی تعاریف مانند پیکربندی های اولیه و نهایی و $\vdash^*$,$\vdash_M^*$,$\vdash$, $\vdash_M$و پذیرش و شناسایی و تصمیم پذیری زبان با یک ماشین پشته ای، مشابه تعاریفی است که برای مبدل های پشته ای ارائه شد.

مثال ۳-۲-۱٤. نمودار انتقال شکل ۳-۲-۱۳ را در نظر بگیرید



شکل ۳-۲-۱۳. نمودار انتقال برای یک ماشین پشته ای قطعی که $\{ a^i b^i \mid i \geq 0 \}$ را می پذیرد.

ماشین پشته ای قطعی با $<\{q_0, q_1, q_2, q_3\}, \{a, b\}, \epsilon, \epsilon, q_1, \epsilon), (q_1, a, \epsilon, q_1, a), (q_1, b, a, q_2, \epsilon), (q_2, b, a, q_2, \epsilon), (q_2, \epsilon, Z_0, q_3, Z_0)\}, q_0, Z_0, \{q_3\}>$ مشخص میشود.ماشین پشته ای زبان $\{ a^i b^i \mid i \geq 0 \}$ را می پذیرد.ماشین aها را از ورودی می خواند و آنها را روی پشته قرار می دهد ، تازمانی که در حالت q_1 باشد. سپس b ها را از ورودی می خواند و به ازای هر b یک a از روی پشته برمی دارد. تازمانی که bها را می خواند، ماشین در حالت q_2 می ماند. ماشین پشته ای تنها زمانی که به یک تعداد aوb خوانده باشد، وارد حالت پذیرش q_3 می شود.



شکل ۳-۲-۱٤. نمودار انتقال یک ماشین پشته ای غیرقطعی که $\{ ww^{rev} \mid w \text{ is in } \}^*$ {a, b} را می پذیرد.

نمودار حالت در شکل ۳-۲-۱٤ مربوط است به یک ماشین پشته ای غیر قطعی که زبان $\{ww^{rev} \mid w \text{ is in } \{a, b\}^*\}$ را می پذیرد. درحالت q_0 ماشین w را می خواند و

به صورت معکوس روی پشته قرار می دهد. از طرفی در حالت q_1 ماشین w^{rev} را می خواند و بارشته ای که داخل پشته است مقایسه می کند.

۳-۳.زبانهای مستقل از متن

اتوماتهای پشته ای میتواند با گرامرهای نوع۲ ، یا به طور معادل با گرامرهای مستقل از متن مشخص شود.

یک گرامر نوع ۰ به صورت , $G = <N,\Sigma,P, S>$ گرامر مستقل از متن خوانده میشود اگر هر قاعده ی تولید آن دقیقا یک نماد غیر پایانه در سمت چپ خود داشته باشد ، یا اگر هر کدام از قوانین تولید آن به صورت $A \to a$ باشند.

گرامری مستقل از متن خوانده میشود که مکانیسمی برای محدود کردن کاربری یک قاعده ی تولید به صورت $A \to a$ داخل برخی از متون خاص نداشته باشد.به هر حال،در یک گرامر نوع ۰ از یک قاعده ی تولید به صورت $\beta A\gamma \to \beta\alpha\gamma$ برای مشخص کردن اینکه $A \to a$ فقط در زمینه β و γ به کار رود ، استفاده میشود.

زبانهایی که به وسیله ی گرامرهای مستقل از متن تولید میشوند زبانهای مستقل از متن خوانده میشوند.

مثال۳-۲-۱. زبان $\{ a^{i}_1 b^{i}_1 a^{i}_2 b^{i}_2 \dots a^{i}_n b^{i}_n \mid n, i_1, \dots, i_n \geq 0 \}$ به وسیله ی یک گرامر مستقل از متن , $<N,\Sigma,P, S>$, تولید میشود که قواعد تولید آن در زیر داده شده است:

$$\begin{aligned} S &\to SS \\ &\to A \\ &\to \epsilon \\ A &\to aAb \\ &\to ab \end{aligned}$$

از گرامرهای مستقل از متن تا گرامرهای نوع۲

یادآوری می شود که یک گرامر نوع ۲ ، گرامر مستقل از متن به صورت, $\Sigma,P, S>$

$G = <N,$ است که در آن $A \to \varepsilon$ در P نشان میدهد که $A = S$ و سمت راست قواعد تولید شامل S نباشد.با این نظریه ، مشخص میشود که گرامرهای مستقل از متن و گرامرهای نوع۲ برای کلاسهای یکسان از زبانها به عنوان گرامرهای "maximal" و "minimal" عمل میکنند.

قضیه۳-۳-۱: هر زبان مستقل از متن یک زبان نوع ۲ نیز هست.

اثبات:هر گرامر مستقل از متن , $G_1 = <N, \Sigma, P_1, S_1>$, را در نظر میگیریم.یک گرامر نوع۲ مانند , $G_2 = <N \cup \{S_2\}, \Sigma, P_2, S_2>$ ، $L(G_2) = L(G_1)$ را ارضا میکند.اگر S_2 یک نمای جدید باشد و P_2 از P_1 به صورت زیربدست آمده باشد : P_2 را مساوی $P_1 \cup \{S_2 \to S_1\}$ در نظر میگیریم. تا زمانی که P_2 شامل یک قاعده ی تولید به صورت $A \to \varepsilon$ برای بعضی $A \neq S_2$ ، P_2 را به صورت زیر تغییر می دهیم:

الف)قاعده ی تولید $A \to \varepsilon$ را حذف میکنیم.

ب)قاعده ی تولید $B \to \alpha A$ را تا زمانیکه چنین قاعده ی تولید جدیدی بتواند ساخته شود به P_2 اضافه میکنیم. αA رشته ای فرض میشود که یک A از آن حذف شده است و a سمت راست قاعده ی تولید $B \to \alpha$ فرض میشود که هنوز در P_2 است.اگر $\alpha A = \varepsilon$ و قاعده ی تولید $B \to \varepsilon$ بتواند به زودی از P_2 حذف شود آنگاه ، قاعده ی تولید دوباره داخل P_2 درج نمیشود.

افزایش قاعده ی تولید $B \to \alpha A$ زبان تولیدی P_2 را تغییر نمیدهد ، زیرا هر کاربرد قاعده ی تولید میتواند به وسیله ی جفت $B \to \alpha$ و $A \to \varepsilon$ از قاعده های تولید شبیه سازی شود.

به طور مشابه ، حذف قاعده ی تولید $A \to \varepsilon$ از P_2 در زبان تولید شده تاثیر میگذارد زیرا هر اشتقاق $C \Rightarrow \beta_1 A \beta_2 \Rightarrow^* \gamma_1 A \gamma_2 \Rightarrow \gamma_1\gamma_2$ که از $A \to \varepsilon$ استفاده میکند میتواند با یک اشتقاق معادل به صورت $C \Rightarrow \beta_1\beta_2 \Rightarrow^* \gamma_1\gamma_2$ جایگزین شود.

مثال۳-۳-۲ G_1 گرامر مستقل از متنی است که قواعد تولید آن در زیر آمده است.

$$
\begin{array}{lcl}
S_1 & \rightarrow & \epsilon \\
 & \rightarrow & S_1 aCC \\
C & \rightarrow & \epsilon \\
 & \rightarrow & Dab \\
D & \rightarrow & S_1
\end{array}
$$

ساختار برهان قضیه ی ۳-۳-۱ به گرامرهای معادل زیر اشاره میکند که G_2یک نوع گرامر نوع۲ است.

G'_1

Added $S_2 \rightarrow S_1$

$$
\begin{array}{lcl}
S_2 & \rightarrow & S_1 \\
S_1 & \rightarrow & \epsilon \\
 & \rightarrow & S_1 aCC \\
C & \rightarrow & \epsilon \\
 & \rightarrow & Dab \\
D & \rightarrow & S_1
\end{array}
$$

G'_2

Removed $S_1 \rightarrow \epsilon$

$$
\begin{array}{lcl}
S_2 & \rightarrow & S_1 \\
 & \rightarrow & \epsilon \\
S_1 & \rightarrow & S_1 aCC \\
 & \rightarrow & aCC \\
C & \rightarrow & \epsilon \\
 & \rightarrow & Dab \\
D & \rightarrow & S_1 \\
 & \rightarrow & \epsilon
\end{array}
$$

G'_3

Removed $C \rightarrow \epsilon$

$$
\begin{array}{lcl}
S_2 & \rightarrow & S_1 \\
 & \rightarrow & \epsilon \\
S_1 & \rightarrow & S_1 aCC \\
 & \rightarrow & S_1 aC \\
 & \rightarrow & S_1 a \\
 & \rightarrow & aCC \\
 & \rightarrow & aC \\
 & \rightarrow & a \\
C & \rightarrow & Dab \\
D & \rightarrow & S_1 \\
 & \rightarrow & \epsilon
\end{array}
$$

G_2

Removed $D \rightarrow \epsilon$

$$
\begin{array}{lcl}
S_2 & \rightarrow & S_1 \\
 & \rightarrow & \epsilon \\
S_1 & \rightarrow & S_1 aCC \\
 & \rightarrow & S_1 aC \\
 & \rightarrow & S_1 a \\
 & \rightarrow & aCC \\
 & \rightarrow & aC \\
 & \rightarrow & a \\
C & \rightarrow & Dab \\
 & \rightarrow & ab \\
D & \rightarrow & S_1
\end{array}
$$

از گرامرهای مستقل از متن تا اتوماتهای پشته ای

اتوماتهای پشته ای و برنامه های بازگشتی با حافظه ی متناهی ورودی هایشان را از چپ به راست پردازش میکنند.برای فعال کردن چنین ورودی هایی برای ردیابی (trace) اشتقاق از گرامر های مستقل از متن ، لم زیر خاصیتهای مشابه در گرامرهای مستقل از متن را بررسی میکند .

لم ۳-۳-۱.اگر یک نماد غیر پایانه ی A یک رشته ی ρ از نماد های پایانه را در گرامر مستقل از متن G تولید کند ، آنگاه ρ دارای یک چپ ترین اشتقاق از A در G است.

برهان: از برهان خلف استفاده میکنیم.میگوییم که در گرامر های مستقل از متن چپ ترین اشتقاق $\rho_1 \Rightarrow \rho_2 \Rightarrow \cdots \Rightarrow \rho_n$ با چپ ترین غیر پایانه ی در هر فرم جمله ای i $i=1,2,\ldots,n-1$ ρ... جایگزین میشود.

برهان بر این مشاهده تکیه میکند که ترتیب در نمادهای غیرپایانه جایگزین شده در شکلهای جمله ای برای اشتقاق ها در گرامر های مستقل از متن اهمیتی ندارد.هر نماد غیر پایانه در هر فرم جمله ای بدون هر همبستگی با محتوایش در فرم جمله ای بسط می یابد.

گرامر مستقل از متن G را در نظر میگیریم.با توجه به برهان فرض میشود که یک رشته ی P از نمادهای پایانه اشتقاقی با طول n از نماد غیر پایانه ی A دارد.به علاوه فرض میشود که P چپ ترین اشتقاق از A ندارد.

$A \Rightarrow \rho_1 \Rightarrow \rho_m \Rightarrow \rho_n = \not\rho \Rightarrow \ldots \Rightarrow \rho_m$ یک اشتقاق با طول n میباشد به صورتی که $\not\rho$ چپ ترین زیر اشتقاق است.علاوه بر این فرض میشود که m روی اشتقاق $A \Rightarrow \rho_1$ $A \Rightarrow^* \not\rho$ با طول n بیشینه میشود .با استفاده از این فرض که $\not\rho$ چپ ترین اشتقاق A نیست ، نتیجه میشود $m < n - 1$.

برای بعضی از رشته های w از نمادهای پایانه ، قاعده ی تولید $B \rightarrow \beta$ و $m < k < n$ و $\hat{\rho}_m, \ldots, \hat{\rho}_k$ اشتقاق مورد نظر $\hat{\rho}_{m+1} = w\rho_{k+1}$ $\hat{\rho}_k, \hat{\rho}, \ldots, \hat{\rho}_k = wB\hat{\rho}_m = w\,B\,{}_m\rho_k$ $\not\rho_{m+1}, \not\rho_m = w\,B$ را ارضا میکند آنگاه $A \Rightarrow \rho_1 \Rightarrow_{m-1} \not\rho_m \Rightarrow w\hat{\rho}\hat{\rho}_m$ $\not\rho$ نیز یک اشتقاق با طول n از A است. $\not\rho_n = \rho_k \Rightarrow_{k+1} \rho_k = \hat{\rho}\hat{\rho}w \Rightarrow_{m+1} \hat{\rho}\hat{\rho}w \Rightarrow$ در این اشتقاق جدید $A \Rightarrow \rho_1 \Rightarrow \ldots \Rightarrow \rho_m \Rightarrow \hat{\rho}w\hat{\rho}_m$ چپ ترین زیر اشتقاق با طول $m + 1$ است.نتیجتاً ، در وجود یک m ماکزیمم که در بالا به آن اشاره شد از فرضی که $\not\rho$ فقط غیر چپ ترین اشتقاق ها را از A دارد تناقض ایجاد میشود.

در نتیجه فرضی که $\not\rho$ چپ ترین اشتقاق از A نیست نیز به تناقض رسیده است.

برهان قضیه ی بالا نشان میدهد چگونه اتوماتای پشته ای میتوانند اشتقاق گرامر های مستقل از متن را ردیابی کنند.

قضیه ۳-۳-۲. هر زبان مستقل از متن توسط یک ماشین پشته ای پذیرفته میشود.

برهان: هر گرامر مستقل از متن, $G = <N,\Sigma,P, S>$, را در نظر میگیریم.بدون از دست

دادن عمومیت فرض میکنیم که Z_0 در $N \cup \Sigma$ نیست. $L(G)$ به وسیله ی ماشین پشته ای $M=<Q,\Sigma,N \cup \Sigma \cup \{Z_0\}, \delta, q_0, Z_0, \{q_f\}>$

که جدول انتقال δ شامل قوانین اشتقاق زیر است پذیرفته میشود.

الف)یک قاعده ی انتقال به صورت $(q_0,\varepsilon,\varepsilon,q_1,S)$.

ب)دنباله ای از قواعد انتقال برای هر $A \to \alpha$ در P .چنین دنباله ای در حالت q_1 شروع شده و در همان حالت پایان می یابد. نمادهای غیر پایانه A بر روی پشته با رشته ی α به صورت معکوس جایگزین میشود.

ج)قاعده ی انتقال به صورت $(q_1,a,a,q_1,\varepsilon)$ برای هر نماد پایانه a در الفبای Σ.

د)قاعده ی انتقال به صورت $(q_1, \varepsilon, Z_0,q_f ,Z_0)$.

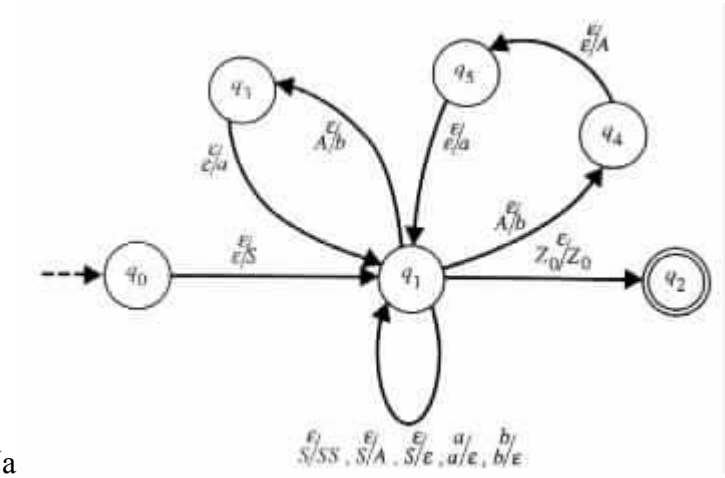
عملا ما میدانیم که با یک ورودی x ماشین پشته ای M به طور غیر قطعی یک چپ ترین اشتقاق در G که از S شروع شده و در x به پایان میرسد را ردیابی میکند.در هر مرحله از ردیابی بخشی از ورودی که با محتوای پشته به صورت معکوس ، خوانده شده است ، فرم جمله ای متناظر بامرحله اشتقاق اشتقاق ثبت میشود.

قاعده انتقال در (a) برای گذاشتن اولین فرم جمله ای S روی پشته استفاده میشود.قواعد انتقال در (b) برای جایگزینی چپ ترین نماد غیر پایانه در یک فرم جمله ای مورد نظر با سمت راست قاعده ی تولید مناسب استفاده میشود .قواعد انتقال (c) برای جفت کردن نمادهای پایانه ی راهنمای منطبق با آن در ورودی x گرفته شده استفاده میشود. هدف قاعده ی انتقال در(d) حرکت ماشین پشته ای در یک حالت پذیرش به محض رسیدن به پایان اشتقاق است.

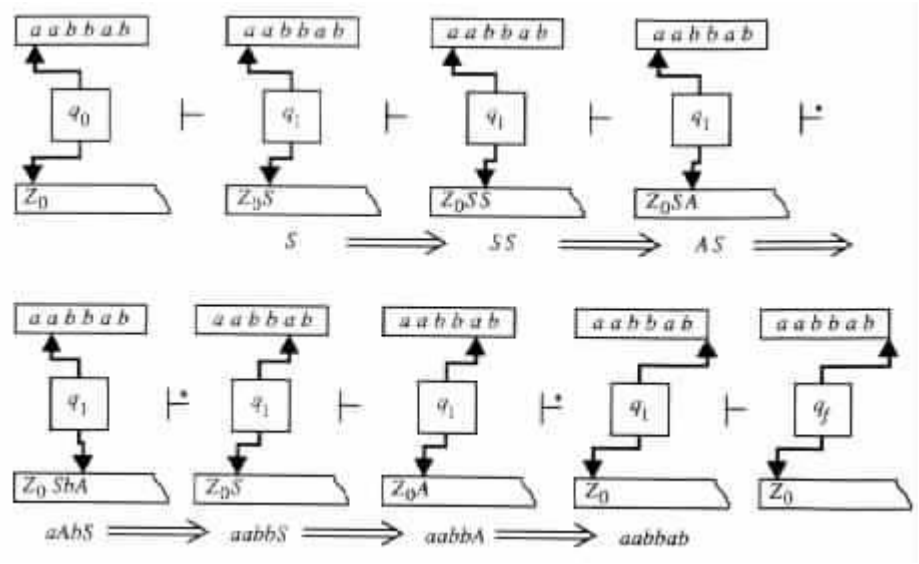
با استقرا روی n این نتیجه میتواند بدست آید کهx یک چپ ترین اشتقاق در G دارد ، اگر و تنها اگر M یک محاسبه ی پذیرش روی x داشته باشد که اشتقاق و محاسبه با $u_i v_i = x$ برای $1 \le i < n$ شکل زیر را داشته باشد.

S	$(q_0x, Z_0) \vdash (q_1x, Z_0S)$
$\Rightarrow u_1A_1\rho_1$	$\vdash^* (u_1q_1v_1, Z_0\rho_1^{rev}A_1)$
$\Rightarrow u_2A_2\rho_2$	$\vdash^* (u_2q_1v_2, Z_0\rho_2^{rev}A_2)$
$\Rightarrow \cdots$	$\vdash^* \cdots$
$\Rightarrow u_{n-1}A_{n-1}\rho_{n-1}$	$\vdash^* (u_{n-1}q_1v_{n-1}, Z_0\rho_{n-1}^{rev}A_{n-1})$
$\Rightarrow x$	$\vdash^* (xq_1, Z_0) \vdash (xq_f, Z_0)$

مثال۳-۳-۳. اگر G گرامر مستقل از متن مثال ۳-۳-۱ باشد آنگاه زبان L(G) با ماشین پشته ای M پذیرفته میشود که نمودار انتقال آن در شکل ۳-۳-۱ داده شده است.



((a



((b

شکل ۳-۳-۱.(a) ماشین پشته ای که زبان ساخته شده با گرامر مثال ۳-۳-۳ پذیرفته میشود.(b) یک چپ ترین اشتقاق در گرامر و محاسبه ی مطابق با آن در ماشین پشته ای.

aabbab چپ ترین اشتقاق S➔SS➔As➔aAbS➔aabbS➔aabbA➔aabbab را در گرامر G دارد.شکل(1-3-3)b پیکربندی های مطابق M در محاسباتش روی یک چنین ورودی را نشان میدهد.

از گرامرهای مستقل از متن تا برنا مه های بازگشتی با دامنه ی متناهی

با توجه به قضیه های ۳-۲-۱-و ۳-۳-۲ هر زبان مستقل از متن با یک برنامه ی با دامنه ی متناهی پذیرفته میشود.برای یک گرامر مستقل از متن گرفته شده به صورت G=<N,Σ,P,S> ، برنامه ی بازگشتی مستقل از متن T که L(G) را میپذیرد میتواند

به صورت زیر باشد.

T با ورودی x به طور غیر قطعی یک چپ ترین اشتقاق ردیابی شده است که در S شروع میشود.اگر چپ ترین اشتقاق رشته ی x را تولید کند آنگاه T ورودی اش را می پذیرد؛ در غیر این صورت T ورودی اش را رد میکند.

T یک رویه برای هر نماد غیر پایانه ی در N و یک رویه برای هر نماد پایانه در Σ دارد.رویه ای که مطابق با یک نماد غیر پایانه A است مسئول آغاز کردن یک ردیابی از یک چپ ترین زیر اشتقاقی است که در A آغاز میشود.رویه این کار را با انتخاب به صورت غیر قطعی یک قاعده ی تولید به صورت $A \rightarrow X_1 \ldots X_m$ انجام میدهد. و آنگاه رویه های مطابق با $X_1, \ldots, X_m$ را به ترتیب فراخونی میکند.از طرفی ، هر رویه ای که با یک نماد پایانه متناظر است مسئول خواندن یک نماد ورودی و تایید مساوی بودنش با نماد پایانه ی متناظر با آن میباشد.

هر کدام از رویه ها به محض اینکه مسئولیت واگذار شده را با موفقیت به پایان رساندند کنترل را به نقطه ی درخواست بر میگردانند.همچنین ، هر کدام از رویه ها به محض تعیین کردن اینکه مسئولیت واگذار شده نمیتواند انجام شود محاسبه را در یک وضعیت پذیرفته نشده به پایان میرسانند.

برنامه ی اصلی یک محاسبه را با درخواست رویه ای که با نماد آغازین S منطبق است شروع میکند. به محض بازگشت کنترل برنامه اصلی محاسبه را به پایان می رساند، پایان دریک وضعیت پذیرفته پیکربندی پذیرش است اگر و تنها اگر باقی مانده ورودی تهی باشد.برنامه بازگشتی با دامنه متناهی T در شکل ۳-۳-۲ داده شده است.

```
call S()
if eof then accept
reject
procedure A()        /* For each nonterminal symbol A.?*/
   do
:
   or                /* For each production rule of the form
 Xm.???                    */... X1 →                    A
```

```
 call Xm()...        call X1()
     return
   or
⋮
   until true
end
procedure a()              /* For each terminal symbol a.?*/
   read symbol
   if symbol = a then return
   reject
end
```

شکل ۳-۳-۲. طرحی از برنامه های بازگشتی با دامنه متناهی که گرامرهای مستقل از متن را شبیه سازی می کند.

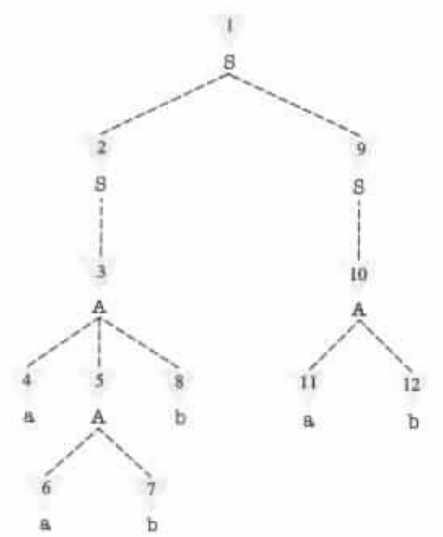
مثال ۳-۳-٤. اگر G گرامر مستقل از متن مثال ۳-۳-۱ باشد ، آنگاه L(G) با برنامه های بازگشتی با دامنه متناهی شکل ۳-۳-۳ پذیرفته می شود.

```
call S()
if eof then accept
reject
procedure S()
 SS  ?*/→    do                                  /* S
       call S() call S() return
 A   ?*/→    or                                  /* S
       call A() return
   ?*/ε →    or                                  /* S
       return
   until true
end
procedure A()
 aAb?*/→     do                                  /* A
       call a() call A() call b() return
 ab  ?*/→    or                                  /* A
       call a() call b() return
   until true
end
procedure a()
```

```
   read symbol
   if symbol = a then return
   reject
end
procedure b()
   read symbol
   if symbol = b then return
   reject
end
```

شکل ۳-۳-۳. یک برنامه بازگشتی با دامنه متناهی برای گرامر مثال ۳-۳-۱.

روی ورودی aabab برنامه بازگشتی بادامنه متناهی اشتقاق S ⇒SS ⇒AS ⇒aAbS را با فراخوانی رویه هایش به ترتیبی که درشکل ۳-۳-٤ نشان داده شده است دنبال می کند.



شکل ۳-۳-٤.فرا خوانی های رویه هایی که برنامه شکل ۳-۳-۳ برای ورودی aabbab انجام میدهد.

از برنامه های بازگشتی با دامنه متناهی تا گرامر های مستقل از متن

یک نگاه دقیق به برهان قضیه ی ۲-۳-۲ نشان می دهد که چگونه یک برنامه با حافظه ی متناهی مفروض مانند P می تواند به وسیله گرامر نوع سوم, $G = \Sigma,P, S>$ $<N,$, شبیه سازی شود.

گرامر از نمادهای غیر پایانه اش برای ثبت حالات P استفاده می کند. هر قاعده تولید به صورت $A \rightarrow aB$ در گرامر استفاده می شود برای شبیه سازی یک زیر محاسبه از P که از حالت ثبت شده توسط A آغاز شده و درحالت ثبت شده توسط B به پایان می رسد و یک نماد ورودی a را می خواند.

به هر حال هر قاعده تولید به صورت $A \rightarrow a$ در گرامر برای شبیه سازی زیر محاسبه ای از P که از حالتی که توسط A ثبت شده است شروع می شود و دریک حالت پذیرش به پایان می رسد استفاده می شود و یک نماد ورودی a را می خواند. نماد شروع S از G به عنوان حالت اولیه P استفاده می شود . قاعده تولید $S \rightarrow \varepsilon$ برای شبیه سازی یک محاسبه پذیرش از P که ازهیچ مقدار ورودی خوانده نشده ، استفاده می شود.

برهان قضیه زیر بر همین مطلب تکیه می کند

قضیه ۳-۳-۳. هر زبان که با یک برنامه بازگشتی بادامنه متناهی پذیرفته می شود یک زبان مستقل از متن است.

برهان. هر برنامه بازگشتی با دامنه متناهی P را در نظر می گیریم. بدون از دست دادن عمومیت، می توان فرض کرد که برنامه دستورات نوشتن ندارد. زبانی که توسط P پذیزفته می شود ، می تواند توسط یک گرامر مستقل از متن G تولید شود که محاسبات P را شبیه سازی می کند. نمادهای غیر پایانه P برای نشان دادن حالتهای شروع و پایان زیر محاسبات P که شبیه سازی شده اند، استفاده می شود و قواعد تولید G برای شبیه سازی انتقال بین حالتهای P استفاده می شوذ.

نمادهای غیر پایانه ی G

نمادهای غیر پایانه ی G شامل موارد زیر است :

الف) یک نماد غیر پایانه A_q ، برای هر حالت q از p . هر نماد غیر پایانه ی این چنینی A_q برای نشان دادن یک زیر محاسبه شبیه سازی شده از P که درحالت q شروع می شود و دریک حالت پذیرش خاتمه می یابد، استفاده می شود. علاوه بر این ، هر اجرا ازیک دستور بازگشت در زیر محاسبه باید برای یک فراخوانی که قبلا درطول زیر محاسبه ساخته شده است باشد.

نماد شروع G نماد غیر پایانه ی A_{q0} است که مطابق با حالت اولیه q_0 از p می باشد.

ب) یک نماد غیر پایانه $A_{q,p}$ ، برای هر جفت p,q مطابق است با قطعه دستوری که در همان رویه ی P است. هرنماد غیر پایانه ی اینچنین $A_{q,p}$ برای نشان دادن یک زیر محاسبه شبیه سازی شده که در حالت q آغاز شده و درحالت p به پایان می رسد ، به کار می رود. در زیر محاسبه تعداد اجراهای دستورات بازگشت با تعداد اجراهای دستورات فراخوانی برابر است.

علاوه بر این، هر اجرا از یک دستور بازگشت در زیر محاسبه باید برای یک فراخوانی که قبلا در طول زیر محاسبه ساخته شده است، باشد.

قواعد تولید G

قواعد تولید G شامل موارد زیر است:

الف) یک قاعده تولید به صورت $A_q \rightarrow \alpha A_r$ و یک قاعده تولید به صورت $A_q \rightarrow \alpha A_{r,p}$ برای هر $A_{q,p}$ p.q,r و α که شرایط زیر را ارضا می کند. قطعه دستورات مطابق باحالت q نه یک دستور فراخوانی است و نه یک دستور بازگشت و اجرایش می تواند برنامه را از حالت q به حالت r در طول خواندن α ببرد.

قاعده تولید به صورت $A_q \rightarrow \alpha A_r$ ،هدف رسیدن به حالت p از حالت q را باهدف بدست آوردن حالت p از حالت r جا به جا می کند.

قاعده تولید $A_{q,p} \rightarrow \alpha A_{r,p}$ ، هدف رسیدن به حالت p از حالت q را با هدف رسیدن به حالت p ازحالت r ، جا به جا می کند.

ب) یک قاعده تولید به صورت $A_q \rightarrow A_{r,p}$ برای هرحالت q مطابق با یک دستور if *eof* then accept می باشد.

ج) یک قاعده تولید به صورت $A_q \to A_r$ ، برای هرحالت q مطابق با یک دستور فراخوانی است ، جایی که r حالت بدست آمده از q است. چنین قاعده تولیدی یک اجرا از یک فراخوانی را که با یک اجرا از بازگشت همخوان نیست، شبیه سازی میکند.

د) یک قاعده تولید به صورت $A_q \to A_{r,s}A_t$ و یک قاعده تولید به صورت $A_{r,s}A_{t,p} \to A_{q,p}$ برای هر q, r, s, t و p به گونه که ای که شرایط زیر را داشته باشد :

۱- حالت q متناظر است با یک دستور فراخوانی که اجرایش در یک حالت سبب وارد شدن به حالت r می شود.

۲- حالت S متناظر است با یک دستور بازگشت در رویه ی فراخوانده شده، و اجرای دستور بازگشت در هر حالت برنامه را به حالت t که با r همساز است ، می برد.

زیر محاسبه ای که در حالت q آغاز میشود ، به دو زیر محاسبه تجزیه می گردد. یکی برای اجرا شدن توسط یک رویه فراخوانده شده که درحالت r آغاز شده و در حالت s خاتمه می یابد: دیگری از لحظه ای به دست می آید که کنترل از رویه فراخوانده شده که در حالت t شروع می شود، بازگردد.

ه) یک قاعده تولید به صورت $A_{q,q} \to \varepsilon$ برای هر حالت q که با یک دستور بازگشت متناظر است.

هر کدام از قواعد تولید بالا برای پایان بردن یک شبیه سازی موفق از یک زیر محاسبه با یک رویه ی فراخوانده شده ، به کار برده می شود.

L(P) شامل L(G) است

یک برهان با استقرا می تواند برای نشان دادن اینکه ساختمان بالا بیان می کند که L(G) = L(P) ، استفاده می شود. برای نشان دادن اینکه L(P) شامل L(G) است ، کافی است نشان داده شود دوشرط زیر برای هر رشته a از نمادهای پایانه برقرار است.

الف) اگر $A_q \Rightarrow^* a$ در G باشد آنگاه P می تواند از حالت q به یک حالت پذیرفته شده در طول خواندن a برسد و در هر پیشوند از دنباله های زیر اجرا باید حداقل به تعداد اجرای دستورات فراخوانی ، اجرای دستورات بازگشت وجود داشته باشد.

ب) اگر $A_{q,p} \Rightarrow^* a$ در G باشد ، آنگاه P می تواند حالت p را از حالت q در طول

خواندن a به دست آورد. در دنباله زیر اجرا تعداد اجراهای دستورات بازگشت باید با تعداد اجراهای دستورات فراخوانی مساوی باشد و هر پیشوند از دنباله زیر اجرا باید حداقل به تعداد اجرای دستورات فراخوانی، اجرای دستورات بازگشت داشته باشد.

.برهان می تواند با استقرا روی تعداد گامهای i دراشتقاقها به نتیجه برسد. برای i=1 تنها اشتقاق ممکن آنست که یکی از دوفرم $A_q \Rightarrow$ یا $A_{q,p} \Rightarrow$ را داشته باشد. در اولین مورد ، q متناظر است با یک دستور پذیرش و در دومین مورد، p متناظر است با یک دستور بازگشت. در هر دونمونه دنباله زیر اجرای برنامه خالی است.

برای i>۱ اشتقاقها باید یکی از دو فرم زیر را داشته باشند

-۱

$$A_q \Rightarrow \alpha_1 A_r \Rightarrow^* \alpha_1\alpha_2 = \alpha, \text{ or } A_{q,p} \Rightarrow \alpha_1 A_{r,p} {}^* \alpha_1\alpha_2 = \alpha$$

در هریک از مواد با تعریف $A_q \Rightarrow \alpha_1 A_r$ و $A_{q,p} \Rightarrow \alpha_1 A_{r,p}$ با دنباله های زیر اجرایی که درحالت p شروع می شود و در حالت r به پایان می رسد، از ورودی a استفاده می کند و هیچ دستور فراخوانی یا بازگشتی اجرا نمی شود. به هرحال ، با توجه به فرض استقرا $A_r \Rightarrow {}^*\alpha_2$ و $A_{r,p} \Rightarrow {}^*\alpha_2$ منطبق است با دنباله های زیر اجرایی که صفات مورد نظر را دارند. در نتیجه $A_q \Rightarrow {}^*\alpha$ و $A_{q,p} \Rightarrow {}^*\alpha$ با ردیف های زیر اجرایی که خواص مورد نظر را دارند نیز مطابق هستند.

۲-

$$A_q \Rightarrow A_{r,s}A_t \Rightarrow^* \alpha_1\alpha_2, \text{ or } A_{q,p} \Rightarrow A_{r,s}A_{t,p} {}^* \alpha_1\alpha_2, \text{ where } A_{r,s} \Rightarrow^* \alpha_1$$

در هریک از موارد با تعریف q مطابق با یک دستور فراخوان، r حالتی است که p از حالت q بدست می آورد، s متناظر است بایک دستور بازگشت و t حالتی است که p از حالت S به آن می رسد. به هرحال با توجه به فرض استقرا $A_{r,s} \Rightarrow^* \alpha_1$ و $A_t \Rightarrow {}^*\alpha_2$ و $A_{t,p} \Rightarrow {}^*\alpha_2$ متناظر است با دنباله های زیر اجرایی که صفات مورد نظر را دارند. در نتیجه $A_q \Rightarrow_2 {}^*\alpha_1\alpha$ و $A_{q,p} \Rightarrow^* \alpha_1\alpha_2$ با دنباله های زیر اجرایی که خواص مورد نظر را دارند نیز متناظر هستند.

L(G) شامل L(P) می باشد

برای نشان دادن اینکه L(G) شامل L(P) می باشد کافی است نشان دهیم که هریک

از شروط زیر برای هر دنباله از زیر اجرایی که a را می خواند و درحالت q شروع می شود و درحالت p پایان می یابد و حداقل به تعداد اجراهای دستورات بازگشت دارای دستورات فراخوانی برای هریک پیشوندهاست ، برقراراست.

الف) اگر p با یک حالت پذیرش مطابق باشد، آنگاه G یک اشتقاق به صورت $\Rightarrow^*$ دارد A_q.

ب) اگر p مطابق با یک دستور بازگشت باشد و دنباله اجراها به همان تعدادی که اجرای اجرای دستورات فراخوانی دارد، اجرای دستورات بازگشت نیز داشته باشد آنگاه G یک اشتقاق به صورت $\Rightarrow^*$ $A_{q,p}$ دارد.

برهان متکی است بر استقرا روی تعداد حرکتهای i در دنباله های زیراجرا، برای i=0 دنباله های زیر اجرا هیچ ورودی استفاده نمی کند و برای آنها G اشتقاق های مطابق $A_q \Rightarrow$ و $A_{q,p} \Rightarrow$ را به ترتیب دارد.

برای i>0 هریک از موارد زیر باید برقرار باشد:

الف) q با هیچ دستور فراخوانی متناظر نیست یا q متناظر است با یک دستور فراخوانی که در ردیف زیر اجرا بایک دستور بازگشت جفت نیست. دراین مورد، با اجرای یک تک قطعه دستور، دنباله های زیراجرا در سوال وارد برخی حالت r از حالت های q می شود، زمانی که برخی ورودی های a را می گیرد.

درنتیجه برحسب تعریف، گرامر G یک قاعده تولید به صورت $A_q \rightarrow a A_r$ دارد اگر p یک حالت پذیرفته شده باشد ویک قاعده تولید به صورت $A_{q,p} \rightarrow a A_{r,p}$ دارد اگر p با یک دستور بازگشت متناظر باشد.

به هرحال، با توجه به فرض استقرا i-1 حرکتی که درحالت r شروع می شود با یک اشتقاق به صورت $A_r \Rightarrow^* x_2$ در G متناظر است اگر p یک حالت پذیرش باشد و به صورت $A_{r,p} \Rightarrow^* x_2$ است اگر p با یک دستور بازگشت متناظر باشد. x_2 به صورتی فرض می شود که $x = x_1 x_2$ را ارضا کند.

ب) q بایک دستور فراخوانی متناظر است که در دنباله زیراجرا با یک دستور بازگشت جفت باشد. دراین مورد دنباله زیر اجرا از حالت q وارد بعضی از حالت های

r می شود با اجرا کردن دستور فراخوانی که باحالت q متناظر است ، به علاوه دنباله زیر اجرا با یک اجرا از یک دستور بازگشتی متناظر است که دنباله زیراجرا را از بعضی از حالت های s به برخی از حالت های t می برد.

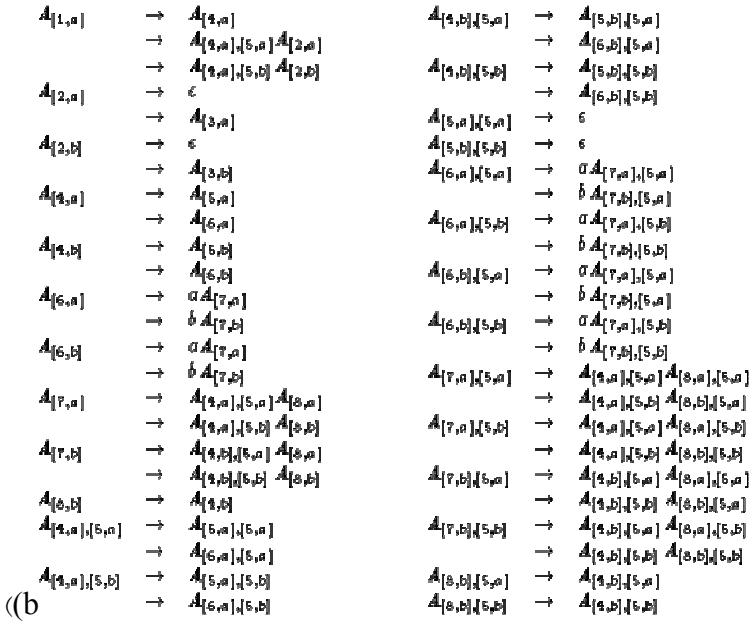
در نتیجه برحسب تعریف ، گرامر G یک قاعده تولید به فرم $A_q \rightarrow A_{r,s}A_t$ دارد اگر p یک حالت پذیرش باشد و یک قاعده تولید به فرم $A_{q,p} \rightarrow A_{r,s}A_{t,p}$ دارد اگر p با یک دستور بازگشت متناظر باشد.

به هرحال ، با توجه به فرض استقرا ، گرامر G اشتقاقی به صورت $A_{r,s} \Rightarrow^* a_1$ برای ورودی a دارد که دنباله زیر اجرا بین حالت t و p استفاده میشود. به علاوه G هریک از اشتقاقهای به فرم $A_t \Rightarrow a_2$ یا اشتقاق به فرم $A_{t,p} \Rightarrow a_2$ را به ترتیب دارد، برای ورودی a_2ای که دنباله زیر اجرا بین حالت t و p استفاده می شود بستگی به این دارد که q یک حالت پذیرش باشد یا نه.

مثال ۳-۳-۵. برنامه بازگشتی بادامنه متناهی p در شکل ۳-۳-۵ (a) را با {a, b} به عنوان دامنه ی متغیرها و a به عنوان مقدار اولیه در نظر بگیرید.

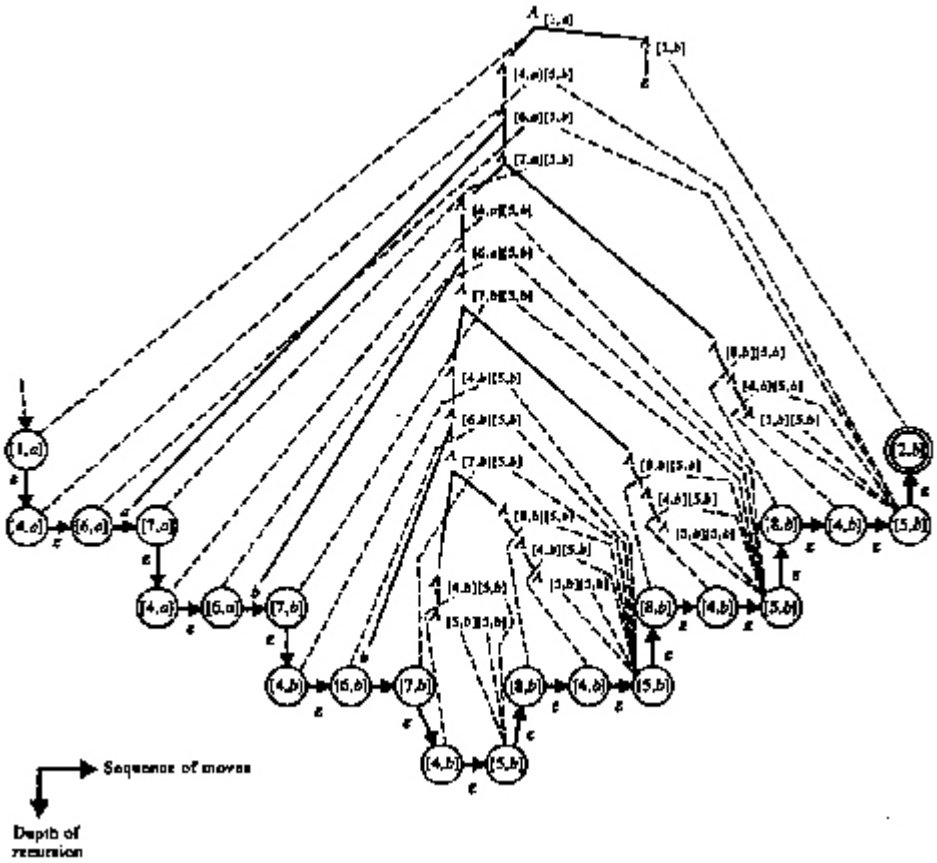
```
Call f(x)                    /* I1/*?
if eof then accept           /* I2/*?
reject                       /* I3/*?
procedure f(x(
   do                        /* I4/*?
      return                 /* I5/*?
   or
      read x                 /* I6/*?
      call f(x)              /* I7/*?
   until x = a               /* I8/*?
end
```

(a)



شکل ۳-۳-۵. گرامر در (b) زبانی را تولید می کند که توسط برنامه ی (a) پذیرفته می شود.

L(P) توسط گرامر G تولید می شود، که قواعد تولید شکل ۳-۳-۵(b) را دارد. [i, x]حالتی از p را مشخص می کنند که با قطعه دستور I_i و مقدار X در x مطابق است. درخت اشتقاق برای رشته abb درگرامر G و انتقالات بین حالت های برنامه p روی ورودی "a, b, b" در شکل ۳-۳-۶ نشان داده شده است ،نماد $A_{[1,a]}$ نشان میدهد که محاسبه ی p در حالت [a ,۱] شروع شده و در یک حالت پذیرش به پایان می رسد، قاعده تولید $A_{[1,a]} \rightarrow A_{[4,a][5,b]}A_{[2,b]}$ متناظر است با یک فراخوانی به f که مقدار b را باز میگرداند .



شکل ۳-۳-٦. یک تطابق بین یک درخت اشتقاق و یک محسبه از برنامه بازگشتی باحافظه متناهی

گرامرهای مستقل ازمتن شبیه اتوماتای پشته ای نیستند، به صورتی که گرامر های نوع سوم با اتوماتای حالت متناهی شبیه هستند، این تفاوت ناشی از آن است که اشتقاقهای گرامرهای مستقل از متن به صورت طبیعی بازگشتی هستند درحالی که محاسبات اتوماتای پشته ای تکراری هستند.

در نتیجه ، برخی از زبانهای مستقل ازمتن به وسیله ی گرامرهای مستقل از متن راحت تر مشخص می شوند و برخی زبانهای مستقل از متن به وسیله اتوماتای پشته ای می توانند راحت تر مشخص شوند.

۳-٤- محدودیت های برنامه های بازگشتی با دامنه متناهی

مطالعه محدویت های برنامه های با حافظه متناهی در بخش ۲-٤ بر مشاهده زیر متکی است : یک زیر محاسبه ی پذیرش از یک برنامه با حافظه متناهی اگر زیر محاسبه در یک حالت یکسان شروع شده و خاتمه یابد، می تواند برای بدست آوردن محاسبه ی پذیرش جدید پمپ شوند.(pumped) برای برنامه های بازگشتی با دامنه متناهی مشابه ، اما با پیچیدگی بیشتر، شرایط نیاز به اجازه ی فشرده کردن زیر محاسبات دارند.

قضیه فشار برای زبانهای مستقل از متن

برهان قضیه ی زیر از خلاصه ی گرامر ها ی مستقل از متن برای مهیا کردن شرایط زیرمحاسبات برنامه های بازگشتی با دامنه ی متناهی که میتوانند پمپ شونداستفاده میکند.قضیه ی مطابق برای موارد از بین رفته ی برنامه های با حافظه ی متناهی با انتخاب $v = u = \varepsilon$ به کار میرود.

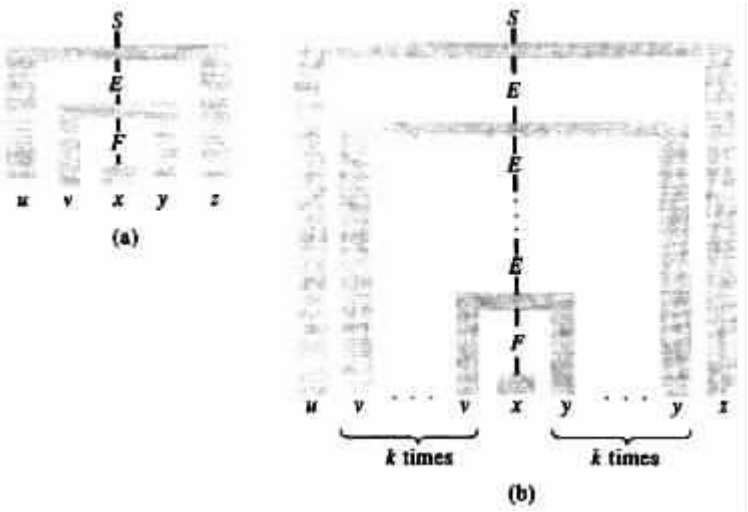
قضیه ۳-٤-۱. (قضیه فشار برای زبانهای مستقل از متن) هر زبان مستقل از متن L یک عدد صحیح مثبت m با خاصیت زیر دارد. اگر w در L باشد و $|w| \geq m$ ، آنگاه w می تواند به صورت uvxyz نوشته شود، جایی که به ازای هر $K \geq 0$ ، uv^kxy^kz در L باشد. علاوه بر این $|vxy| \leq m$ و $|vy| > 0$.

اثبات. , $G = <N,\Sigma,P, S>$ یک گرامر مستقل از متن است. از t برای نشان دادن تعداد نمادها در طولانی ترین سمت راست قواعد تولید G استفاده می کنیم. بدون از دست دادن عمومیت، فرض می کنیم که $t \geq 2$ باشد. از |N| برای نشان دادن تعداد نمادهای غیر پایانی در N استفاده میشود. M را مساوی با $t^{|N|+1}$ انتخاب می کنیم.

هر w در L(G) را چنان که $|w| \geq m$ باشد ، بررسی می کنیم. T را نشان دهنده ی یک درخت اشتقاق برای w که کم ترین گره را دارد، در نظر می گیریم. π را طولانی

ترین مسیر از ریشه به یک برگ در T در نظر میگیریم. N نشان دهنده ی تعداد گره ها در π می باشد.

تعداد برگ ها در T حداکثر برابر است با t^{n-1} . پس $t^{n-1} \geq |w|$ و $|w| \geq m = t^{|N|+1}$ | نشان می دهد که $n \geq |N|+2$. مسیر π باید دو گره معادل نمادهای غیر پایانی داشته باشد که به آنها E و F گفته می شود و برابر هستند. در نتیجه، w میتواند به صورت uvxyz نوشته شود جایی که vxy و x رشته ای معادل با برگ های زیر درخت هایی از T هستند که ریشه های آنها به ترتیب E و F می باشد.(شکل ۳-٤-۱ (a) را ببینید.)



شکل ۳-٤-۱. (a) : درخت اشتقاق T با E=F : b) . درخت اشتقاق T_k .

$$\begin{aligned} S &\rightarrow AA \\ &\rightarrow ab \\ A &\rightarrow SS \\ &\rightarrow a \end{aligned}$$

T_k را اصلاح شده ی درخت اشتقاق T در نظر می گیریم چنان که زیر درخت E ، k بار پمپ شود(pumped) ، در حالی که زیر درخت F را شامل نمی شود (شکل ۳-٤-۱(b) را ببینید.). T_k یک درخت اشتقاق برای هر $k \geq 0$ در G نیز می باشد. نتیجه می شود که uv^kxy^kz ، که معادل با برگ های T_k است ، نیز برای هر ≥ 0

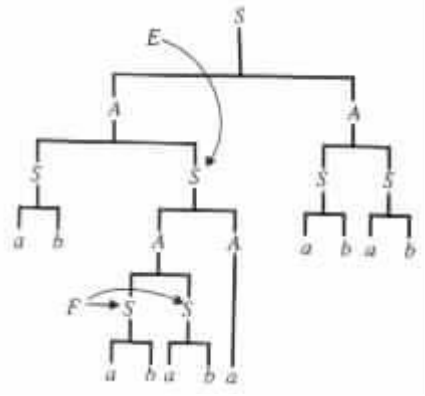
kدر L(G) می باشد.

یک انتخاب E و F از انتهای $|N|+1$ نماد غیر پایانی در مسیر π ، نشان می دهد که $|vxy| \geq t^{|N|+1} = m$ ، زیرا هر مسیر از E به یک برگ، حداکثر شامل $|N|+2$ گره می باشد. در هرحال، $|vy| \neq 0$ ، زیرا در غیر این صورت T_0 نیز یک درخت اشتقاق برای w است و این فرض اینکه T کوچکترین درخت اشتقاق برای w است را نقض می کند.

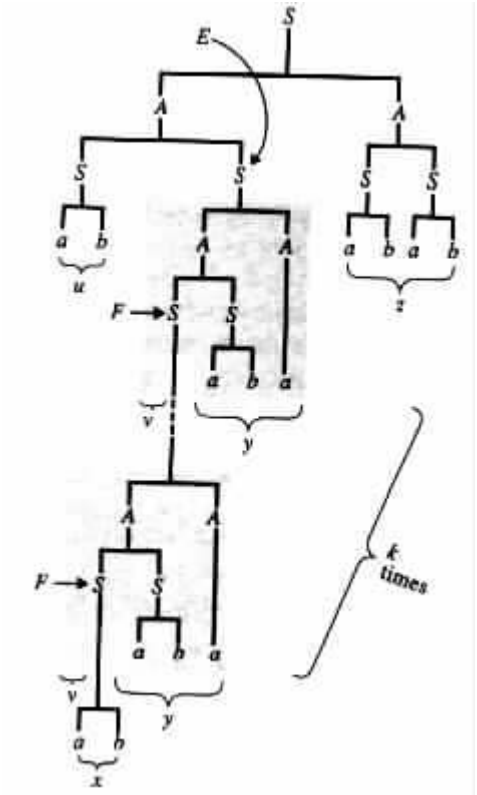
مثال ۳-٤-۱. , $G = <N,\Sigma,P, S>$, یک گرامر مستقل از متن است که قواعد تولید آن در زیر آمده است.

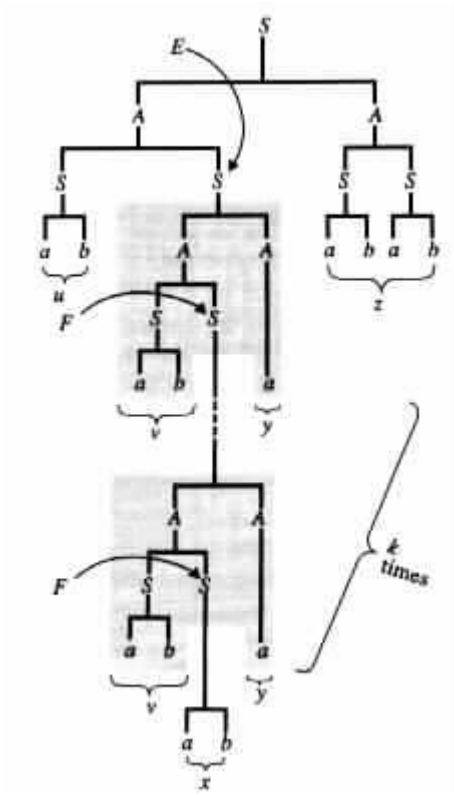
$$\begin{aligned} S &\rightarrow AA \\ &\rightarrow ab \\ A &\rightarrow SS \\ &\rightarrow a \end{aligned}$$

برای G اصطلاحات برهان قضیه ی قبل را به کار می بریم : $t = 2, |N| = 2$ و $m =$ 8. رشته ی $w = (ab)^3a(ab)^2$ درخت اشتقاقی دارد که در شکل ۳-٤-۲(a) داده شده است.



(a)





(c) (b)

شکل ۳-٤-۲. (a) ـ یک درخت اشتقاق T برای $(ab)^3a(ab)^2$. (b) ـ یک درخت اشتقاق T_k برای $(ab)^2(aba)^k(ab)^2$. (c) ـ یک درخت اشتقاق T_k برای $ab(ab)^kaba^k(ab)^2$.

طولانی ترین مسیر در درخت از ریشه تا برگ شامل شش گره می باشد.

w دو تجزیه دارد که شرایط برهان قضیه ی فشار را ارضا می کند. یکی به فرم $u = ab, v = ab, x = ab, y = a, z = abab$ و دیگری به فرم $u = ab, v =, x = ab, y = aba, z = abab$.

$(ab)^2(aba)^k(ab)^2$ و $ab(ab)^kaba^k(ab)^2$ رشته های جدید برای هر $k \geq 0$ در

زبان هستند که برهان برای w بوسیله فشار به کار میبرد. شکل ۳-٤-۲(b) و ۳-٤-۲(c)، به ترتیب درخت اشتقاق T_k را برای این رشته ها نشان می دهد.

موارد استفاده ی قضیه فشار

قضیه فشار برای زبان های مستقل از متن می تواند برای نشان دادن اینکه یک زبان مستقل ار متن نیست، استفاده شود. این روش شبیه به استفاده قضیه فشار برای زبان های منظم است که نشان می داد یک زبان منظم نیست.

مثال ۳-٤-۲. L زبان $\{ a^n b^n c^n \mid n \geq 0 \}$ می باشد. برای نشان دادن اینکه L یک زبان مستقل از متن نیست، برعکس فرض می کنیم L مستقل از متن است. بررسی می کنیم انتخاب $w = a^m b^m c^m$ جایی که m ثابت اشاره شده توسط قضیه فشار برای L می باشد.

طبق قضیه ، $a^m b^m c^m$ می تواند به صورت uvxyz نوشته شود، جایی که $|vy|, m$ $|vxy| \leq 0$ و تجزیه شرایط زیر را ارضا می کند:

الف) vy حاوی a ها و b ها است اما نه cها.

ب) vy حاوی a ها و c ها است اما نه bها.

ج) vy حاوی cها و b ها است اما نه aها.

به علاوه، طبق قضیه فشار $uv^k xy^k z$ نیز برای هر $k \geq 0$ در L می باشد. بهر حال ، برای (a) انتخاب k=0 نشان می دهد که $uv^0 xy^0 z$ در L نیست به دلیل تعداد زیاد c ها. به همان صورت برای (b) انتخاب k=0 نشان می دهد که $uv^0 xy^0 z$ در L نیست به دلیل تعداد زیاد b ها.و برای (c) انتخاب k=0 نشان می دهد که $uv^0 xy^0 z$ در L نیست به دلیل تعداد زیاد a ها.

از این رو، قضیه فشار برای $a^m b^m c^m$ صدق نمی کند، برای L نیز صدق نمیکند. بنابراین فرض اینکه L یک زبان مستقل از متن است اشتباه می باشد.

در چنین موردی از قضیه فشار برای زبان های منظم، وقتی که سعی می شود نشان داده شود که زبان، مستقل از متن نیست، انتخاب رشته ی w اهمیت فوق العاده ای

دارد.

مثال ۳-٤-۳. زبان $L = \{ww \mid w \text{ is in } \{a, b\}^*\}$ را در نظر بگیرید. برای نشان دادن اینکه L یک زبان مستقل از متن نیست، خلاف آن فرض می شود.m را ثابت اشاره شده توسط قضیه فشار برای L در نظر میگیریم .

برای انتخاب $w = a^m b^m a^m b^m$ قضیه فشار یک تجزیه uvxyz در نظر می گیرد چنان که $|vxy| \leq m, |vy| > 0$. برای انتخاب $uv^0xy^0z = uxz = a^i b^j a^s b^t$ با هر $i \neq s$ و $j \neq t$. در هر نمونه uxz در L نیست. در نتیجه L نمی تواند مستقل از متن باشد. از طرف دیگر، برای انتخاب $w = a^m b a^m b a$ ، یک تجزیه uvxyz که $|vy| \leq m$, $|vxy| > 0$ را ارضا کند، ممکن است به فرم $v = y = a^j$ با b درx باشد. یک تجزیه ی $uv^k xy^k z = a^{m+(k-1)j} b a^{m+(k-1)j} b$ نیز برای همه ی $k \geq 0$ در L می باشد. در نتیجه انتخاب اخیر برای w تناقض مورد نظر را به وجود نمی آورد.

تعمیم قضیه فشار

قضیه فشار برای زبان های مستقل از متن می تواند به ارتباطی که توسط مبدل های پشته ای قابل محاسبه است، تعمیم داده شود.از سوی دیگر تعمیم قضیه فشار ، می تواند برای تعیین کردن روابطی که توسط مبدل های پشته ای قابل محاسبه نیستند ، استفاده شود.

قضیه ۳-٤-۲. برای هر رابطه ی R که توسط مبدل پشته ای قابل محاسبه است ، یک ثابت m وجود دارد طوری که برای هر (w_1, w_2) در R شرایط زیر برقرار است. اگر $|w_1| + |w_2| \geq m$ ، آنگاه w_1 می تواند به صورت $u_1v_1x_1y_1z_1$ نوشته شود و w_2 می تواند به صورت $u_2v_2x_2y_2z_2$ نوشته شود، جایی که $(u_1v_1^k x_1 y_1^k z_1, u_2 v_2^k x_2 y_2^k z_2)$ نیز برای هر $k \geq 0$ در R باشد. علاوه بر این $|v_1x_1y_1| + |v_2x_2y_2| \leq m$ و $|v_1y_1| + |v_2y_2| > 0$.

اثبات. هر مبدل پشته ای M_1 را بررسی می کنیم . M_2 را اتوماتای پشته ای بدست آمده از M_1 توسط جابه جا کردن هر قانون انتقال از (q,a,β,p,γ) , با یک قانون

انتقال از(β , p, γ) ,،اگر نامساوی $[\alpha, \beta] \neq [\varepsilon, \varepsilon]$ برقرار باشد، و با یک قانون انتقال از(β , p, γ) , ، اگر برابری $[\alpha, \beta] = [\varepsilon, \varepsilon]$ برقرار باشد، در نظر میگیریم.

h_1 و h_2را توابع تصویر تعریف شده در روش زیر در نظر میگیریم

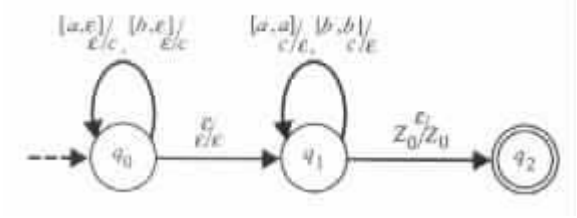
$h_1(\varepsilon) = h_2(\varepsilon) = \varepsilon$, $h_1([\alpha, \beta]) = \alpha$, $h_2([\alpha, \beta]) = \beta$, $h_1([\alpha, \beta]w) = h_1([\alpha, \beta])h_1(w)$, and $h_2([\alpha, \beta]w) = h_2([\alpha h_2(w)$

طبق ساختار، M_2 در ورودی های خود ، ورودی ها و خروجی های M_1 را کدگذاری می کند. h_1 و h_2 به ترتیب این ورودی و خروجی های کدگذاری شده را معین میکنند. در نتیجه (w_1, w_2) در $R(M_1)$ هستند اگر برای w هایی که $h_1(w) = w_1$, $h_2(w) = w_2$، w در $L(M_2)$ باشد. از m برای نشان دادن ثابت اشاره شده توسط قضیه فشار برای زبان های مستقل از متن استفاده می کنیم برای $L(M_2)$ و انتخاب m $= 2m'$.

هر (w_1, w_2) در رابطه $R(M_1)$ بررسی می شود طوری که $|w_1| + |w_2| \geq m$.

سپس چند w در زبان $L(M_2)$ داریم که $h_1(w) = w_1, h_2(w) = w_2$ و $\geq m/2 = m'$. بوسیله فشار برای زبان های مستقل از متن w می تواند به صورت uvxyz $|w|$ نوشته شود، جایی که $|vy| > 0$ m' , $|vxy| \leq m'$ و uv^kxy^kz برای هر $k \geq 0$ در $L(M_2)$ باشد. نتیجه برقرار می شود اگر انتخاب ها $u_1 = h_1(u), u_2 = h_2(u), v_1 = h_1(v), v_2 = h_2(v), x_1 = h_1(x), x_2 = h_2(x), y_1 = h_1(y), y_2 = h_2(y), z_1 = h_1(z)k_2(z)z_2=$ باشد.

مثال ۳-٤-۲. M_1 مبدل پشته ای است که نمودار انتقال آن در شکل ۳-۲-۳ داده شده است. اصطلاحات برهان قضیه ۳-٤-۲ را به کار می بریم، M_2 اتوماتای پشته ای است که نمودار انتقال آن در شکل ۳-٤-۳ داده شده است.



شکل ۳-۲-۳. یک اتوماتای پشته ای که مبدل پشته ای شکل ۳-۲-۳ را کدگذاری می کند.

محاسبات M_1 روی ورودی aabbaa ، خروجی baaرا داده است. محاسبات M_1 روی ورودی aabbaa معادل است با محاسبات M_2 روی ورودی [a,ε][a,ε][b,ε][b,b][a,a][a,a].

۳-۵- خاصیت بستار برای برنامه های بازگشتی با دامنه متناهی

از M تا M_{eof} ،حذف حالات مرکب ، تغییری در M_{eof} ،یک اتوماتای پشته ای که متمم $L(M)$ را می پذیرد.

با برهانی شبیه به آنچه برای قضیه ۲-۵-۱ به کار رفت، کلاس روابط محاسبه پذیر توسط مبدل های پشته ای و درنتیجه کلاس زبانهای مستقل از متن تحت اجتماع بسته هستند. به هرحال این کلاسها تحت اشتراک و متمم بسته نیستند. برای مثال ، زبان $\{ a^n b^n c^n \mid n \geq 0 \}$ که مستقل از متن نیست اشتراکی از زبانهای مستقل از متن $\{ a^i b^i c^j \mid i, j \geq 0 \}$ و $\{ a^i b^j c^j \mid i, j \geq 0 \}$ است .

به طور مشابه کلاس روابط محاسبه پذیر با مبدل های پشته ای با روابط محاسبه پذیر با مبدل های حالت متناهی تحت اشتراک بسته نیستند. به طور مثال $\{ \geq 0 \}$ $\{ (a^i b^j c^k, d^k) \mid i, j, \geq 0 \}$ با یک مبدل پشته ای قابل محاسبه است و $\{ (a^i b^i c^j, d^i) \mid i, j \}$ $\{ k \ 0 \}$ بایک مبدل حالت متناهی قابل محاسبه است. درهر صورت اشتراک $\{ \geq 0 \}$ $\{ (a^n b^n c^n, d^n) \mid n \}$ این دو رابطه نمیتواند با یک مبدل پشته ای محاسبه شود.

برای زبان های مستقل از متن قضیه ی زیر وجود دارد.

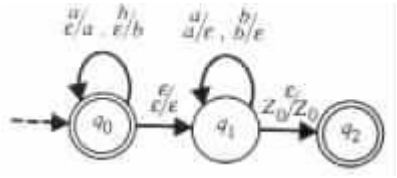
قضیه ۳-۵-۱. رده ی زبان های مستقل از متن با زبان های منظم تحت اشتراک بسته هستند.

اثبات. اتوماتای پشته ای $M_1 = <Q_1, \Sigma, \Gamma, \delta, q_{01}, Z_0, F_1>$ 1 , و اتوماتای حالت

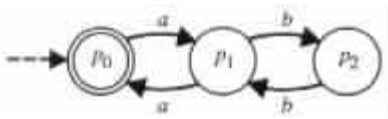
متناهی , $M_2 = <Q_2, \Sigma, \delta_2, q_{02}, F_2>$, را در نظر می گیریم. بدون از دست دادن عمومیت فرض می کنیم که M_2 بدون ε و قطعی است. (قضیه ی ۲-۳-۱ را ببینید.) اشتراک $L(M_1)$ و $L(M_2)$ با اتوماتای پشته ای $< \Sigma, \Gamma, \delta_3, [q_{01}, q_{02}], Z_0, F_1 \times F_2 >$ $M_3 = <Q_1 \times Q_2,$ پذیرفته می شود. جدول انتقال δ_3 شامل $([q,q'],\alpha,\beta,[p,p'],\gamma)$ است اگر و تنها اگر $(q, \alpha, \beta, p, \gamma)$, در δ_1 باشد و M_2 با صفر یا یک حرکت می تواند با خواندن a از حالت'q به حالت'p برسد.

در نتیجه ، M_3 یک اتوماتای پشته ای است که محاسبات M_1 و M_2 را به صورت موازی شبیه سازی میکند به گونه ای که محاسبات شبیه سازی شده برای خواندن هر نماد ورودی M_1 و M_2 هماهنگ هستند. با استقرا روی n می توان نشان داد که M_3 یک ورودی $a_1 \ldots a_n$ را می پذیرد اگر وتنها اگر M_1 وM_2 آن را قبول کنند.

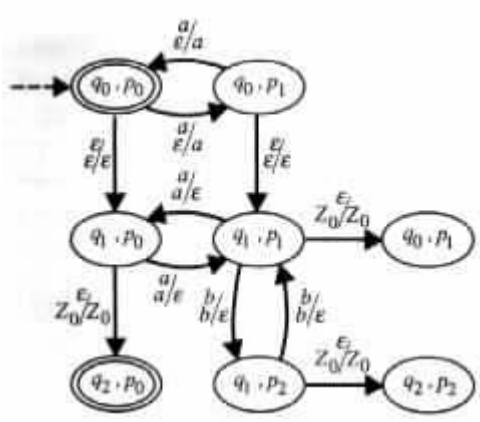
مثال ۳-۵-۱.اتوماتای پشته ایM_3 که نمودار انتقال آن د رشکل ۳-۵-۱ (c) داده شده است.



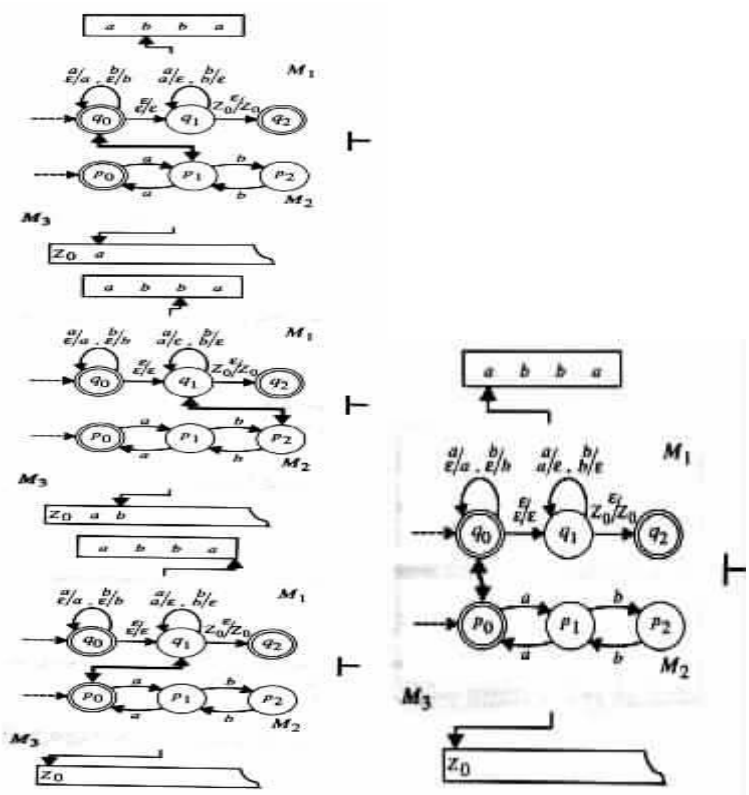
(a)

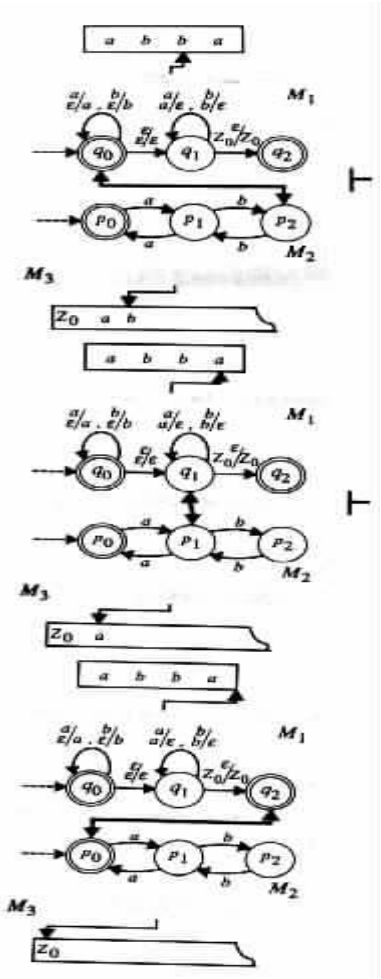


(b)



(c)





(d)

شکل ۳-۵-۱.زبان پذیرفته شده با اتوماتای پشته ای M_3 در (c) اشتراک زبان پذیرفته شده توسط اتوماتای پشته ای M_1 در (a) و زبان پذیرفته شده توسط اتوماتای حالت متناهی M_2 در (b).محاسبه ای M_3 روی ورودی abba در (d) نشان داده شده است.

اشتراک زبان پذیرفته شده توسط اتوماتای پشته ای M_1 که نمودار انتقال آن در شکل ۳-۵-۱(a) و زبان پذیرفته شده با اتوماتای حالت متناهی M_2 که نمودار انتقال آن درشکل ۳-۵-۱(b) داده شده است را می پذیرد . محاسبه ی M_3 روی ورودی abba

در شکل ۳-۵-۱(d) نشان داده شده است.

زبان $\{ a^i b^i c^j \mid i, j \geq 0 \}$ و $\{ a^i b^j c^j \mid i, j \geq 0 \}$ زبان هایی هستند که با اتوماتای پشته ای قطعی پذیرفته میشوند و اشتراک $\{ a^n b^n c^n \mid n \geq 0 \}$ از این زبان ها مستقل از متن نیست. در نتیجه کلاس زبان هایی که با اتوماتای پشته ای قطعی پذیرفته می شوند تحت اشتراک بسته نیستند.

به هرحال قضیه بعد نشان میدهد که این کلاس تحت متمم بسته است. برهان قضیه از لم زیر استفاده می کند.

تعریف : دنباله ای از حرکات $(uq_1v, z_1) \vdash \ldots \vdash (uq_kv, z_k)$ از یک اتوماتای پشته ای M یک حلقه خوانده می شود اگر $M, k>1$ میتواند از ترکیب (uq_1v, z_1) با قوانین انتقال یکسان با ترکیب (uq_kv, z_k) حرکت کند. و z_1 یک پیشوند از z_i است برای $i=2,\ldots,k$. حلقه ، یک حلقه ساده نامیده می شود اگر هیچ حلقه ای به جز خود را نپذیرد.

لم ۳-۵-۱.هر اتوماتای پشته ای قطعی M_1 یک اتوماتای پشته ای قطعی M_2 دارد که روی تمام ورودی ها توقف می کند.

اثبات. M_1 را به عنوان اتوماتای پشته ای قطعی در نظر میگیریم. t را به عنوان عددی که قوانین انتقال M_1 را مشخص می کند در نظر می گیریم. M_1 روی یک ورودی x داده شده توقف می کند اگروتنهااگر روی x وارد یک حلقه ساده شود. به علاوه هر حلقه ساده ی M_1 بیشتر از t^t حرکت را شامل نمی شود. اتوماتای پشته ای M_2 مطلوب می تواند با به کارگیری این مشاهده از M_1 ساخته شود.

به خصوص M_2 همان M_1 است که برای استفاده از نمادهای نشانه گذاری شده در پشته ی خود همانند یک شمارنده تغییر یافته است. در این صورت c در کنترل حالت متناهی خودش است. به محض خواندن یک نماد ورودی و به محض حذف کردن یک نماد نشانه گذاری شده از پشته ، M_2 بالاترین نماد در پشته اش را نشانه گذاری می کند و مقدار c را در شروع هر محاسبه صفر تعیین می کند.

از طرفی M_2 هر بار که یک حرکت از M_1 را شبیه سازی میکند، مقدار c را افزایش می

دهد. به محض دریافت یک مقدار $t^t + 1$ در c اتوماتای پشته ای M_2 زمانی که M_1 وارد یک حلقه ساده می شود و M_2 در یک پیکربندی پذیرفته نشده توقف می کند ، قطعی می شود.

اثبات قضیه ی زیر یک پالایش از قضیه ی ۲-۵-۲ است ، برای نشان دادن اینکه کلاس زبان های منظم تحت متمم بسته است.

قضیه ۳-۵-۲. کلاس زبان هایی که توسط اتوماتای پشته ای قطعی پذیرفته می شود تحت متمم بسته است.

برهان. اتوماتای پشته ای قطعی M را در نظر می گیریم. با استفاده از لم ۳-۵-۱ می توان فرض کرد که M فقط محاسبات متوقف شده دارد و بدون از دست دادن عمومیت ، $|\mathcal{T}| \leq ۱$ در هر قانون انتقال $(q,a,\beta,p,\mathcal{T})$, را می توان فرض کرد.

از M تا M_{eof}

M_{eof} را یک اتوماتای پشته ای قطعی در نظر میگیریم که L(M) را می پذیرد . و در هرکدام از محاسباتش بعد از استفاده کردن از همه ی ورودی ها متوقف می شود. M_{eof} میتواند از M به روش زیر ساخته شود. در ابتدا M_{eof} را یک M با حالت تله q_{trap} اضافه شده ، در نظر می گیریم . و برای هر نماد ورودی a قانون انتقال را به صورت $(q_{trap}, a, \varepsilon , q_{trap}, \varepsilon)$, اضافه می کنیم . آنگاه تا زمانی که M_{eof} یک حرکت از حالت q روی ورودی a ندارد و روی پشته محتوی β است، به طور تکراری یک قانون انتقال جدید به صورت $(q , a, \beta , q_{trap} , \beta)$ را به M_{eof} اضافه می کنیم.

حذف حالات مرکب

یک حالت q از یک اتوماتای پشته ای را حالت خواندن می نامیم اگر نشانه ورودی در هر قانون انتقال به صورت $(qa ,\beta,p,\mathcal{T})$, باشد که درحالت q شروع میشود. حالت q را یک حالت ε می نامیم اگر در هر قانون انتقال به صورت $(q\varepsilon ,\beta,p,\mathcal{T})$, که درحالت q شروع می شود $a = \varepsilon$ باشد.

اگر q حالت مرکب از M_{eof} باشد آنگاه قوانین انتقال $(q,a,\beta,p,1)$, از M که $|a|=1$ را ارضا می کند می تواند با یک جفت از قوانین انتقال $(q,\varepsilon,\beta,q,\varepsilon)$, و $(q,\varepsilon,p,a,1)$, $(q_f,a,p,a,1)$ جایگزین شود ، جایی که q_f یک واسطه ی جدید باشد حالت غیر قابل پذیرش است. چینن انتقالهای M_{eof} می تواند به وضعیتی که حالت مرکب ندارد تغییر میکند.

تغییرات M_{eof}

M_{eof} می تواند برای بدست آوردن یک اتوماتای پشته ای قطعی M_{eof_max} فقط با تفاوت در نقطه ی توقف بیشتر تغییر کند.به صورتی که M_{eof_max} در یک حالت خواندن باشد. اصلاحات می تواند به صورت زیر انجام شود:

الف) درابتدا M_{eof_max} به صورت M_{eof} باشد. ب) هر حالت q از M را به [q, accept] تغییر نام می دهیم اگر در یک حالت پذیرش باشد و به حالت [q, reject] تغییر نام می دهیم اگر در یک حالت پذیرفته نشده باشد.

ج) تازمانی که اتوماتای پشته ای M_{eof_max} یک قانون انتقال به صورت $([p, a], [p, 1], [q, accept,], \varepsilon, \beta reject],$, , داشته باشد آن را با یک قاعده ی انتقال به صورت $([q, accept], \varepsilon, \beta, [p, accept], 1)$ [,جایگزین می کنیم . به علاوه اگر [p, accept] یک حالت جدید باشد آنگاه برای هر قاعده به صورت $([p, reject], a, \beta', p', '1)'$, اضافه می شود :

۱- یک قانون انتقال به صورت $([p, accept], a, \beta, p', '1)$ اگر $p' \neq [p, reject]$ یا $a \neq \varepsilon$

۲- یک قانون انتقال به صورت $([p, accept], a, \beta, [p, accept], 1)$ [,اگر [p, =p'reject] و $a = \varepsilon$

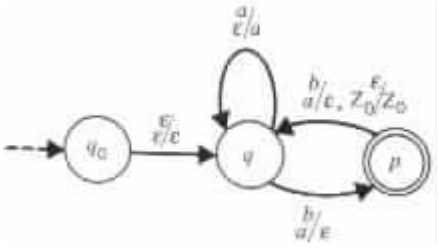
د) یک حالت M_{eof_max} را یک حالت پذیرش در نظر می گیریم اگر و تنهااگر یک حالت خواندن به صورت [q, accept] باشد.

تبدیلات بالا خاصیت پذیرش حالت های ε را تاحالت خواندن بلاکه(blocking) شدن منتشر می کند.

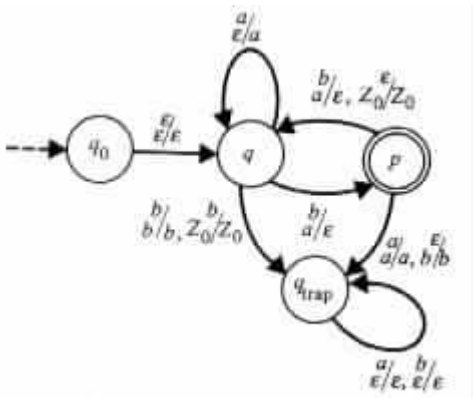
اتوماتای پشته ای که متمم L(M) را می پذیرد

اتوماتای پشته ای M_{eof_max} ساخته شده روی یک ورودی یک دنباله منحصر به فرد از حرکت هایی که انتهایشان در یک حالت خوانده شدن بعد از استفاده کردن از همه ی ورودی هاست ، دارد. دنباله حرکت ها حتی زمانی که یک زیر مجموعه ی متفاوت از مجموعه ی حالت های خواندن برای مجموعه ای از حالات پذیرش بودن انتخاب می شود، تغییر نمی کند. سپس اتوماتای پشته ای قطعی که متمم L(M) را می پذیرد می تواند با درخواست اینکه حالت خواندنی به صورت [q, reject] حالات پذیرش باشند، از M_{eof_max} به دست بیاید.

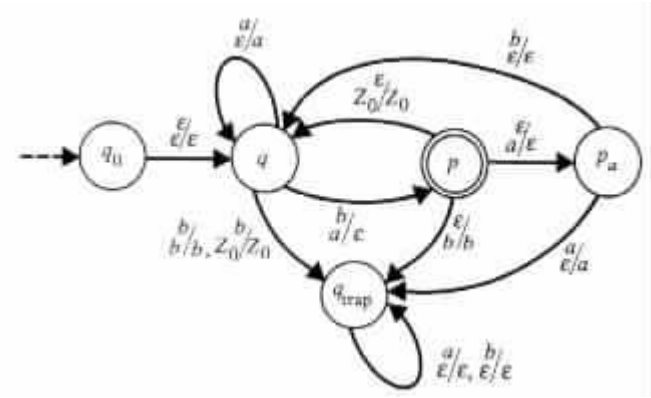
مثال ۳-۵-۲. اتوماتای پشته ای قطعی M را که نمودار انتقال آن درشکل ۳-۵-۲(a) آمده است درنظر میگیریم.



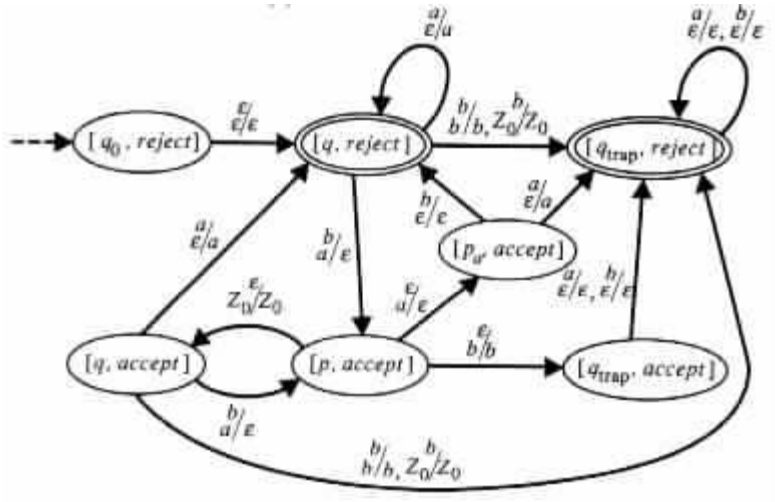
(a)



(b)



(c)



(d)

شکل ۳-۵-۲.(a) یک اتوماتای پشته ای M.(b) متمم M_{eof} از M . (c) تغییرات M که شامل حالت های مرکب نمی باشد.(d) یک اتوماتای پشته ای که متمم L(M) را می پذیرد.

با استفاده از اصطلاحاتی که در برهان قضیه ی ۳-۵-۲(b) آمده است. حالت q_0 از M یک حالت ε است، حالت q یک حالت خواندن است و حالت p یک حالت مرکب است.

نمودار انتقال M_{eof} در شکل ۳-۵-۲(b) آمده است ، نمودار انتقال M_{eof} تغییر یافته شده به وضعیتی که شامل حالت مرکب نباشد درشکل ۳-۵-۲(c) آمده است. نمودار انتقال شکل ۳-۵-۲(d) از یک اتوماتای پشته ای قطعی است که متمم $L(M)$ را می پذیرد.

بسته بودن تحت متمم برای کلاس زبان هایی که توسط اتوماتای پشته ای قطعی پذیرفته می شوند و بسته نبودن کلاس ها تحت اشتراک وبا استفاده از قوانین دمورگان،می توان فهمید که این کلاس تحت اجتماع بسته نمی باشند.

نتیجه ۳-۵-۱. زبان هایی وجود دارند که با اتوماتای پشته ای غیر قطعی پذیرفته می شوند اما نمی توانند با هر اتوماتای پشته ای قطعی پذیرفته شوند.

۳-۶ خاصیت تصمیم پذیری برنامه های بازگشتی با دامنه ی متناهی

اولین قضیه ی این بخش یک تعمیم برای تصمیم پذیری مساله ی تهی بودن برای اتوماتای حالت متناهی بیان میکند.

قضیه ۳-۶-۱. مساله ی تهی بودن برای اتوماتای پشته ای تصمیم پذیر است.

اثبات:اتوماتای پشته ای $M_1 = <Q,\Sigma, \Gamma, \delta_1,q_0, Z_0, F>$. را در نظر میگیریم.C را نیز به عنوان یک نماد جدید که در Σ نیست در نظر میگیریم.همچنین δ_2 را به جای δ_1 در نظر میگیریم.با توجه به اینکه هر قاعده ی انتقال به صورت $(q,\varepsilon, \beta, p,\gamma)$, با یک قاعده ی انتقال به صورت (q, c,β , p,γ) , جایگزین شود. M_2را به عنوان اتوماتای پشته ای $_2$. $<Q,\Sigma \cup \{c\} , \Gamma, \delta_2, q_0, Z_0, F>$, در نظر میگیریم.

از طریق برهان میبینیم که M_2اتوماتای پشته ای M_1است که به صورتی تغییر یافته است که هرگاه M_1حرکتی بسازد که هیچ نماد ورودی نخواند ، بتواند نماد c را بخواند.با این ساختار ، M_1میتواند وضعیت (uqv, w) را در t حرکت به دست بیاورد اگر و تنها اگر u_c وجود داشته باشد که M_2بتواند وضعیت (u_cqv, w) را در t حرکت به دست آورد به صورتی که u_c رشته ای باشد که از u با الحاق تعدادی c بدست آمده باشد و $|u_c| = t$.سپس $T(M_1) = \emptyset$ اگر و تنها اگر $T(M_2) = \emptyset$.

m به عنوان ثابتی مشخص میشود که لم فشار برای زبانهای مستقل از متن برای

$L(M_2)$ به کار رود.کوتاهترین رشته ی x در $L(M_2)$ نمیتواند بلندتر از M باشد در غیر اینصورت یک تناقض رخ میدهد.زیرا با توجه به لم فشار اگر x در $L(M_2)$ باشد و طولش کمتر از M باشد آنگاه یک رشته ی کوتاهتر نیز در $L(M_2)$ وجود دارد.

روی ورودی x اتوماتای پشته ای M_2 میتواند حداکثر |x| تا حرکت داشته باشد.در نتیجه تهی بودن $L(M_2)$ یا به طور معادل $L(M_1)$ میتواند با بررسی تمام دنباله های اجرای ممکن M_2 یا به طور معادل M_1 چک شود که شامل تعداد حرکت بیشتر از M نباشد.

تصمیم پذیری مساله ی تهی بودن برای اتوماتای پشته ای میتواند برای نشان دادن تصمیم پذیری یک مسائل دیگر برای مبدل های حالت متناهی استفاده شود.به عنوان یک مثال تصمیم پذیری مسئله ی هم ارزی برای مبدل های حالت متناهی قطعی.برای کلاس های عمومی مبدل های پشته ای همانند کلاس اتوماتای پشته ای ،این مسئله تصمیم ناپذیر است.(نتیجه ی ٤-٧-١ و نتیجه ی ٤-٧-٢ به ترتیب).از طرفی برای اتوماتای پشته ای قطعی و مبدل ها ی پشته ای قطعی مسئله باز است.

نتیجه ٣-٦-١.مساله ی هم ارزی برای مبدل های حالت متناهی قطعی تصمیم پذیر است.

اثبات:مبدل حالت متناهی قطعی M_1 و M_2 را در نظر میگیریم.از M_1 و M_2 یک اتوماتای حالت متناهی M_3 میتواند ساخته شود به صورتی که M_3 مجموعه ی تهی را بپذیرد اگر و تنها اگر $L(M_1) = L(M_2)$.این ساختار میتواند به صورت برهان قضیه ی ٤-٦-٢ باشد.

از طرفی از M_1 و M_2 یک اتوماتای پشته ای M_4 نیز میتواند ساخته شود که یک ورودی را اگر و تنها اگر M_1 و M_2 هر دو آن را بپذیرند، تا زمانی که خروجی های متفاوت بوجود آورند بپذیرد. M_4 مجموعه ی متناهی را اگر و تنها اگر M_1 و M_2 در خروجی هایشان روی ورودی هایی که هر دو میپذیرند توافق داشته باشند ، میپذیرد.

یک محاسبه ی M_4 روی یک ورودی شامل شبیه سازی های موازی است ، به صورتی که در برهان قضیه ی ٣-٥-١ محاسبات M_1 و M_2 روی چنین ورودی ای

انجام شود.شبیه سازی مطابق با هر یک از موارد زیر است ، جایی که انتخاب به طور غیر قطعی ساخته شده است.

مورد ۱: M_4 محاسبات پذیرش M_1 و M_2 را شبیه سازی میکند که خروجی هایی با طول متفاوت تولید میکند.در طول شبیه سازی M_4 از خروجی های M_1 و M_2 چشم پوشی میکند . بنابراین ، در هر مثال از شبیه سازی ، پشته ی M_4 مقدار مطلق تفاوت بین طول خروجی های تولید شده توسط M_1 و M_2 را بدست میآورد. M_4 ورودی ای را میپذیرد اگر و تنها اگر در پایان ورودی با یک پشته ی غیر تهی به حالت پذیرش M_1 و M_2 برسد.

مورد ۲: M_4 محاسباتی از M_1 و M_2 را میپذیرد که خروجی های متفاوت در jامین نمادشان تولید کند.برای jهایی که بزرگتر از طولشان نباشد شبیه سازی به مورد ۱ شبیه است .تفاوت اصلی این است که M_4 در پشته تغییرات در طول خروجی M_i را فقط تا زمانی ثبت میکند که به طور غیر قطعی مشخص شود که M_i برای $i=1,2$ به jامین نماد خروجی اش میرسد.علاوه بر این M_4 در کنترل حالت متناهیش jامین نماد خروجی M_1 و M_2 را ثبت میکند.به محض کامل شدن شبیه سازی M_4 ورودی را میپذیرد اگر و تنها اگر پشته اش خالی باشد و کنترل حالت متناهی مجزا باشند.

با گرفتن M_3 و M_4 یک اتوماتای پشته ای M_5 میتواند برای پذیرفتن $L(M_4)$ ل $L(M_3)$ ساخته شود. M_5 مجموعه ی تهی را میپذیرد اگر و تنها اگر M_1 و M_2 معادل باشند.این تیجه ی از قضیه ی ۳-٦-۱ بدست آمده است.

مسئله ی توقف یکنواخت برای اتوماتای پشته ای غیر قابل تصمیم گیری است(نتیجه ی ٤-۷-۳).بنابراین تصمیم گیری مساله ی تهی بودن برای اتوماتای پشته ای میتواند برای نشان دادن تصمیم پذیر بودن مساله ی توقف یکنواخت برای اتوماتای پشته ای قطعی استفاده شود.

قضیه ۳-٦-۲. مساله ی توقف یکنواخت برای اتوماتای پشته ای قطعی تصمیم پذیر است.

اثبات: اتوماتای پشته ای قطعی M_1 را در نظر میگیریم.از M_1 یک اتوماتای پشته ای

قطعی M_2 شبیه به آنچه در برهان لم ۳-۵-۱ آمده است میتواند ساخته شود.تنها تفاوت این است که اینجا M_2 یک ورودی را میپذیرد اگر و تنها اگر تعیین کند که M_1 به یک حلقه ساده میرسد.با این ساختار، M_2 یک مجموعه ی تهی را میپذیرد اگر و تنها اگر M_1 روی تمام ورودی ها متوقف شود.

برهان قضیه ی قبل برای اتوماتای پشته ای غیر قطعی شکست میخورد زیرا محاسبات پذیرش ی اتوماتا ی پشته ای غیر قطعی میتواند شامل حلقه ساده باشد بدون اینکه مجبور باشد به یک حلقه نا متناهی وارد شود.

قضیه ۳-۶-۳.مساله ی توقف برای اتوماتای پشته ای تصمیم پذیر است.

اثبات:زوج (M, x) از یک اتوماتای پشته ای M و یک ورودی x برای M را در نظر میگیریم.از x یک اتوماتای حالت متناهی M_x میتواند ساخته شود که فقط ورودی x را بپذیرد.بنابراین از M یک اتوماتای پشته ای M_1 میتواند ساخته شود برای پذیرفتن یک ورودی اگر و تنها اگر M دنباله ای از قوانین انتقالی که M را به یک حلقه ساده روی ورودی هدایت میکند.ساختار میتواند شبیه برهان قضیه ی ۳-۶-۲ باشد.

از M و M_x یک اتوماتای پشته ای $M_{a,x}$ میتواند ساخته شود که اشتراک $L(M)$ با $L(M_x)$ را بپذیرد و(قضیه ی ۳-۵-۱ را ببینید.) با این ساختار $M_{a,x}$ یک مجموعه ی غیر تهی را میپذیرد اگر و تنها اگر M، x را بپذیرد.با توجه به قضیه ی ۳-۶-۱ اگر $M_{a,x}$ یک مجموعه ی غیرتهی را بپذیرد میتواند قطعی شود.هم چنین آنگاه M در توقف روی ورودی x قطعی می شود .در غیر اینصورت با یک راه مشابه یک اتوماتای پشته ای $M_{1,x}$ میتواند ساخته شود که اشتراک $L(M_1)$ و $L(M_x)$ را بپذیرد. با این ساختار $M_{1,x}$ مجموعه ی تهی را میپذیرد اگر و تنها اگر M فقط محاسبات متوقف شده روی ورودی x داشته باشد.این نتیجه از قضیه ی ۳-۶-۱ بدست آمده است.

فصل چهارم

ماشین تورینگ

اهداف

در پایان فصل، دانشجو با مفاهیم زیر آشنا می‌شود:

مبدل های تورینگ

حالتها و حرکات مبدل های تورینگ

قطعیت و عدم قطعیت در مبدل های تورینگ

محاسبات مبدل های تورینگ

ماشین های تورینگ

مقدمه

مطالعه برنامه های با حافظه های متناهی و برنامه های با دامنه متناهی بازگشتی، به طور قابل ملاحظه ای درمعرفی مبدل های متناهی و سیستم های ریاضی و مبدل های پشته ای مفید واقع شده است . مزیت این سیستم های ریاضی میله ای از تجرد آنها از معطوف بودن به ماشین های محاسبه اولیه که رفتار برنامه ها را شبیه سازی می کنند است . با توجه به این مطلب طبیعی است که سعی کنیم تا به دنبال راهکارهای مشابه در مطالعه کلاس عمومی برنامه ها باشیم .

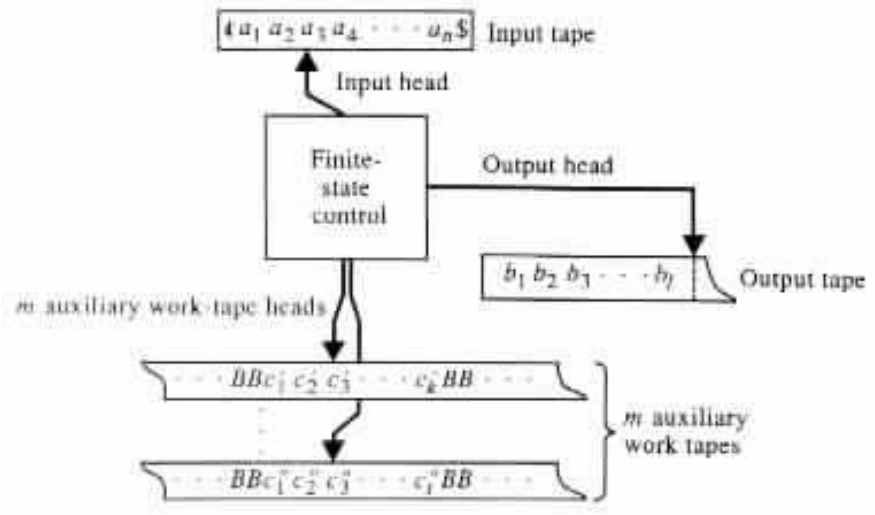
برنامه های با با دامنهٔ متناهی بازگشتی، به عنوان تعمیمی از برنامه های با حافظه متناهی معرفی شده اند . همینطور مبدل های پشته ای به عنوان تعمیمی از مبدل های با حالت متناهی معرفی شده اند . با رفتن به سمت برنامه های با کلاس عمومی پیشنهاد می شود که سعی گردد یک تعمیم برای مبدل های متناظر قرار گیرد .

۱-٤ مبدل های تورینگ

در میان عمومی ترین مدلهای مبدلها که به ذهن می آیند احتمالاً همانهایی هستند که اجازه دارند که بیشتر از یک نوار کار کمکی ،نوارهای کمکی محدود ،ورودی با سر دوطرفه ، قرار گرفتن ورودی بین نشانه گزارهای پایانی وپذیرش هر جایی در ورودی ها ،استفاده کنند . کلاس چنین مدل هایی که مبدل ها ی تورینگ نامیده می شود در زیر معرفی می شوند .

هر مبدل تورینگ M می تواند مانند یک ماشین محاسبه انتزاعی دیده شود که شامل یک کنترل حالت متناهی ، یک نوار ورودی ،m نوار کار کمکی برای $m > 0$ ، یک هد خواندن و نوشتن با نوارکار برای هد نوار کار کمکی یک نوار خروجی و یک نوار خروجی فقط نوشتنی است .

به شکل ۱-۱-٤ توجه شود .



شکل ٤,١,١ شمای یک مبدل تورینگ

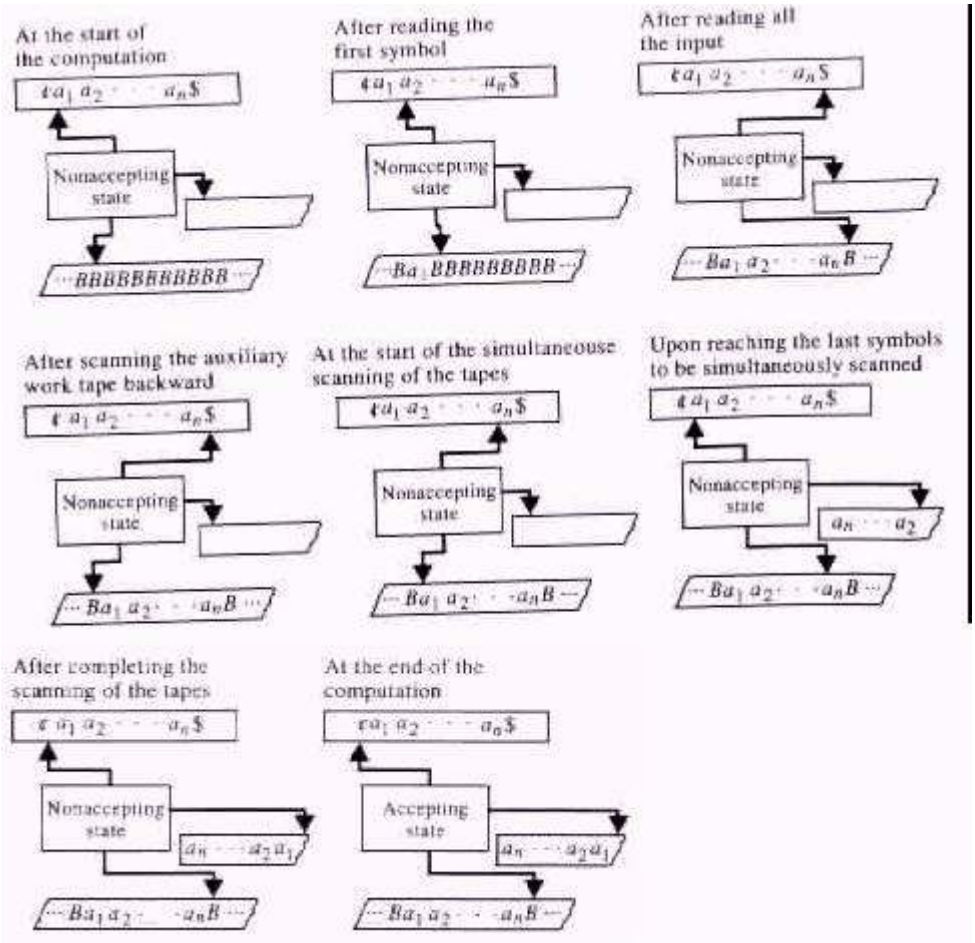
هر حرکت M ، به وسیله حالت M ، نماد زیرهد ورودی و سمبل های زیر هد نوار های کار کمکی ، تعریف می شود .هر حرکت M ، شامل تغییر حالت M ، تغییرسمبل زیر هد هر نوار کار کمکی ، تغییر محل هر هد بوسیله قرار گرفتن در هر جهت ، ونوشتن یک سمبل در داخل نوار خروجی می باشد .

در ابتدا فرض می شود که M ، ورودی ها یش را که a_1،...، a_n هستند در نوار ورودی بین یک end marker چپ و یک end marker راست $ ذخیره نماید . با توجه به این، فرض می شود که هد ورودی در ابتدای ورودی قرار گیرد، نوارهای کار معین فقط شامل سمبل B هستند و نوار خروجی خالی فرض می شود .

مثال: ٤-١-١ : یک مبدل تورینگ نوار کار معین M می تواند رابطه

$\{ (x, x^{rev}) \mid x \text{ is in } \{a, b\}^* \text{ and } x = x^{rev} \}$ را با چک کردن هر ورودی a_1،...،a_n را محاسبه کنید طوریکه a_1،...،a_n = $a_1, \ldots a_n$.

محاسبات Mمی تواند در روش ذیل انجام شود .



شکل ٤-١-٢ شرح نمایش محاسبه رابطه $\{ (x, x^{rev}) \mid x \text{ is in } \{a, b\}^* \text{ and } x = x^{rev} \}$.

بوسیله مبدل تورینگ M هرمحاسبه رابا یک حرکت به جلودرنوار ورودی وهمزبان نوارکارکمکی ، یک تغیرمکان به راست$ end marker در زمان واحد(یک زمان) که در نوار خروجی مواجه می شود، شروع می گردد، در زمانی که M روی نوارها حرکت

می کند سمبل های خوانده شده از ورودی را روی نوار کار کمکی کپی می نماید.سپس M نوار کار معین را به عقب برمی گرداند و اولین سمبل غیر تهی را مکان یابی می کند. در نهایت M نوار ورودی را رو به عقب و هم زمان نوار کار کمکی را به جلو، سمبل به سمبل می پیماید. وقتی که M دو نوار را می پیماید برای اینکه سمبل هایی در حرکت خوانده شده اند مساوی باشد چک می کند.

یک سیستم ریاضی M شامل یک هشت تایی $<Q,\Sigma,\Gamma,\Delta,\delta,q_0,B,F>$ است که یک مبدل تورینگ با m نوار کار کمکی نامیده می شود. برای $m\geq o$ که از شروطی پیروی می نماید.

Q: یک مجموعه متناهی است و اعضای Q حالتهای M نامیده می شوند.

Σ,Γ,Δ: الفبا هستند که شامل سمبل های $ و ¢ نیستند . Σالفبای ورودی M ، و اعضایΣ سمبل های ورودی M، نامیده می شوند. Γالفبای نوارکمکی است و عناصر آن نمادهای نوار کار کمکی نام دارند .Δ الفبای خروجی M است و عناصر آن نمادهای خروجی M نام دارند .

δ:یک رابطه از

$$Q\times(S\ U\ \{¢,\$\})\times Gm \quad to \quad Q\times\{-1,0,+1\}\times(G\times\{-1,0,+1\})m\times(D\ U\ \{e\})$$

که جدول انتقال M و

اعضای $(q, a, b_1, b_2, \ldots, b_m, (p, d_0, c_1, d_1, c_2, d_2, \ldots, c_m, d_m, \rho))$

و یابرای سادگی $(q, a, b_1, b_2, \ldots, b_m, p, d_0, c_1, d_1, c_2, d_2, \ldots, c_m, d_m, \rho)$ از جدول انتقال δ قوانین انتقال M نامیده می شود .

q_0: یک عضو از Q است که حالت ابتدایی M نامیده می شود.

B: یک سمبل در Γ است که سمبل تهی M نامیده می شود.

F: یک زیر مجموعه از Q است و حالتهای در F ، حالتهای پذیرش یا پایانی M نامیده می شود.

¢: یک سمبل است که end marker چپ و $ end marker راست نامیده می شود.

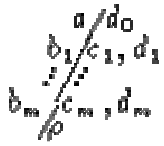
مثال ٤-١-٢ . $M = < Q, \Sigma, \Gamma, \Delta, \delta, q_0, B, F >$ یک مبدل تورینگ با یک نوار کار کمکی است که

$Q=\{q_0,q_1,q_2,q_3,q_4\}$, $\Sigma = \Delta =\{a,b\}$, $\Gamma =\{a,b,B\}$ and $\delta =$ $\{(q_0,a,B,q_1,+1,a,+1,a)$, $(q_0,b,B,q_1,+1,b,+1,b)$, $(q_0,\$,B,q_4,0,B,0,\varepsilon)$, $(q_1,a,B,q_1,+1,a,+1,a)$, $(q_1,b,B,q_1,+1,b,+1,b)$, $(q_1,a,B,q_2,0,B,-1,\varepsilon)$, $(q_1,b,B,q_2,0,B,-1,\varepsilon)$,$(q_2,a,a,q_2,0,a,-1,\varepsilon)$, $(q_2,b,a,q_2,0,a,-1,\varepsilon)$, $(q_2,a,b,q_2,0,b,-1,\varepsilon)$, $(q_2,b,b,q_2,0,b,-1,\varepsilon)$, $(q_2,a,B,q_3,0,B,+1, \varepsilon)$, $(q_2,b,B,q_3,0,B,+1, \varepsilon)$, $(q_3,a,a, q_3,+1,a,+1, \varepsilon)$, $(q_3,b,b,q_3,+1,b,+1,$ $(q_3,\$,B,q_4,0,B,0, \varepsilon)$, (εمبدل تورینگ M ، ۵ حالت و ١٦ قاعده دارد . } .

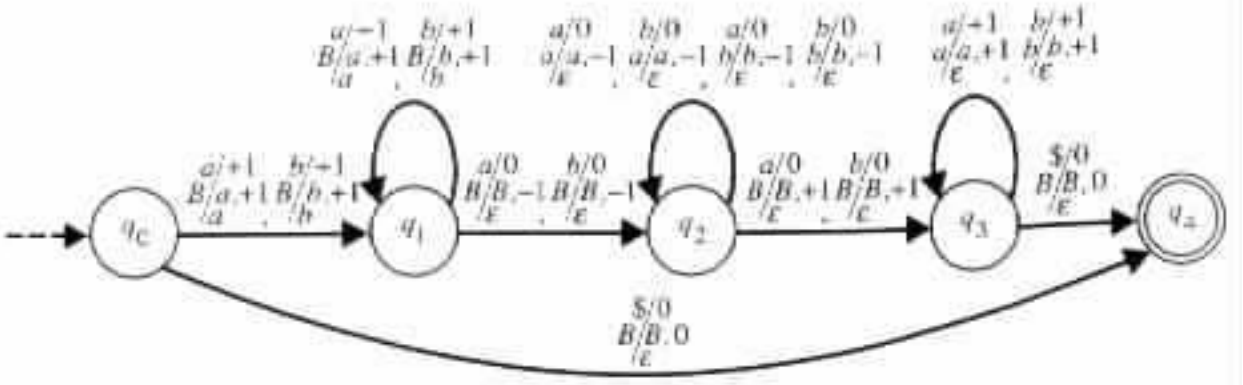
Mاز حالت q_0 و به عنوان یک حالت ابتدایی استفاده می نماید و حالت q_4 یک حالت پذیرش است. سمبلB به عنوان سمبل خالی M در نظر گرفته شده است . یک سیستم ریاضی M مبدل تورینگ نامیده می شود که آن یک مبدل تورینگ نوار کار معین mتایی برای ۰ =< m است .

هر مبدل تورینگ $M = < Q, \Sigma, \Gamma, \Delta, \delta, q_0, B, F >$ می تواند بوسیله یک دیاگرام انتقال به صورت گرافیکی ارائه شود. برای هر مرحله در Q دیاگرام انتقال با یک گره متناظر به وسیله یک دایره نشان داده می شود. حالت اولیه بوسیله یک پیکان از جای ابتدایی به یک گره نشان داده می شود. هر حالت پذیرش بوسیله دو دایره نمایش داده می شود.

هر قاعدهٔ انتقال $(q, a, b_1, b_2, \ldots, b_m, p, d_0, c_1, d_1, c_2, d_2, \ldots, c_m, d_m, \rho)$ در δ بوسیله یک یال از گره که از حالت q بوسیله گره به حالت p می رود ارائه می شود که هر یال یک بر چسب از فرم زیر دارد.



در بر چسب ردیف بالا"a/d_0" با نوار ورودی متناظر است ، ردیف پایینی "$/\rho$" با نوار خروجی متناظر است و ردیف "bi/ci, di" با نوارIام از نوار کارهای کمکی متناظر است .برای راحتی کارها یال هایی که مبدا و مقصد آنها یکی است ادغام می شوند و بر چسب های آنها با کاما جدا می شوند.



شکل ٤,١,٣

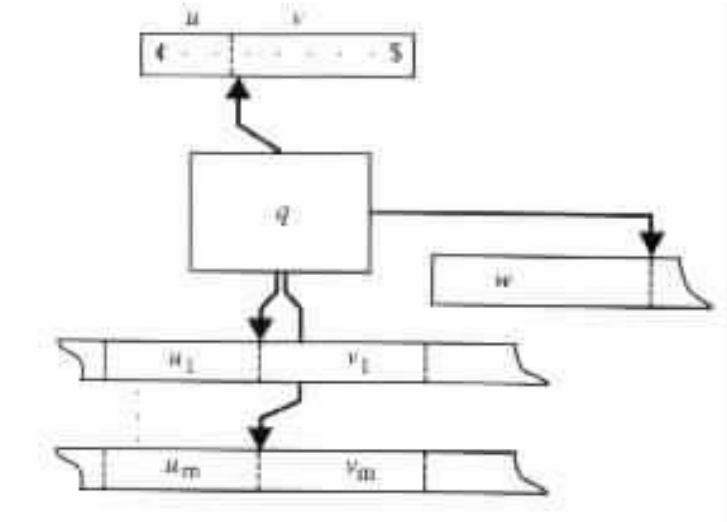
مثال : ٤-١-٣ : دیاگرام انتقال در شکل ٤-١-٣ یک مثال از ارائه ٤-١-٢ مبدل تورینگ است. قاعدهٔ انتقال $(q_0, a, B, q_1, +1, a, +1, a)$ در دیاگرام انتقال بوسیله یک

یال از حالت q_0 به حالت q_1 که بر چسب زیر را دارد ارائه شده است.

$$B/a,+1$$

پیکربندی ها و حرکت های مبدل های تورینگ :

در هر ورودی x از Σ^* ، مبدل تورینگ M تعدادی مجموعه از پیکربندی های ممکن را دارد . هر پیمربندی یا توصیف آنی از مبدل تورینگ M یک (m+2) تایی (w وu_mqv_m و.....وu_1qv_1 و uqv) است که q یک حالت M است و uv =¢x$ و uivi یک رشته Γ^* برای هر 1<= i<=m است و w یک رشته در Δ^* میباشد . مستقیماً می بینیم که پیکربندی (w وu_mqv_m و.....وu_1qv_1 و uqv) می گوید که M در حالت q با هد ورودی در اولین سمبل v با iامین نگهدارندهٔ نوار کار کمکی ..... Bu_iv_iB..... با iامین هد نوار کاری کمکی در اولین سمبل v_iB و با نگهدارندهٔ نوار خروجی w است .



شکل ٤,١,٤

با حفظ جامعیت فرض می شود Q و $\Sigma \cup \Gamma \cup \{¢,\$\}$ دوبه دو مجزا هستند .

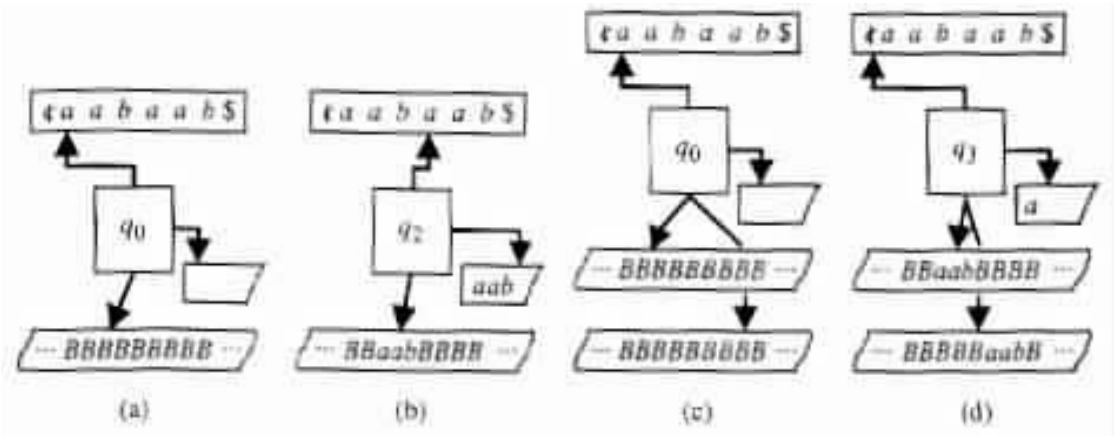
به پیکربندی ، پیکربندی اولیه گفته می شود که اگر $q = q_0, u = ¢, w = \varepsilon$, and $u_i v_i = \varepsilon$ برای هر $1 <= i <= m$. پیکربندی اولیه می گوید که در ابتدای یک محاسبه ، ورودی در نوار ورودی ذخیره می شود و بوسیله end marker در چپ و $ end marker در راست محدود شده است . هد ورودی روی سمبل سمت راست ¢ قرار گرفته است . واین علامت در سمت چپ ترین سمبل ورودی زمانی که ورودی خالی نیست وجود داردو در سمت راست $end marker است. نوارهای کار کمکی فقط با B تنظیم شده اند و کنترل حالت متناهی در حالت اولیه قرار گرفته شده است .

یک پیکربندی را پیکربندی پذیرش می گویند اگر q در یک حالت پذیرش در F باشد.

مثال ٤-١-٤ : فرض کنید M_1 یک مبدل تورینگ با یک نوار کار کمکی باشد (شکل٤-١-٣).

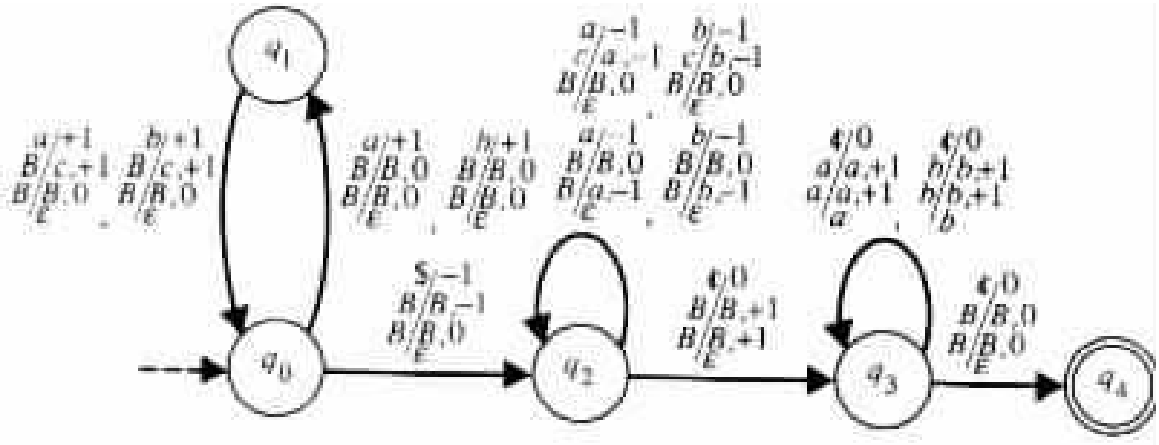
پیکربندی ابتدایی m در ورود aabaab به صورت $(¢q_0aabaab\$, q_0, \varepsilon)$است .

درچنین موارد ورودی همچنین M_1 وضعیت $(¢aabq_2aab\$, aq_2ab, aab)$ را دارد . پیکربندی ها در شکل (a) ٤-١-٥ (b) ٤-١-٥ نشان داده شده است.



شکل ٤,١,٥

فرض کنید M_2 مبدل تورینگ با دو نوار کار معین را در شکل ٤-١-٦ نمایش دهیم .



شکل ٤-١-٦ یک مبدل تورینگ نوار کار کمکی دوتایی

مبدل تورینگ در زمان ورودaabaab پیکربندی ابتدایی $(\text{¢}q_0aabaab\$, q_0, q_0, \varepsilon)$ رادارد. به طور مشابه $(q_3\text{¢}aabaab\$, aq_3ab,$, $q_3Baab, a)$یک پیکربندی درM_2 با چنین ورودی است. قاعده های انتقال تورینگ M برای تعریف حرکات ممکنM استفاده می شوند.حالت ها در شکل های

4.1.5.(c) , 4.1.5.(d) نشان داده شده اند.

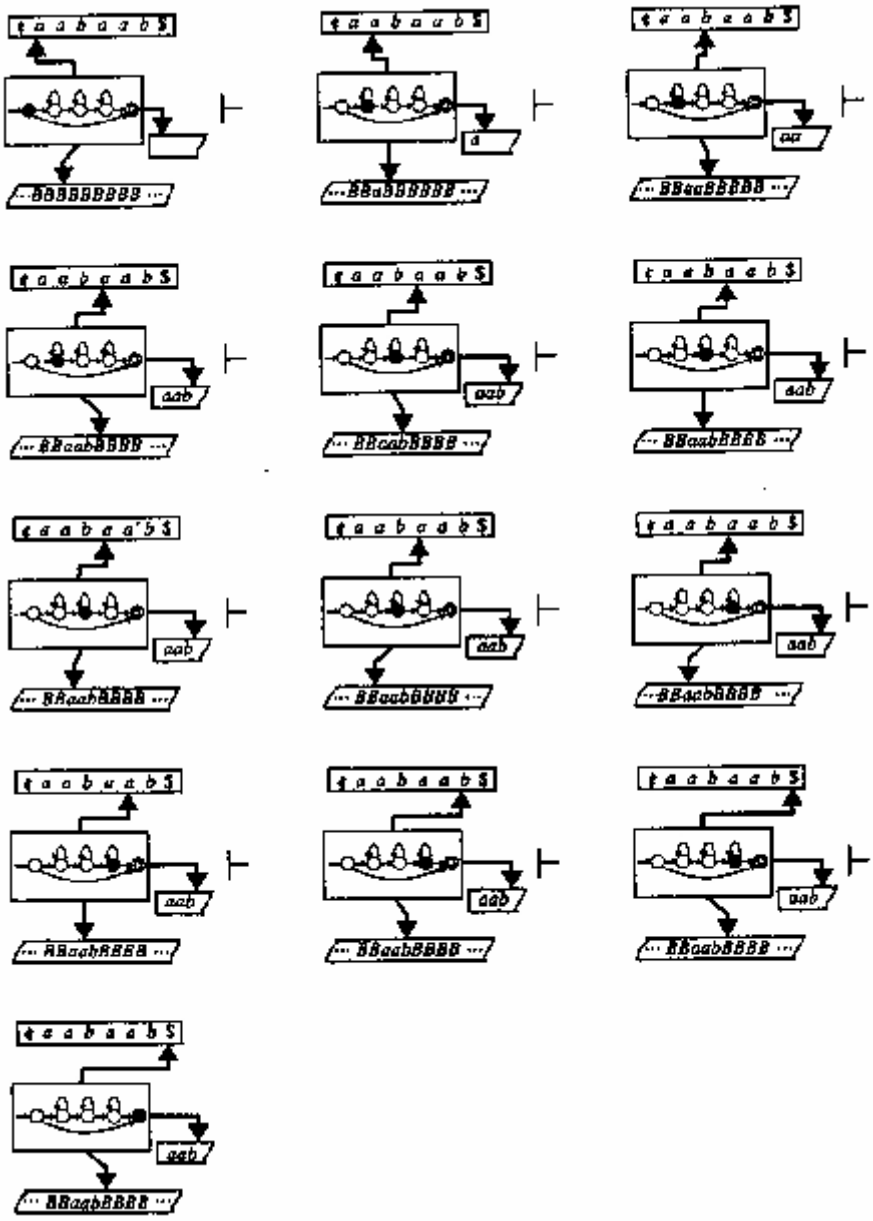
قواعد انتقال مبدل تورینگ M برای تعریف حرکات ممکن M استفاده میشود.هر حرکت با برخی از قواعد انتقال متناظر است. یک حرکت در قاعدهٔ انتقال

$(q, a, b_1, b_2, \ldots, b_m, p, d_0, c_1, d_1, c_2, d_2, \ldots, c_m, d_m,)$حالت کنترل حالت متناهی را از q به p تغییر می دهد ، یک نوار ورودی را می پیماید ، موقعیت هد ورودی را به اندازه d_0 به راست حرکت می دهد، ρ را درنوار خروجی می نویسد،موقعیت هد خروجی را به اندازه|ρ| به راست تغییر می دهد ، سمبل b_i را در

iامین نوار کار کمکی می پیماید ، سمبل b_i را با c_i جایگزین می نماید، و موقعیت iامین نوار کار کمکی d_i را به راست تغییر می دهد . برای $1<= i <=m$.

حرکتی از M از حالت C_1 به C_2 بوسیله $C_1 \vdash_M C_2$ یا $C_1 |- C_2$ نشان داده می شود، اگر M تعریف شده باشد. دنباله ای از صفر یا چند حرکت M از حالت C_1 به C_2 با $C_1 \vdash_M^* C_2$ یا به صورت ساده تر با $C_1 \vdash^* C_2$ نمایش می دهند.

مثال. ٤-۱-۵ مبدل تورینگی که نمودار انتقال آن درشکل ٤-۱-۳ داده شده است را درنظر بگیرید. در ورودی aabaab دنباله حرکات زیر بین پیکربندی هاروی می دهد.(به شکل ٤-۱-۷ نگاه کنید.



شکل ٤-١-٧ دنباله حرکات بین حالت های مبدل تورینگ شکل ٤-١-٣ میباشد.

$(¢q_0aabaab\$, q_0, e)$ $(¢aq_1abaab\$, aq_1, a)$

$|-$ $(¢aaq_1baab\$, aaq_1, aa)$

$|-$ $(¢aabq_1aab\$, aabq_1, aab)$

$|-$ $(¢aabq_2aab\$, aaq_2b, aab)$

$|-$ $(¢aabq_2aab\$, aq_2ab, aab)$

$|-$ $(¢aabq_2aab\$, q_2aab, aab)$

$|-$ $(¢aabq_2aab\$, q_2Baab, aab)$

$|-$ $(¢aabq_3aab\$, q_3aab, aab)$

$|-$ $(¢aabaq_3ab\$, aq_3ab, aab)$

$|-$ $(¢aabaaq_3b\$, aaq_3b, aab)$

$|-$ $(¢aabaabq_3\$, aabq_3, aab)$

$|-$ $(¢aabaabq_4\$, aabq_4, aab)$.

دنباله فقط می تواند در پیکربندی اولیه شروع شود ودر پیکربندی پذیرش برای ورودی aabaab پایان یابد. .

قطعیت وغیر قطعیت درمبدل های تورینگ:

طبیعت قطعیت و غیرقطعیت در مبدل ها تورینگ مشابه آنچه که در مبدل های پشته ای و مبدل های با حالت متناهی است می باشد .به هر حال بیان این ویژگی ها برای مبدل های تورینگ ساده شده است ،به این دلیل که قواعد انتقال دقیقأیک سمبل را در یک نوار در هر حرکت می پیماید .در مورد مبدل های با حالت متناهی و مبدل های پشته ای هدها می توانند صفریایک سمبل را بپیمایند .

در آغاز ،ما می گوییم مبدل های تورینگ قطعی هستند ،اگر هر زوج از قواعد انتقال که از یک حالت آغاز می شوند ،در پیمایش نوار ها ، سمبل های مختلفی را ببینند. به طور واضح یک مبدل تورینگ $M = <Q,\Sigma ,\Gamma ,\Delta ,\delta , q_0, B, F>$ قطعی نامیده می شود اگر هیچ جفت قواعد انتقالی مانند زیر وجود نداشته باشد

$$(q, a, b_1, \ldots, b_m, p, d_0, c_1, d_1, \ldots, c_m, d_m, \rho)$$

و

$$(q', a', b'_1, \ldots, b'_m, p', d'_0, c'_1, d'_1, \ldots, c'_m, d'_m, \rho')$$

در δ که م $(q, a, b_1, \ldots, b_m) = (q', a', b'_1, \ldots, b'_m)$.

یک مبدل تورینگ غیرقطعی نامیده می شود اگر یک مبدل تورینگ قطعی نباشد.

مثال ٤-١-٦ مبدل تورینگ مثال ٤-١-٢ یک مبدل تورینگ غیر قطعی ست .زوج قواعد انتقال

$$(q_1, a, B, q_1, +1, a, +1, a) \quad \text{and} \quad (q_1, a, B, q_2, 0, B, -1, \epsilon)$$

و زوج

$$(q_1, b, B, q_1, +1, b, +1, b) \quad \text{and} \quad (q_1, b, B, q_2, 0, B, -1, \epsilon).$$

دلایلی برای غیر قطعی بودن مبدل تورینگ هستند. زوج اول قواعد انتقال در پیشوند (q,a,B) با هم یکی هستند ودومین زوج در پیشوند (q,b,B) تطابق دارد.

به هرحال مبدل تورینگ در شکل ٤-١-٦ قطعی است. هیچ یک ازقواعد انتقال که از یک حالت مشترک آغاز می شوند ، در سمبل های زیر هدهای متناظر یکسان نیستند .

برای مثال زوج

$$(q_0,a,B,B,q_1,+1,B,0,B,0,\varepsilon) \text{ and } (q_0,b,B,B,q_1,+1,B,0,B,0,\varepsilon)$$

از قواعد انتقال در سمبل هایی که در نوار ورودی می پیمایند یکسان نیستند و زوج

$$(q_3,c,a,a,q_3,0,b,+1,b,+1,b) \text{ and } (q_3,c,b,b,q_3,0,a,+1,a,+1,a)$$

از قواعد انتقال در سمبل هایی که در نوار های کار کمکی پیمایش می کنند یکسان نیستند.

محاسبات مبدل های تورینگ:

تعاریف محاسبات برای مبدل های با حالت متناهی و مبدل های پشته ای برای مبدل های تورینگ نیز به کار می رود. به خصوص محاسبه پذیرش مبدل تورینگ M که دنباله حرکات M است که در پیکربندی اولیه شروع می شود و در پیکربندی پذیرش پایان می یابد.یک محاسبه عدم پذیرش یا مردودM ، دنباله ای از حرکات در ورودی x که شرایط زیر در آن برقرار باشد:

a- دنباله از پیکربندی اولیه M در ورودی x شروع شود.

b- اگر دنباله متناهی باشد،در پیکربندی پایان می پذیرد که هیچ حرکتی ممکن نباشد.

M-c در ورودی x محاسبه پذیرش نداشته باشد.

هر محاسبه پذیرش و غیرپذیرش M یک محاسبه از M نامیده می شود.

یک محاسبه را ،محاسبه توقف گویند اگر تعداد متناهی حرکت در آن موجود باشد.

مثال٤-١-٧ مبدل قطعی تورینگ که نمودار انتقال آن در شکل ٤-١-٦ آمده را در نظر بگیرید.مبدل تورینگ یک محاسبه پذیرش در ورودی داده شده دارد اگر و فقط اگر ورودی به صورت WWباشد کهW رشته ای از $\{a,b\}^*$ است. در یک ورودی به فرم WW مبدل تورینگ رشته W را در نوار خروجی می نویسد.

هر محاسبه مبدل تورینگ با خواندن ورودی آغاز می شود.به محض خواندن سمبل های فرد از ورودی،آن را از حالت q_0 به q_1 حرکت می دهد در حالیکه در زمان رفتن نوارهای کار کمکی تغییر نمی کنند. به محض خواندن سمبل های زوج از ورودی،مبدل تورینگ در حالیکه c را در اولین نوار کار کمکی می نویسد از حالت q_0 به q_1

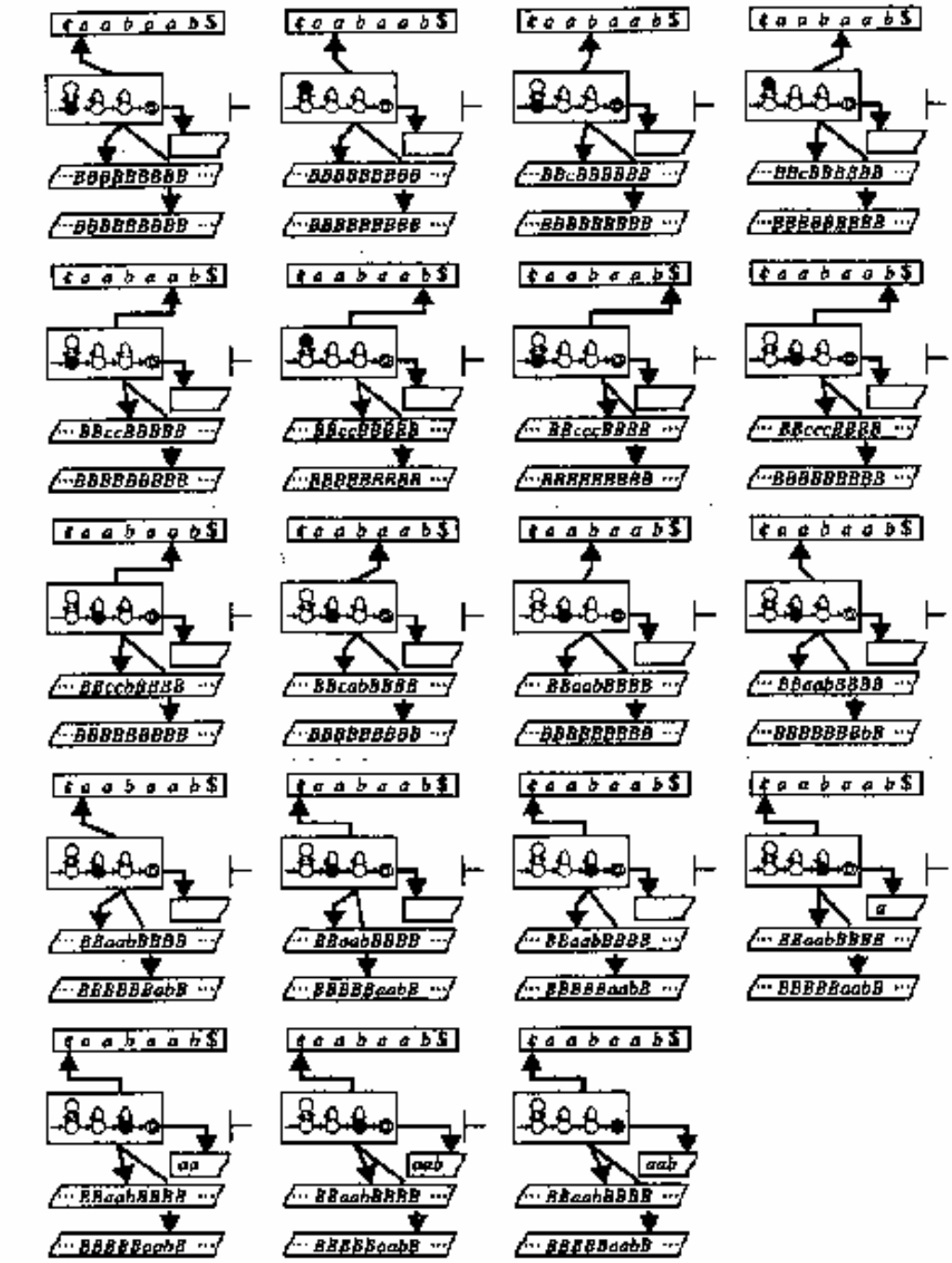
میرود.

در ورودی های با طول فرد تورینگ در حالت q_1 وقتی به endmarker $ برسد، توقف می کند .در ورودی های با طول زوج مبدل تورینگ وقتی به endmarker $ راست برسد وارد حالت q_2 می شود. در حرکت از حالت q_0 به q_2 تعداد C در دومین نوار کار کمکی برابر نصف طول ورودی است.

در حالت q_2 مبدل تورینگ ورودی از فرم xy را که |x|=|y| باشد رابه طور وارونه می خواند. همینطور مبدل تورینگ y را به طور وارونه می خواند و محتوای $C^{|y|}$ اولین نوار کار کمکی را با رشته y جایگزین می نماید. سپس مبدل تورینگ x را به طور وارونه می خواند و آن را در دومین نوار کار کمکی می نویسد.

به محض رسیدن به endmarker ¢ چپ مبدل تورینگ یک انتقال از حالت q_2 به q_3 انجام می دهد. در حالت q_3 دو نوار کار کمکی را برای کنترل اینکه x=y است می پیماید. اگر مساوی باشند مبدل تورینگ از حالت q_3 به q_4 حرکت می نماید در غیر این صورت در q_3 متوقف می شود.

محاسبه ای که مبدل تورینگ ورودی aabaab را دارد در شکل 4.1.8 نشان داده شده است.



شکل ٤-١-٨ یک محاسبه مبدل تورینگ

بنا به تعریف،هر حرکت در محاسبه باید در یک قاعدهٔ انتقال باشد که در نهایت موجب می شود که محاسبه در یک حالت پذیرش متوقف شود. به هر حال بیشتر قواعد انتقال ممکن است برای یک حرکت باشند هر یک می تواند به ترتیب انتخاب شوند.به طور مشابه،هرگاه هیچ یک از قواعد انتقال برای راهبری محاسبه برای توقف در یک حالت پذیرش ممکن نباشند ، ،سپس دوباره هر یک از قواعد انتقال ممکن می تواند انتخاب شوند.

یک ورودی x توسط مبدل تورینگ M پذیرفته شده یا شناخته شده نامیده می شود اگر M یک محاسبه پذیرش در ورودی x داشته باشد. یک محاسبه پذیرش روی ورودی x که در پیکربندی به فرم $(uqv, u_1qv_1, \ldots, u_mqv_m, w)$ پایان می یابد دارای خروجی w می باشد. خروجی یک محاسبه غیرپذیرش تعریف نشده است.

همانند مبدل های با حالت متناهی و مبدل های پشته ای یک مبدل تورینگ،ممکن است دنباله ای از حرکت ها روی ورودی داشته باشد که پذیرفته شوند و آنها به عنوان محاسبه در نظر گرفته نشوند.

مثال ٤-١-٨ به مبدل تورینگ غیر قطعی که دیاگرام انتقال آن در شکل ٤-١-٣ داده شده است توجه نمایید. مبدل تورینگ یک ورودی را می پذیرد اگر و تنها اگر آن به صورت WWباشد که W رشته ای از $\{a,b\}^*$ باشد . روی چنین ورودی WW، خروجی W را تولید می کند.

هر محاسبه در M_1 در ورودی غیر تهی xy با خواندن x شروع می شود زمانی که M_1 ، x را از نوار ورودی اش می خواند ، رشته را در نوار کار کمکی می نویسد. در پایان x ، که به صورت غیر قطعی پیدا می شود، M_1 ، از حالت q_1 به q_2 منتقل می شود.

در حالت q_2، مبدل تورینگ M_1 روی کپی ذخیره شده x در نوار کار کمکی عقب گرد می نماید تا اینکه مکان اولین نماد در رشته را بیابد .سپس M_1 به حالت q_3 می رود .
در حالت q_3 ، M_1 کنترل می نماید که باقیماندهٔ y از ورودی برابر رشته x ذخیره شده در نوار کار کمکی می باشدیا نه . مبدل تورینگ ورودی را می پذیرد اگروتنها اگر تعیین کند که x=y .

تعاریف دیگر،از قبیل روابطی که مبدل های تورینگ محاسبه می کنند،زبانهای پذیرفته شده و زبانهای تصمیم پذیر توسط آنها ، مشابه موارد گفته شده در بخش 2.2 در مبدل با حالت متناهی و در بخش 3.2 در مبدل پشته ای است.

مثال ٤-١-٩ دیاگرام انتقال مبدل تورینگ غیرقطعی M_1درشکل ٤-١-٣ داده شده است ودیاگرام مبدل تورینگ قطعی M_2 درشکل 4.1.6 نمایش داده شده است که رابطه { (ww, w) | w is in {a, b}* } را محاسبه می نماید.

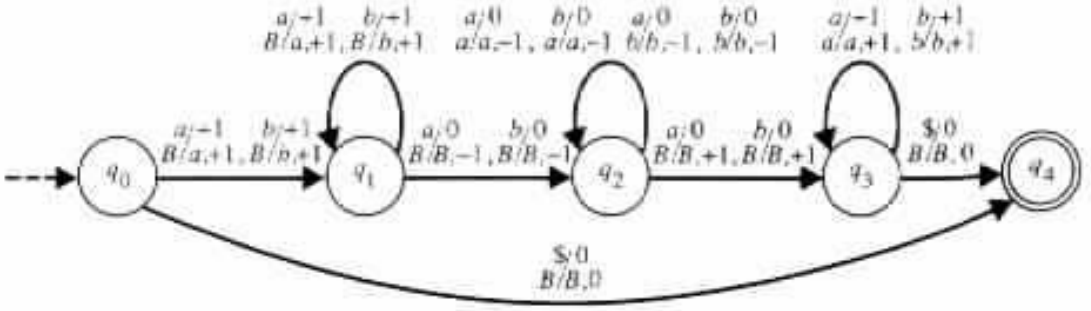
یک زبان را به صورت بازگشتی شمارش پذیر گویند اگر بوسیله مبدل تورینگ قابل پذیرش باشد. یک زبان بازگشتی است اگر بوسیله مبدل تورینگ تصمیم پذیر باشد.

ماشین های تورینگ:

مبدل های تورینگی که مؤلفه های خروجی آنها نادیده گرفته شود را ماشین تورینگ می نامند. به عبارت دیگر برای $m\geq 0$ یک ماشین تورینگ با m نوار کار کمکی یک هفت تایی شامل $M = <Q,\Sigma ,\Gamma ,\Delta ,\delta , q_0, B, F>$ است که $Q, \Sigma, \Gamma, q_0, B, F$مانند مبدل های تورینگ تعریف می شوند و δ یک رابطه از $Q \times (\Sigma \text{ U } \{¢, \$\}) \times \Gamma^m$ to $Q \times \{-1, 0, +1\} \times (\Gamma \times \{-1, 0, +1\})^m$ است . یک سیستم ریاضی M ، یک ماشین تورینگ نامیده می شود اگر آن یک ماشین تورینگ با m نوار کار کمکی برای بعضی ازmها باشد.

دیاگرام های انتقال مشابه آنچه که در مبدل های تورینگ استفاده شد، می توانند برای ماشین های تورینگ استفاده شوند. تنها تفاوتی که دارد بر چسب یال ها شامل خروجی نمی شوند.

مثال ٤-١-١٠ ماشین تورینگی که از مبدل تورینگ شکل ٤-١-٣ گرفته شده است در شکل ٤-١-٩ نشان داده می شود.



شکل ٤-١-٩ یک مبدل تورینگ با یک نوار کار معین

یک ماشین تورینگ را یک LBA یا ماشین خطی کراندار گویند اگر برای هر ورودی x داده شده ماشین تورینگ حداکثر$\max(|x|,1)$ محل را در نوار کار کمکی ملاقات نماید.

نظریه چرچ: در سال های اخیر شخصیت های گوناگون به توصیف مفهوم محاسبه پذیری پرداخته اند. این شخصیت ها استفاده از راههای مختلفی را پیشنهاد نموده اند که شامل مدلهایی از مبدل های تورینگ قطعی هستند. به هر حال واضح است که این شخصیت ها از لحاظ فکری همراستا هستند و می توانند از یک شخصیت به شخصیت دیگر موثر باشند. موازی بودن فکر شخصیت های متفاوت ما را به حدس هایی راهنمایی می کند.

نظریه چرچ: یک تابع محاسبه پذیر (محاسبه پذیر پاره ای) است اگرو فقط اگر توسط

مبدل تورینگ قطعی قابل محاسبه (قابل محاسبه پاره ای) باشد.

هیچکس نمی تواند انتظار داشته باشدکه قادر است درستی نظریه چرچ را اثبات کند،چون مشخصات دقیقی برای درک مستقیم تصور محاسبه پذیری وجود ندارد. بهترین چیزی که می توان انتظار داشت افزایش اطمینان برای پیدا کردن عدم موفقیت و پرداختن به مثال های گفته شده است.

٤-۲. برنامه ها و مبدل های تورینگ

۱) از برنامه ها تا مبدل های تورینگ

۲) از مبدل های تورینگ تا برنامه ها

تعریف یک برنامه به مفهوم محاسبه پذیری توابع و گزاره های آن تکیه دارد. در مورد برنامه ها با حافظه متناهی و برنامه های بازگشتی با دامنه متناهی محاسبه پذیری توابع برنامه ها و گزاره ها آن به متناهی بودن دامنه متغیرها دلالت دارد. از طرف دیگر برای کلاس عمومی برنامه ها موضوع محاسبه پذیری توابع و گزاره ها نیازمند حل صریح است.

از برنامه ها به مبدل های تورینگ:

با استفاده از نظریه چرچ، توابع برنامه ها و گزاره های آنها، می توان دید که محاسبات بوسیله مبدل های تورینگ قطعی شبیه سازی می شوند. در نتیجه می توان از یک فرض مشابه استفاده کرد وقتی که بتوان برنامه ها را بوسیله مبدل تورینگ شبیه سازی کرد.

هر برنامه p را در نظر بگیرید، D دامنه متغیر های Pرا مشخص می نماید وE،نمایش باینری برای D است. پس D می تواند بوسیله یک مبدل تورینگ M به فرم ذیل شبیه سازی شود.

M برای هر متغیر برنامه p یک نوار کار کمکی اختصاص می دهد. هر ورودی
$v_1, \ldots, v_n$ از برنامه p در M به وسیله یک رشته از
$E(V_1)\#\ldots\ldots\#E(V_n)$ نمایش داده می شود.هر خروجی $w_1,\ldots,w_t$ از P
در M با رشته ای به فرم $\#E(w_1)\#\ldots\#E(w_t)\#$ نمایش داده می شود.
E(u) برای نمایش با ینری u به کار می رود .

برای هر دستورالعمل که از x خوانده می شود مبدل تورینگ M یک مؤلفه داردکه مقدار(v)E را از ورودی بعدی v از P را می خواند ودر نوار کار کمکی که با x تناظر دارد ذخیره می نماید . به طور مشابه برای هر دستور العمل نوشته شده از x مبدل تورینگ M مؤلفه ای دارد محتوای نوار کار کمکی متناظر با x را به روی نوار خارجی کپی می کند.

برای هردستورالعمل $y := f(x_1, \ldots, x_m)$ مبدل تورینگ یک مؤلفه مشابه مبدل تورینگ قطعی دارد
که تابع $f(x_1, \ldots, x_m)$ رامحاسبه می نماید. تفاوت اساسی در این است که مؤلفه هایی که مقادیر $x_1, \ldots, x_m$ را (از نوار کار معین که با متغیر ها متناظر شده اند)، به جای ورودی می گیرند ودر عوض درمؤلفه نوارخروجی مقادیر تابع رادر نوارکار معین که با y متناظر شده است می نویسد .

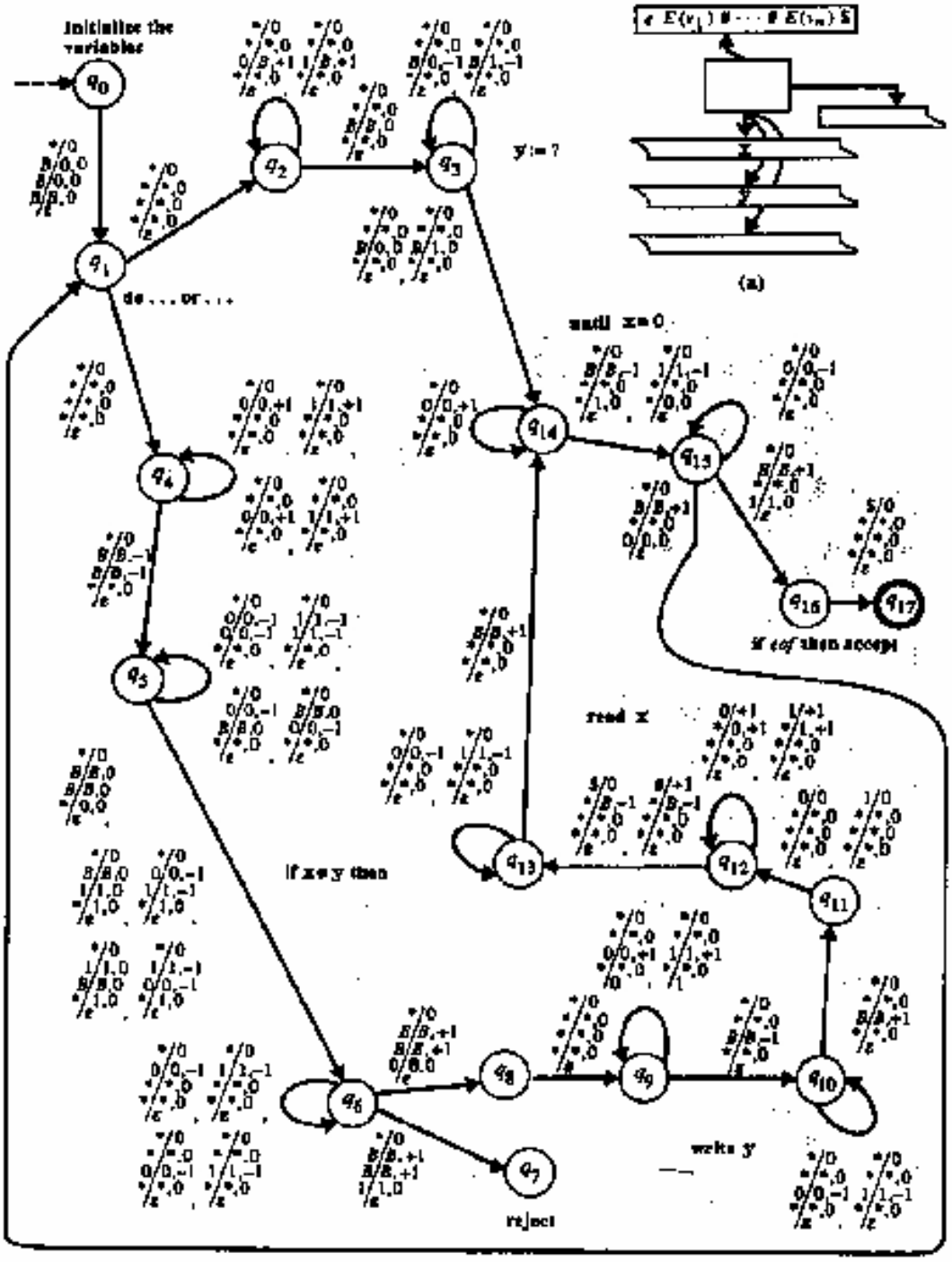
با استفاده از یک روش مناسب یک مؤلفه متناظر با هر قسمت دستور العمل در برنامه P وB همچنین یک مؤلفه برای ثبت مقادیر اولیه متغیر هایP در نظر گرفته می شود.به هر حال ، مؤلفه ها در Mهمان طور که در P مرتب شده اند ، قراردارند.
نکته:مبدل تورینگM زمانی قطعی است که برنامه P قطعی باشد .

مثال ٤-۲-۱ .برنامه Pرا٤۱-۲-۱ ببینید.

```
do
   y := ?
or
   if x ≠ y then
      reject
   write y
   read x
until x = 0
if eof then accept
```

برنامه شکل ٤-۲-۱

فرض می شود که دامنه متغیر های برنامه مجموعه اعداد طبیعی باشد وصفربه عنوان مقدار اولیه باشد .



شکل ٤-٢-٢ نمایی از مبدل تورینگ M که برنامه شکل ٤-٢-١ را شبیه سازی می کند را نشان می دهد .

نمای مبدل تورینگ P، M را شبیه سازی می کند .اولین نوار کار کمکی Mبرای ثبت مقادیر x استفاده می شود .دومین نوار برای ثبت مقادیر y وسومین نوار برای ثبت مقادیر گزاره ها به کار می رود .(0 برای غلط و یک 1 برای درست).

شکل ٤-٢-٢ (b)،دیاگرام انتقالMرانشان می دهد. هر یک از مؤ لفه های M هر زیر محاسبه را در شرایطی شروع می کنند و به پایان می رسانند که هر هدروی سمت چپ ترین سمبل غیر خالی از نوار متناظر قرار دارد .

قسمت مربوط به مقدار دهی اولیه متعیرها ، مقدار• رابرای اولین ودومین نوارکارکمکی قرار می دهد. قسمت "do ... or ..."به صورت غیر قطعی انتخاب بین مولفه های "y=?" یا "if x≠ y then" را انجام می دهد .

در حالتq_2 ،مولفه "y := ?" مقدار ذخیره شده در دومین نوار کار کمکی را برای y پاک می کند . سپس مؤلفه وارد حالت q_3 می شود که یک مقدار جدید برای yذخیره می نماید که به صورت غیرقطعی مشخص می شود.

قسمت "if x≠y Then" در حالت q_4 در سمت راست ترین ارقام در x وy قرار گرفته شده است. در حالت q_5 مولفه از جهت عکس میان ارقام x و y حرکت می نماید و تعیین می کند که آیا ارقام متناظر مساوی هستند یا نه . اگر مساوی بود، مولفه مقدار 0 را در سومین نوار کار کمکی قرارمی دهد و گرنه مقدار 1 را قرار می دهد. در حالت q_6 ، مولفه در سمت چپ ترین رقم x وy قرار می گیرد و بسته به مقدار ذخیره شده در سومین نوار کار کمکی کنترل به یکی از دو مولفه "reject" یا "write y" منتقل می شود . مولفه "write y" سمبل # را در حالت q_8 و y را در حالت q_9 به خروجی می فرستد . مولفه "read x" در حالت q_{11} بررسی می کند که ورودی

مقداری برای خواندن دارد وآن را در q_{12} می خواند . سپس در حالت q_{13} مولفه سمت چپ ترین رقم x را مکان یابی می کند .

مولفه"until x = 0" در حالت q_{14} بررسی می کند که آیا x صفر است یا نه ؟ اگر این طور بود ، مولفه یک را در سومین نوار کار کمکی ذخیره می کند . در غیر این صورت صفر ذخیره می گردد . در حالت q_{15}مولفه مکان سمت چپ ترین رقم x را معلوم می کند و سپس بسته به مقدار ذخیره شده در سومین نوار کار معین ، حرکت به یکی از مولفه های "do..... or....." یا " if eof then accept"انجام می شود .

قسمت "if eof then accept ازحالت q_{16} به حالت پذیرش q_{17} حرکت می نماید اگروفقط اگربه پایان ورودی رسیده باشد .

از مبدل ها ی تورینگ به برنامه ها :

به عنوان نتیجه ای از بحث قبلی ، می بینیم که الگوریتمی وجود دارد که هر برنامه داده شده به یک مبدل تورینگ هم ارز ترجمه می نماید . بر عکس الگوریتمی برای هر مبدل تورینگ $M = <Q,\Sigma ,\Gamma ,\Delta ,\delta , q_0, B, F>$ وجود دارد که یک برنامه هم ارز ارائه می دهد.

برنامه می تواند به صورت شکل ٤-٢-٣ باشد.

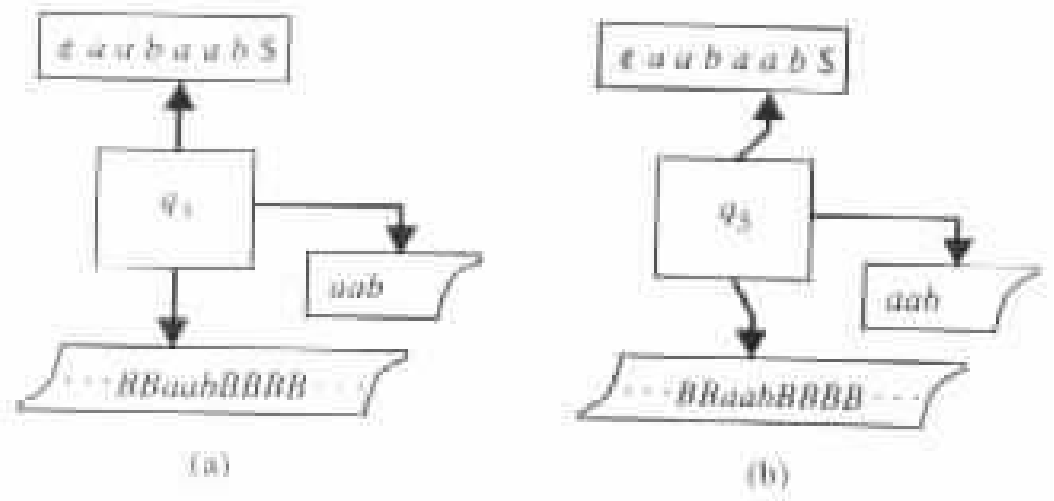
برنامه می تواند مبدل تورینگ را با روشی شبیه به روشی که یک برنامه با حافظه متناهی در بخش ۲-۲ به مبدل با حالت متناهی شبیه سازی شد ، شبیه سازی کند. همچنین برنامه مشابه یک برنامه بازگشتی با دامنه متناهی در بخش 3.2 یک مبدل پشته ای را شبیه سازی می نماید. تفاوت اصلی در ثبت محتوای نوار ها است.

```
/ * Record the initial configuration          (uqv, u1qv1, … , umqvm, w)
of M (see Figure 4.1.4).                                                    * /
state := q0
u := ¢
v := get(input)
for j := 1 to m do
   begin
      uj := B … B
     vj := B … B
  end
w := e
do
   / * Check for acceptance conditions. * /
   if F(state) then
   begin
      write w
      if eof then accept
      reject
   end
   / * Determine the transition rule
         (state, a, b1, … , bm, next_ state, d0, c1, d1, … cm, dm, r)
  to be used in the next simulated move.                                    * /
   a := top (v);  b1 := top (v1); … ; bm := top (vm)
  (next_ state, d0, c1, d1, … , cm, dm, r) := d(state, a, b1, … , bm)
  / * Record the changes in the input head position. * /
   case
      d0 = - 1 : a := top (u);  pop (u);  push (v, a)
     d0 = + 1 : push (u, a);  pop (v)
  end
   / * Record the changes in the auxiliary work tapes and in their corresponding
   head positions.                                                          * /
   for j = 1 to m do
      case
         dj = - 1 : pop (vj);  push (vj, cj);  bj := top (uj);
  pop (uj);  push (vj, bj)
        dj = 0 : pop (vj);  push (vj, cj)
       dj = + 1 : push (uj, cj);  pop (vj)
     end
   / * Record the output and modify the state. * /
   w := append (w, r)
  state := next_ state
until false
```

برنامه شکل ٤-۲-۳

فرض می شود که متغیرهای برنامه دارای دامنه اعداد طبیعی باشند. به طور بدیهی،به هر حال ما متغیرها را در دامنه $Q\ Y\ (\{¢, \$\} Y\Sigma)^* Y\Gamma^* Y\Delta^* Y\ \{-1, 0, +1\}\ )$ مشخص می نمائیم. برای هر یک از نوارهای غیر ورودی M، برنامه یک زوج متغیر پشته ای دارد. یک متغیر پشته ای برای نگهداری رشته کاراکترها در نوار در سمت چپ هد متناظر استفاده می شود. متغیر پشته ای دیگری برای نگهداری دنباله کاراکترها در نوار از وضعیت هد متناظر به سمت راست آن استفاده می گردد(در ترتیب معکوس). زوج متغیرهای پشته ای uوv برای ورودی نوار استفاده می شوند. زوج های v_i و u_i iامین نوار کار کمکی استفاده می شوند. متغیر W برای ثبت خروجی و متغیر state برای ثبت حالت به کار می رود.

مثال ٤-۲-۲ . برنامه پیکربندی (¢aabq3aab$,q3aab,aab) را با روش ذیل ثبت می نماید.



شکل ٤-٢-٤ مبدل تورینگ در حالت پذیرش

state	=	q3
u	=	¢aab
v	=	$baa
u1	=	B…B
v1	=	B…Bbaa
w	=	aab

به طور مشابه برنامه حالت $(¢aabaq_3ab\$, aq_3ab, aab)$ را با روش ذیل ثبت می کند.

state	=	q3
u	=	¢aaba
v	=	$ba
u1	=	B…Ba
v1	=	B…Bba
w	=	aab

شکل ٤-۲-۲ (b)

شبیه سازی حرکت یک هد به سمت راست شامل push کردن یک سمبل در اواین متغیر پشته و POP کردن یک سمبل از دومین. به طور مشابه،شبیه سازی حرکت هد به سمت چپ شامل pop کردن یک سمبل از اولین متغیر پشته و push کردن یک سمبل به دومین متغیر پشته می باشد.

برنامه از top(var) برای تعیین بالاترین سمبل در var استفاده می نماید. همچنین از POP (var) برای برداشتن سمبل از var و از push(var,ch) و append (var,ρ) برای push کردن به ترتیب ρ,ch در var استفاده می نماید.

v := get (input) به عنوان یک قطعه کد در شکل٤-۲-۵ a نشان داده شده است.

برنامه شکل ٤,۲۵ (a)

```
read input
v := $
if not empty (input) then
do
char := top (input)
if not input_ symbol (char) then reject
pop (input)
push (v, char)
until empty (input)
(a)
```

$_{state}$(state, a, b_1, . . . , b_m) next_ state :=

$_{c1}$ (state, a, b_1, . . . , b_m) c_1 :=

$_{cm}$(state, a, b_1, . . . , b_m) c_m :=

$_{d1}$ (state, a, b_1, . . . , b_m) d_1 :=

$_{dm}$(state, a, b_1, . . . , b_m) d_m :=

(state, a, b_1, . . . , b_m) :=

(b)

next_ state := ?

c_1 := ?

c_m := ?

d_0 := ?

d_m := ?

:= ?

$_{tran}$(state, a, b_1, . . . , b_m, next_ state, if not

) then reject d_0, c_1, d_1, . . . , c_m, d_m,

(c)

Figure 4.2.5 (a) The code segment v := get (input). (b) The code segment (next_ state, d_0, c_1, d_1, . . . (state, a, b_1, . . . b_m) for a deterministic Turing transducer. (c)) := . , c_m, d_m, (state, a, b_1, . . . ,) := The code segment (next_ state, d_0, c_1, d_1, . . . , c_m, d_m, b_m) for a nondeterministic Turing transducer.

شکل ٤,٢٥ (a)

F(state) به عنوان یک جدول مراجعه ویژهٔ تابع می باشد و بررسی می کند که آیا state حالت پذیرش است یا نه؟

$$(\text{next\_ state}, d_0, c_1, d_1, \ldots, c_m, d_m, \rho) := \delta(\text{state}, a, b_1, \ldots, b_m)$$

به عنوان یک قطعه کد مانند شکل 4.2.5.b می باشد که برای مبدل های تورینگ قطعی و یک قطعه کد مانند شکل 4.2.5.c برای مبدل های تورینگ غیرقطعی است .

$\delta_{state}, \delta_{c1}, \cdots, \delta_{cm}, \delta_{d0}, \cdots, \delta_{dm}, \delta_{r}, \delta_{tran}$ به عنوان جدول مراجعه ویژهٔ توابع با اطلاعات خواسته شده است.

مثال ٤-٢-٣ برای مبدل تورینگ قطعی M_1 ، که دیاگرام آن در شکل ٤-١-٦ آمده است تساوی ها را دنبال نمایید.

$$\delta_{state}(q_0, a, B, B) = q_1$$
$$\delta_{t1}(q_0, a, B, B) = B$$
$$\delta_{t2}(q_0, a, B, B) = B$$
$$\delta_{p}(q_0, a, B, B) = \varepsilon$$
$$\delta_{t0}(q_0, a, B, B) = +1$$
$$\delta_{t1}(q_0, a, B, B) = 0$$
$$\delta_{t2}(q_0, a, B, B) = 0$$
$$\delta_{state}(q_2, a, c, B) = q_2$$
$$\delta_{t1}(q_2, a, c, B) = a$$
$$\delta_{t2}(q_2, a, c, B) = B$$
$$\delta_{p}(q_2, a, c, B) = \varepsilon$$
$$\delta_{t0}(q_2, a, c, B) = -1$$
$$\delta_{t1}(q_2, a, c, B) = -1$$
$$\delta_{t2}(q_2, a, c, B) = 0$$

برای مبدل تورینگ غیر قطعی M_2 که دیاگرام آن در شکل 4.1.3 آمده است تساوی های زیر برقرار هستند.

$$\delta_{tran}(q_0, a, B, q_1, +1, a, +1, a) = true$$
$$\delta_{tran}(q_0, b, B, q_1, +1, b, +1, b) = true$$
$$\delta_{tran}(q_0, \$, B, q_4, 0, B, 0, \varepsilon) = true$$
$$\delta_{tran}(q_0, a, b, q_2, 0, B, +1, \varepsilon) = false$$

برای M_1وM_2 تساوی های $F(q_4)=True$ و $F(q_0)=F(q_1)=F(q_2)=F(q_3)=$ false بر قرار هستند .

برنامه هر یک سمبل های موجود در $\{¢, \$, -1, 0, +1\}$ $Q \cup \Sigma \cup \Gamma \cup \Delta \cup$ را با اعداد مجزا بین 0 تا k-1

نمایش می دهد که k تعداد اعضای درون مجموعه $Q \cup \Sigma \cup \Gamma \cup \Delta \cup \{¢, \$, -1, 0, +1\}$ می باشد . بویژه سمبل تهی B با ۰ متناظر نشان داده می شود. متغیر ها اعداد طبیعی را نگهداری می نمایند که به صورت رشته هایی تعبیر می شوند که متناظر با نمایش در مبنای k اعداد هستند.

top (var) باقیماندهٔ تقسیم var بر k را بر می گرداند .

push(var,ch) مقدار var*k+ch را به متغیر var منسوب می کند.

pop(var) مقدار صحیح تقسیم var بر k را به متغیر var نسبت می دهد.

empty(var) مقدار True را اگر var=0 ، و مقدار False را اگر var=1 باشد برمی گرداند.

input- symbol(char) اگر char یک سمبل از Σ را نگهداری کند مقدارTrve وگرنه مقدار False را بر می گرداند.

append (var, ρ)، اگر ρ≠0 باشد مقدار k*var+r را و اگر $\rho = 0$ مقدار var را بر می گرداند.

مثال ٤-٢-٤ . M مبدل تورینگ قطعی است که دیاگرام انتقال آن در شکل ٤-١-٦ آمده است. برای هر M ، مجموعه $Q \cup \Sigma \cup \Gamma \cup \Delta \cup \{¢,\$,-1,0,+1\}$ با مجموعه $\{B,a,b,c,¢,\$,-1,0,+1,q_0,q_1,q_2,q_3,q_4\}$ برابر است و تعداد اعضای آن، k=14 می باشد. در زیر پیشنهاد شده است که عناصر مجموعه $Q \cup \Sigma \cup \Gamma \cup \Delta \cup \{¢,\$,-1,0,+1\}$رشته تهی ε مانند هر رشته B...B از سمبل های تهی با ۰ نمایش داده شود. و به ترتیب ،a با 1 و b با 2 نشان داده می شود. از سوی دیگر ، رشته ورودی abbab با اعداد طبیعی نمایش داده می شود.

$$44312=(((1.14+2).14+2).14+1).14+2$$
$$=1.14^4+2.14^3+2.14^2+1.14^1+2.14^0$$

٤-٣ غیر قطعیت در مقابل قطعیت

مبدل های با حالتهای متناهی غیرقطعی می توانند بعضی از توابع را محاسبه کنند که،

مبدل های با حالت متناهی قطعی نمی توانند محاسبه نمایند. به طور مشابه ، ماشین های پشته ای غیرقطعی می توانند بعضی زبانهایی را بپذیرند که ماشینهای پشته ای قطعی نمی توانند آنها را بپذیرند. به هر حال زبانهایی که بوسیله زبانهای با ماشین های حالت متناهی غیر قطعی پذیرفته می شوند بوسیله ماشینهای با حالت متناهی قطعی نیز پذیرفته می شوند .

از غیر قطعیت به قطعیت :

تئوری ذیل غیر قطعیت به قطعیت را در مبدل های تورینگ به هم مربوط می کند.

قضیه ٤-٣-١ . هر مبدل تورینگ مانند M_1 یک تناظر با مبدل تورینگ قطعی مانند M_2 دارد به طوریکه :

a. M_2 همان ورودی M_1 را می پذیرد که در نتیجه : $L(M_1)=L(M_2)$

b. M_2 دقیقاً در همان ورودی هایی متوقف می شود که M_1 روی آنها توقف دارد .

c. M_2 یک خروجی مانند y به ازای یک ورودی داده شده دارد اگر M_1 بتواند y را به ازای همان ورودی به خروجی بفرستد که در نتیجه : $R(M_2)<R(M_1)$.

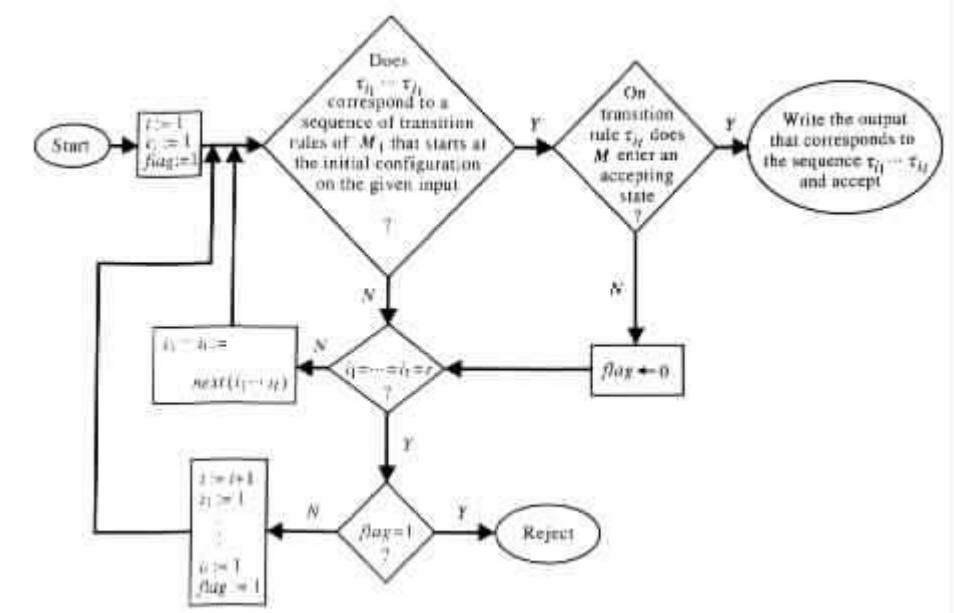
اثبات : مبدل تورینگ دارای m نوار کار کمکی مانند M_1 و هر ورودی مانند x برای M_1 را در نظر بگیرید. $\tau_1,\dots\tau_r$ قواعدانتقال M_1 را مشخص می نمایند. $C_{i_1\cdots i_t}$ حالتهایی را مشخص می نماید که M_1 در ورودی x از پیکربندی اولیه است به آن می رسد در میان دنباله ای از حرکت ها که دنباله ای از قواعد انتقال $\tau_{i_1}\dots\tau_{i_t}$ را به کار می برند .

اگر هیچ چنین دنباله ای از حرکت ها ممکن نباشد ، سپس $C_{i_1\cdots i_t}$ فرض می شوند که یک پیکربندی تعریف نشده را مشخص می نماید . یک مبدل تورینگ مطلوب مانند M_2 می تواند مبدل تورینگ با $m+1$ نوار کار کمکی به صورت ذیل باشد .

M_2 روی ورودی داده شده x ،در طول دنباله $C\varepsilon, C_1, C_2, \dots, C_r, C_{11}, C_{12}, \dots, C_{rr}, C_{111}, C_{112}, \dots, C_{rrr}, \dots$ برای یک پیکربندی پذیرش از M_1 روی ورودی x جستجو می نماید .یک محاسبه از M_2 روی

ورودی x به صورت ذیل پیش می رود .

M_2 رشته های $\alpha = (i_1 \cdots i_t)$ در $\{1$ و.....و $r\}^*$ را در اولین نوار کار کمکی ،یکی یکی ،با ترتیب استانداردتا زمان برقراری یکی از دو شرطذیل لیست می نمایند. (به فلو چارت شکل 4.1.3 نگاه کنید) .



شکل ٤-٣-١ شبیه سازی یک مبدل تورینگ غیر قطعی مانند M_1 بوسیله مبدل تورینگ قطعی مانند M_2 .

a. M_2 رشته $\alpha = (i_1 \cdots i_t)$ را در $\{1$ و.....و $r\}^*$ تعریف می نماید به طوریکه $C_{i_1 \cdots i_t}$ یک پیکربندی پذیرش از M_1 روی ورودی x باشد.چنین پیکربندی با محاسبه پذیرش

$C_\varepsilon \phi C_{i_1} \phi \ldots \phi C_{i_1 \cdots i_t}$ در M_1 روی ورودی x متناظر می شود. در چنین

مواردی، M_2 همان خروجی حالت پذیرش را تولید کرده و در یک پیکربندی پذیرش متوقف می شود.

از M_2 می فهمید که آیا پیکربندی $C_{i_1 \cdots i_t}$ یک پیکربندی تعریف شده بوسیله پیمایش رشته $i_1 \ldots i_t$ در حالیکه سعی می کند تا یک دنباله حرکات $C_\varepsilon \vdash C_{i_1} \vdash \ldots \vdash C_{i_1 \cdots i_t}$ از M_1 را روی ورودی X دنبال کند است یا نه .
در طول پیمایش M_2 از خروجی M_1 چشم پوشی می نماید . M_2 از هد ورودی برای دنبال کردن حرکات هد ورودی M_1 استفاده می نماید.
M_2 از کنترل حالت متناهی آن برای ثبت حالتهای M_1 استفاده می نماید. همچنین M_2 از m نوار کار کمکی برای دنبال کردن تغییرات در نوارهای کار کمکی M_1 استفاده می نماید.

M_2 خروجی M_1 را که با رشته داده شده $i_1 \cdots i_t$ تناظر دارد و بوسیله پیمایش رشته و گرفتن خروجی متعلق به آن با تناظر ترتیب قواعد انتقال $\tau_1, \ldots \tau_r$، تعریف می نماید.

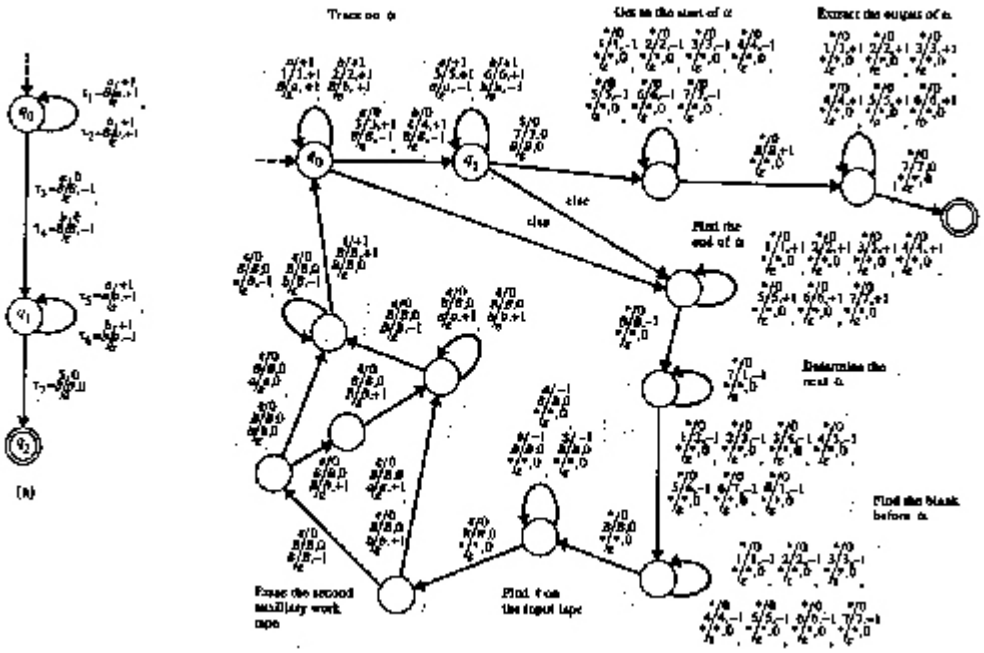
b. M_2 یک tرا تعریف می نماید که $C_{i_1 \cdots i_t}$ یک پیکربندی تعریف نشده برای تمام رشته های $\alpha = (i_1 \cdots i_t)$ در* {r و.....و1} است . در چنین مواردی ، M_2 در یک پیکربندی عدم پذیرش متوقف می شود . M_2 با بررسی اینکه M_1 هیچ دنباله حرکاتی به فرم $C_\varepsilon \vdash C_{i_1} \vdash \ldots \vdash C_{i_1 \cdots i_t}$ در ورودی xدارد تعیین می کند که یک رشته داده شده $i_1 \cdots i_t$ متناظر با با یک پیکربندی تعریف نشده $C_{i_1 \cdots i_t}$است یا نه. این بررسی با یک پیشروی مشابه به حالت تشریح شده در (a) انجام می شود.
M_2 از یک flag (پرچم) در کنترل حالت متناهی برای تعیین وجود یک t استفاده می نماید طوریکه $C_{i_1 \cdots i_t}$ پیکربندی های تعریف نشده برای همه $i_1 \cdots i_t$ در

* {r و.....و1} هستند. M_2 پرچم را با ۱ علامت گذاری می نماید هرگاه t با یک واحد افزایش یافته باشد. همینطور پرچم را با ۰ علامت گذاری می نماید هرگاه رشته ای مانند $i_1 \cdots i_t$ تعریف شده باشد طوریکه $C_{i_1 \cdots i_t}$ یک پیکربندی تعریف شده باشد. M_2 برقرار بودن خصوصیت را زمانی تعیین می کند که t در یک پرچم حاوی مقدار یک افزایش یافته باشد.

مثال ٤-٣-١ . فرض کنید M_1 یک مبدل تورینگ غیر قطعی دارای یک نوار کمکی باشد،که در شکل (a)٤-٣-٢ نمایش داده شده است.

شکل ٤-٣-٢ (a) M یک مبدل تورینگ غیرقطعی . (b) مبدل تورینگ قطعی که همان توابع M را محاسبه می نماید.

!



M_1 رابطه $\{(ww^{rev}, w^{rev}) | w \text{ is in } \{a, b\}+\}$ را محاسبه می کند. با ورودی abba مبدل تورینگ M_1 یک محاسبه پذیرش مانند $C_\varepsilon \vdash C_1 \vdash C_{12} \vdash C_{124} \vdash C_{1246} \vdash C_{12465} \vdash C_{124657}$ دارد که با ترتیب قواعد انتقال $\tau_1\tau_2\tau_4\tau_6\tau_5\tau_7$ متناظر است.

فرض نمایید M_2 مبدل تورینگ قطعی در شکل 4.3.2(b) باشد. M_2 همان توابع M_1 را محاسبه می کنند و مشابه با مبدل تورینگ در اثبات نظریه 4.3.1 است. تفاوت اصلی در آنست که M_2 در اینجا فقط در محاسبات پذیرش آن متوقف می شود.
با ورود abba مبدل تورینگ M_2 رشته ها $\{1, \ldots, 7\}^*$ را در اولین نوارکار کمکی یک به یک در ترتیب استاندارد لیست نماید. مبدل تورینگ M_2 چک می نماید که آیا هر یک از این رشته های $\alpha = i_1 \cdots i_t$ یک محاسبه پذیرش $C_\varepsilon \vdash C_{i_1} \vdash \cdots \vdash C_{i_1 \cdots i_t}$ از M_1 را تعریف می نماید یا نه.

مبدل تورینگ M_2 مشخص می نماید که هیچ یک از رشته های "", "1", ..., "7", "11", ..., "77", ..., "111111", ..., "124656" با نمایش دنباله های

$\varepsilon, \tau_1, \ldots, \tau_7, \tau_1\tau_1, \ldots, \tau_7\tau_7, \ldots, \tau_1\tau_1\tau_1\tau_1\tau_1\tau_1, \ldots, \tau_1\tau_2\tau_4\tau_6\tau_5\tau_6$،

به ترتیب با یک محاسبه پذیرش M_1 روی ورودی abba متناظر است. سپس M_2 تعیین می کند که رشته "124657" که نمایش دنباله $\tau_1\tau_2\tau_4\tau_6\tau_5\tau_7$ است،با یک محاسبه پذیرش در M_1 با ورودی abba مطابقت دارد.با این تعریف مبدل تورینگ M_2 خروجی این محاسبه را می نویسد و در یک حالت پذیرش متوقف می شود.
M_2 از مولفه "Trace on a" برای چک کردن،بوسیله گشتن در ورودی داده شده استفاده می نماید که در اینجا دنباله ای از حرکات M_1 به فرم

$C_\varepsilon \phi C_{i_1} \phi \ldots \phi C_{i_1 \cdots i_t}$ برای $\alpha = i_1 \cdots i_t$ که در اولین نوارکار کمکی M2 ذخیره شده است وجود دارد. چنین دنباله ای از حرکات متناظر با یک محاسبه پذیرش M_1 است اگروتنهااگر $\tau_{i_t} = \tau_7$. مولفه "Trace on a" در اصل تغییری در M_1 است که دنباله قواعد انتقال دیکته شده بوسیله محتوای اولین نوارکار کمکی M_2را دنبال کند.

M_2 از مولفه های

"Find the end of a,""Determine the next a,""Find the blank before a," "Find ¢ on the input tape," and "Erase the second auxiliary work tape"

برای آماده ساختن خودشان برای رسیدگی a بعدی از ترتیب استاندارد مجموعه {1, . . . , 7}* استفاده می نماید.

مبدل های تورینگ با دو نوار کار معین :

گزاره ذیل ایجاب می نماید که نظریه 4.3.1 زمانی که M_2 مبدل تورینگ قطعی با ۲نوار کار کمکی است نیز برقرار باشد.

گزاره 4.3.1 . هر مبدل تورینگ قطعی مانند M_1 یک مبدل تورینگ قطعی مانند $M_۲$ با دونوار کار معین دارد که با آن برابر است.

اثبات. هر مبدل تورینگ مانند M_1 با m نوار کار کمکی را در نظر بگیرید. در یک ورودی مانند x ،مبدل تورینگ M_2 محاسبه $C_\varepsilon \phi C_1 \phi C_2 \ldots$ را شبیه سازی می نماید که M_1 روی ورودی x انجام می دهد. M_2 با ثبت پیکربندی اولیه $C_0 = (¢q_0x\$, q_0, \ldots, q_0, \varepsilon)$ از M_1روی ورودی x شروع به کار می کند . M_2 مکرراًپیکربندی های ثبت شده C_i از M_1 را با پیکربندی بعدی C_{i+1} از محاسبات شبیه سازی شده جایگزین می نماید.

M_2 به محض اینکه به پیکربندی توقف در M_1 می رسد ،توقف می کند. M_2 روی یک محاسبه پذیرش توقف می نماید اگرو تنهااگر معلوم شود که M_1 همین کار راانجام داده است.

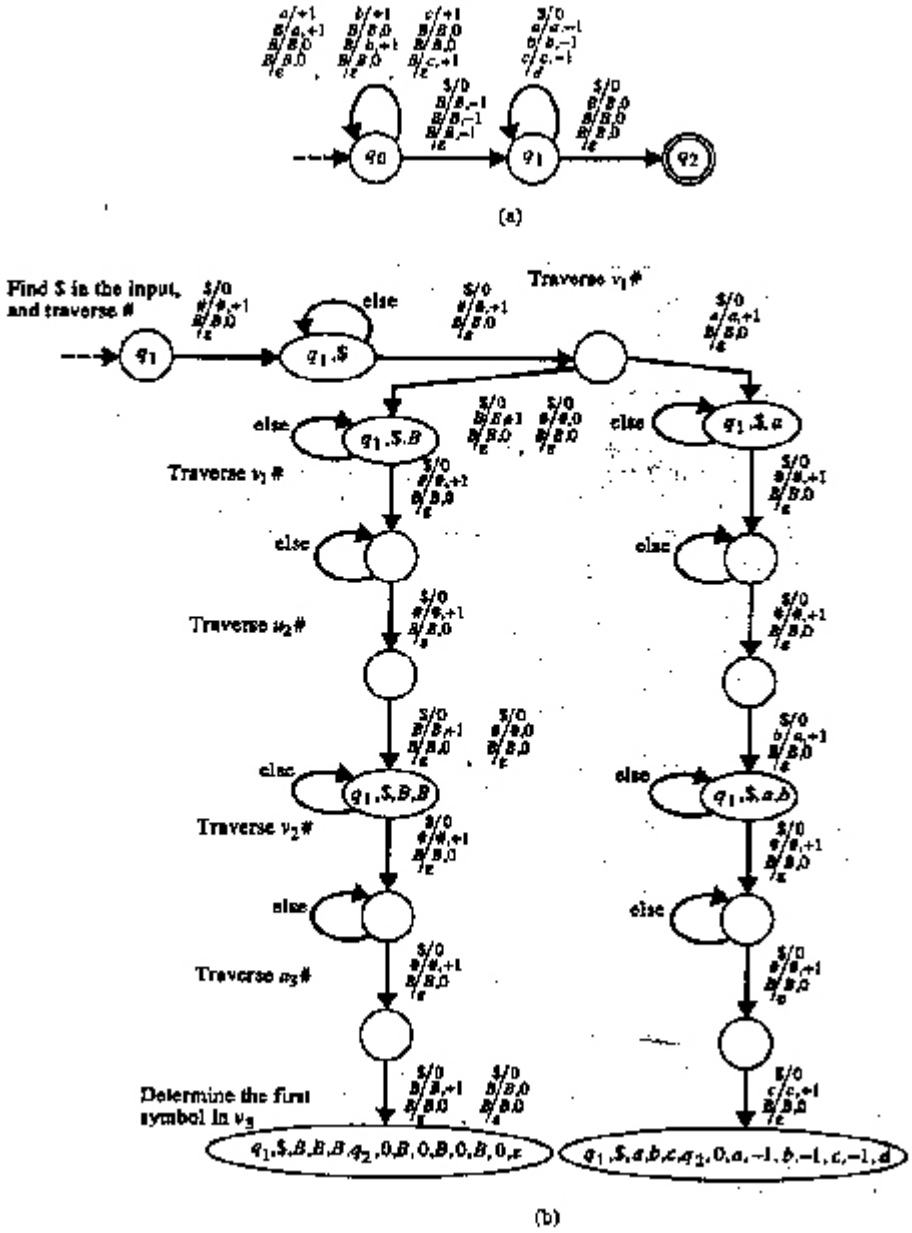
M_2 یک پیکربندی $C_i = (u\ q_l v\ , u_1 q_l v_1, \cdots, u_m q_l v_m, w)$ از M_1 را بوسیله روش ذیل ثبت می نماید. حالتq در کنترل حالت متناهی M_2 ذخیره شده است. موقعیت هد ورودی M_1 بوسیله محل ورودی M_2 ثبت شده است. خروجی W ازM_1 در نوار خروجی M_2 ثبت شده است. چندتایی $(u_1, v_1, \cdots, u_m, v_m)$به عنوان یک رشته ای به فرم $\#u_1\#v_1\#\ldots\#u_m\#v_m\#$ در یک نوار کار معین M_2 ثبت می گردد. # به معنای سمبل جدید است.

این چندتایی زمانیکه i زوج است در اولین نوارکار معین M_2 و زمانیکه فرد است در دومین نوارکارمعین ذخیره می شود.

مبدل تورینگ M_2 یک محاسبه را باقرار دادن یک رشته با $(2m + 1)$ سمبل # در اولین نوارکارمعین آغاز می نماید. یک چنین رشته ای وضعیت $u_1 = v_1 = \ldots = u_m = v_m = \varepsilon$ را نمایش می دهند. M_2 قواعد انتقال $(q, a, b_1, b_2, \ldots, b_m, p, d_0, c_1, d_1, c_2, d_2, \ldots, c_m, d_m, \rho)$ را تعریف می نماید تا در یک حرکت داده شده با گرفتن q از کنترل حالت متناهی،aاز نوار ورودی و $b_1 \cdots b_m$ از نوارکارمعین که $\#u_1\#v_1\#\ldots\#u_m\#v_m\#$ را ثبت می نماید استفاده می شود.

مثال ٤-۳-۲ فرض نمایید M_1 مبدل تورینگی باشد که قواعد انتقال آن در شکل (a) ٤-۳-۳ داده شده باشد.

!Error



شکل ٤-٣-٣ قسمتی از مبدل تورینگ در بخش (b) را تعریف می نماید که قواعد انتقال بوسیله مبدل تورینگ در بخش (a) از حالت q_1 استفاده شده.

مبدل تورینگ M_2 به دلیل قضیه ٤-۳-۱ می تواند از یک بخش D مانند شکل ٤-۳-b۳. برای تعریف قواعد انتقالی که M_1 در حرکت از حالت q_1 بکار می برد استفاده می نماید.

D فرض می کند که M_1 در پیکربندی $(u\ q_1 v\ , u_1 q_1 v_1, \ldots, u_m q_1 v_m, w)$ است،M_2 در حالت q_1است که، $\#u_1\#v_1\#\ldots\#u_m\#v_m\#$ در اولین نوارکارمعین ذخیره شده است و هد اولین نوارکارمعین بر روی اولین سمبل قرار گرفته است.

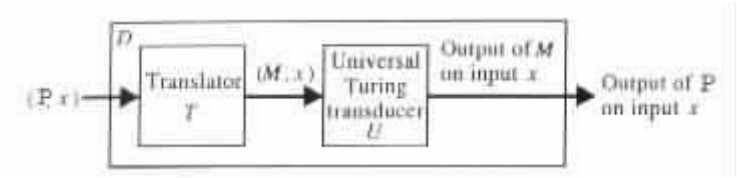
از آنجایی که مبدل های تورینگ می توانند روابطی را محاسبه نمایند که تابع نیستند،نتیجه می گیریم که مبدل های تورینگ غیرقطعی قدرت تعریف بیشتری از مبدل های تورینگ قطعی دارند. به هر حال نظریه ٤-۳-۱ برای ماشین های تورینگ دلالت نمی نماید. در واقع نظریه ٤-۳-۱ به اتفاق قضیه ٤-۳-۱ نتیجه ذیل را می دهند.

نتیجه ٤-۳-۱ . یک تابع بوسیله مبدل تورینگ محاسبه پذیر (محاسبه پذیر پاره ای) است اگرو تنهااگر آن تابع با یک مبدل تورینگ قطعی با دو نوار کارمعین محاسبه پذیر(محاسبه پذیر پاره ای) باشد.

٤,٤ مبدلهای تورینگ عمومی

برنامه ها نوشته می شوند تا فرمان چگونه حل کردن مسائل را به ماشینهای محاسباتی بدهند.برنامه P برای ماشین A قابل اجرا است اگر A بتواند محاسبات P را با هر ورودی x برای P انجام دهد.

درخیلی از موارد یک ماشین محاسباتی ساده می تواند بیش از یک برنامه را اجرا کند و همچنین می تواند برای محاسبه توابع متفاوت برنامه ریزی شود. اگرچه از بحث قبل واضح نیست که یک ماشین محاسباتی تا چه حد می تواند عمومی باشد.قضیه ی زیر بهمراه تز چرچ بیانگر این نکته هستند که ماشینهایی وجود دارند که می توانند برای محاسبه توابع محاسبه پذیر برنامه ریزی شوند. یک مثال ماشین محاسباتی D است که از دو بخش مترجم T و مبدل استاندارد تورینگ U تشککیل شده است که به قرار زیر است.(شکل ٤-٤-۱)



شکل ٤-٤-١ ماشین محاسباتی برنامه پذیر D

U یک مبدل قطعی تورینگ است که می تواند هر مبدل قطعی تورینگ M را اجرا کند. U با هر زوج مرتب (M, x) بعنوان ورودی ، محاسبات M را روی ورودی x انجام می دهد.

(رجوع شود به اثبات قضیه ٤-٤-١)

T یک مبدل قطعی تورینگ است که ورودی آن زوج مرتب (P, x) است. P برنامه ایست که با یک زبان برنامه نویسی ثابت نوشته شده و x یک ورودی برای P است. T برای یک ورودی (P, x) خروجی x را بهمراه تورینگ قطعی تبدیل یافته M که معادل با P است را خواهد داشت. در حالت خاص اگر P به زبان ماشین نوشته شده باشد P و M بعنوان ورودی و خروجی یکسان هستند ولی اگر P به زبان های سطح بالا نوشته شده باشد M یک ترجمه از P است که برای U و توسط ماشین T انجام می گیرد.

وقتی زوج مرتب (P, x) بعنوان ورودی وارد ماشین قطعی تورینگ D می شوند ابتدا ماشین زوج مرتب را به T میدهد سپس (M, x) بعنوان ورودی وارد مبدل U شده و در نهایت خروجی دلخواه از P برای ورودی x بدست می آید.

تعریف یک مبدل تورینگ عمومی U یک مبدل قطعی تورینگ است که روی هر زوج مرتب (M, x) ،یک مبدل تورینگ M و ورودی x برای M ، از T به U عملیات M را روی ورودی x اعمال می کند.ماشین ورودی هایی که به فرم زوج مرتب (M, x) نیستند را بر می گرداند.برای ماشین تورینگ عمومی تعریف مشابهی ارائه میشود.

باید به این نکته توجه کرد که زوج مرتب (M, x) بصورت فرم کد شده برای مبدل

قطعی تورینگ تعریف می شوند و خروجی مبدل نیز بصورت کد شده ایی از M برروی ورودی x است .برای آسانتر شدن مطلب، بخش کد گذاری حذف شده تا مشکل گیج کنندگی بوجود نیاید.در حالت کلی،بغیر از حالات خاص، فرم استاندارد باینری برای کد گذاری معرفی می شود.

نمایشی از یک مبدل تورینگ

طبق آنچه در پی می آید، یک رشته ، نمایش استاندارد باینری یک مبدل تورینگ $M = < Q, \Sigma, \Gamma, \Delta, \delta, q_0, B, F >$ است اگراگر مساوی با $E(M)$ باشد که E مطابق زیر به صورت بازگشتی تعریف می شود :

۱) $E(M) = E(F)01E(\delta)$

۲) $E(F) = E(p_1)...E(p_k)$ برای چند تایی مرتب $\{p_1, \ldots, p_k\}$ که حالت هایی متعلق به F هستند.

۳) $E(B)=0$ نمایش باینری برای سمبل خالی است .

٤) $E(\delta) = E(\tau_1)01E(\tau_2)01...01E(\tau_r)01$ برای ترتیب $\tau_1,...,\tau_r$ که قواعد انتقالی از δ هستند.

۵) $E(\tau) = E(q)E(a)E(b_1)\ldots E(b_m)E(p)E(d_0)E(c_1)E(d_1)\ldots E(c_m)\,E(d_m)E(\rho)$ برای هر $\tau = (q, a, b_1, \ldots, b_m, p, d_0, c_1, d_1, \ldots, c_m, d_m, \rho)$ متعلق به δ.

٦) $E(d) = 011 \text{ for } d = -1,\ E(d) = 0111 \text{ for } d = 0,\ E(d) = 01111 \text{ for } d = +1$ و $E(\rho) = 0$ برای هر خروجی $\rho = \epsilon$

۷) $E(q_i) = 01^{i+2}$ برای هر q_i در Q ، که ترتیب $q_1,...,q_s$ حالت هایی متعلق به Q است.در ایت ترتیب فرض اینست که با حالت اولیه q_0 در ابتدا قرار می گیرد.

۸) $E(e_i) = 01^{i+2}$ برای هر e_i متعلق به $(\Sigma \cup \Gamma \cup \Delta \cup \{¢,\$\}) \cup (B)$ و مجموعه ترتیبی $\{e_1, \ldots, e_t\}$ که $e_1 = ¢$ و $e_2 = \$$.

برای درک بهتر، ما می بینیم که E نمایش باینری را برای نشانه های الفبایی مبدل

تورینگ مهیا می کند، یک نمایش باینری برای حالت های از مبدل و یک نمایش باینری برای حرکت های ممکن هدها فراهم می کند.پس نمایشی برای دنباله ای از چنین موجودیت هایی ، با اتصال نمایش هر کدام به هم ، به دست می آید . رشته 01 برای جداسازی استفاده می شود که باعث دوری از ابهام می شود.

با توجه به تعریف، یک مبدل تورینگ تعداد متناهی از نمایش های باینری استاندارد می تواند داشته باشد.

هر کدام از این نمایش ها وابسته به ترتیب انتخابی برای حالات Q میباشد، ترتیب انتخابی برای نشانه های متعلق به $(\Sigma \cup \Gamma \cup \Delta \cup \{¢, \$\}) - \{B\}$ ، ترتیب انتخابی برای حالات F ، و ترتیب انتخابی برای قوانین انتقال δ می باشد.

از طرف دیگر مبدل های تورینگ مختلف می توانند نمایش های استاندارد باینری یکسان داشته باشند اگرآنها یکریخت باشند، یعنی، اگر مبدل ها به جز در نام حالت هایشان و نمادهای الفبایشان مساوی باشند.

مثال ٤-٤-١ اگر M یک مبدل تورینگ که دیاگرام انتقال آن در شکل ٤-١-٣ داده شده باشد، آنگاه E(M) می تواند نمایش استاندارد باینری زیر را داشته باشد که $E(q_0) = 0$.

Error!

$$E(q_4)11E(q_0, a, B, q_1, +1, a, +1, a)11 \cdots 11E(q_3, \$, B, q_4, 1, B, 1, ¢)11 =$$
$$E(q_4)11E(q_0)E(a)E(B)E(q_1) \cdots E(q_3)E(\$)E(B)E(q_4)E(1)E(B)E(1)E(¢)11$$

به ازای

E(q0)=011, E(q1)=0111, E(q2)=01111, E(q3)=011111, E(q4)=0111111, E(B)=0, E(¢)=011, E($)=0111, E(a)=01111, …

0 و 00 مثالهایی از رشته های باینری هستند که نمایش استاندارد باینری برای مبدلهای تورینگ نیستند.

رشته ی

!Error

$$a = 01^{4}0101^{2}01^{4}001^{2}01^{4}01^{4}01^{4}01^{6}0101^{2}01^{5}001^{3}01^{3}0$$
$$01^{2}00101^{3}01^{4}01^{4}01^{3}01^{4}001^{2}001101^{3}01^{3}001^{4}01^{4}001^{3}001$$

یک مبدل تورینگ با یک حالت پذیرش با چهار قانون انتقال که فقط اولین قانون انتقال آن خروجی غیر تهی دارد، را تصویر می کند، این مبدل تورینگ یک نوار کار معین دارد.

$E(M)01E(¢x\$)$ یک نمایش استاندارد باینری از (M,x) است بطوریکه $E(¢x\$) = E(¢)E(a_1)...E(a_n)E(\$)$ وقتی $x = a_1...a_n$.

٤-٤ مبدل تورینگ عمومی

در زیر یک مثال از مبدل تورینگ عمومی آمده است :

قضیه ٤-٤-١ یک مبدل تورینگ عمومی U وجود دارد.

اثبات :U می تواند یک مبدل تورینگ با دو نوار کار معین باشد شبیه M_2 در اثبات قضیه ٤-٣-١ بطور واضحتر، U تمامی محاسبا ت را با چک کردن ورودی های به شکل زوج مرتب (M,x) شروع می کند،

مبدل قطعی تورینگ برای $M = <Q,\Sigma,\Gamma,\Delta,\delta, q_0, B, F>$ و ورودی های x برای M ، که به نمایش استاندارد باینری داده شده، اگر ورودی ها به این شکل نباشند، U در یک حالت پذیرش ناموفق متوقف می شود.

درهر صورت، اگر ورودی به چنین شکلی باشد، U محاسبا ت M را بروی x شبیه سازی می کند.U ، مثل M_2 ، از دو نوار کار معین برای نگداری محتویات نوار کار M استفاده می کند.همچنین U می تواند از چندین نوار برای نگداری جزئیات حالت ها و مکان هدهایورودی M استفاده کند. بطور واضحتر، یک مبدل استاندارد تورینگ U برای ذخیره ساز ی $(uqv, u_1qv_1, \ldots, u_mqv_m, w)$ از M ، $\#E(q)\#|u|\#E(u_1)\#E(v_1)\#...\#E(u_m)\#E(v_m)\#$ را در نوار وضعیت و E(w) را در نوار خروجی ذخیره می کند.

برای تعیین قاعده انتقال $(q, a, b1, \ldots, bm, p, d0, c1, d1, \ldots, c_m, d_m, \rho)$ که M در یک حرکت شبیه سازی شده استفاده می کند، U حالات q و سمبل $a, b_1, ..., b_m$ را بدست می آورد.U رشته ی E(q)E(a)E(b1)...E(bm) را در نوار کار ثبت می کند که در نتیجه پیکربندی از M نگه داشته نمی شود. سپس U $p,d_0,c_1,d_1,\ldots,c_m,d_m,\rho$، را توسط جستجو E(m) ، برای زیر رشته هایی به فرم 01E(q)E(a)E(b1)... E(bm) تعیین می کند.

٤-٥:تصمیم ناپذیری

متناهی بودن حافظه و محدودیت دسترسی به آنها ، به ترتیب، باعث کم شدن توانایی های مبدلهای قطعی و مبدلهای Pushdown شده است. در مبدلهای تورینگ نبود محدودیت های روی حافظه پر اهمیت است، زیراآنها می توانند همه داده ها را پاک کنند (واطلاعات جدید را وارد کنند) که در این مورد قدرت بیشتری نسبت به بقیه پیدا می کنند.

اما هنوز زبانهایی وجود دارند که مبدلهای تورینگ توانایی پذیرش و یا تصمیم گیری در مورد آنها را ندارند.یک تصویر ذهنی از این موضوع، هر ماشین تورینگ با یک زبان تعریف می شود و خود آن زبان با رشته ها. در نتیجه زبانهای زیادی برای ماشین های تورینگ هست. بخصوص، هر زبان روی الفبای Σ زیر مجموعه ای از Σ^* است که همه ی مجموعه های روی Σ برابرست با 2^{Σ^*} ، که بینهایت زیاد و غیر قابل شمارش است. از طرف دیگر، تعداد ماشینهای تورینگ که روی الفبای Σ تعریف می شوند زیاد

و قابل شمارش هستند، زیرا آنها همه با رشته هایی از Σ^* قابل نمایش اند.

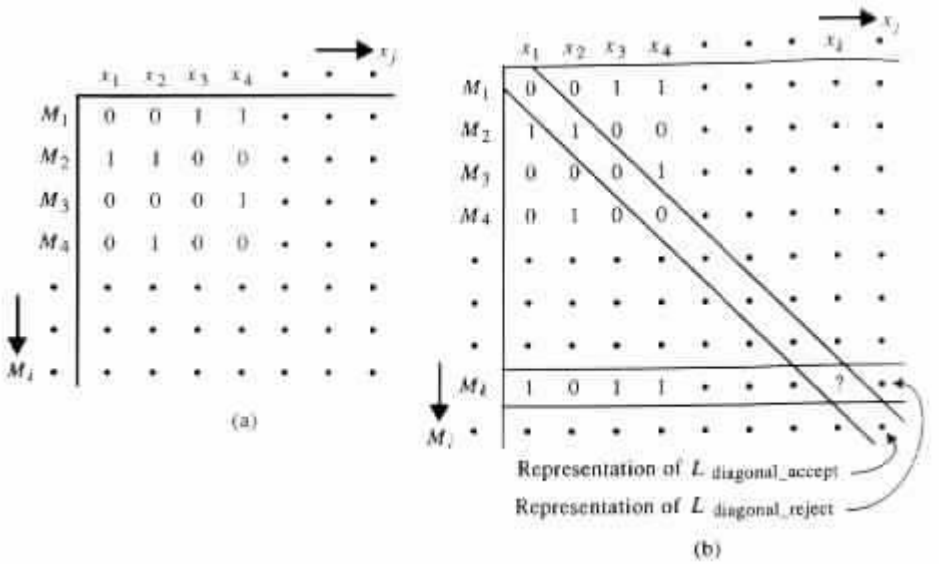
اثبات با یک راهکار کلی

اثبات قضیه بعد بطور ضمنی در ملاحظات قبلی تکرار شده چنانچه محدودیت های برنامه های قطعی در بخش ۲-٤ و محدودیت های برنامه های قطعی بازگشتی در بخش ۳,٤ بیان شده است. ما در اینجا از اصل کاهش استفاده کرده و به تناقض می رسیم. روش استفاده شده در اینجا، اثبات به روش قطری سازی تعریف می شود. که از قطرها در جداول برای انتخاب زبان ها استفاده می شود و به تناقض می رسیم.

قرارداد : در این قسمت x_i مشخص کننده i امین رشته در مجموعه مرتب شده از رشته های باینری است. بطور مشابه M_i مشخص کننده ماشین تورینگی در مجموعه مرتب شده از ماشینهای تورینگ با نمایش استاندارد باینری است که با i امین رشته نمایش باینری یکسانی دارد.(بطور کلی ماشینهای تورینگ یکریخت ، مساوی فرض می شوند.)

قضیه ٤-٥-١ زبانهایی قابل شمارش غیر بازگشتی هستند که توسط هیچ ماشین تورینگی قابل پذیرش نباشند.

اثبات : زبان L_{accept} را بصورت {ماشین تورینگ M رشته x را می پذیرد|(M,x)} در نظر بگیرید زبان L_{accept} یک نمایش جدولی بصورت T_{accept} دارد که در آن سطرها بصورت $M_1, M_2, M_3, \ldots$ و ستونها بصورت $x_1, x_2, x_3, \ldots$ نشانه گذاری می شوند و هر درایه در سطر M_i و ستون x_j 0 یا 1 است بسته به این که M_i پذیرنده x_j هست یا نه.(شکل ٤-٥-١ قسمت (a))



شکل ٤-٥-١ (a) یک جدول فرضی T_{accept} شامل x_j پذیرفته شده توسط ماشین تورینگ M_i

(b) فرم هایی از $L_{diagonal\_accept}$(پذیرش قطری) و $L_{diagonal\_reject}$(عدم پذیرش قطری) در T_{accept}

هر زبان L می تواند توسط یک بردار نمایش داده شود که در صورتی j امین ورودی 1 است که x_j در L موجود باشد و اگر x_j موجود نباشد j امین ورودی صفر است. زبان $L(M_i)$ توسط i امین سطر در T_{accept} بیان می شود.

یک راه برای اثبات این است که، زبانی پیدا کنید که با هیچ سطری در T_{accept} رابطه نداشته باشند، بنابراین نمی تواند توسط یک ماشین تورینگ پذیرفته شود. در حالات مختلف می توان زبان را از روی قطر ساخت.

قطر جدول T_{accept} نمایشی از $L_{diagonal\_accept}$(پذیرش قطری) است. $L_{diagonal\_accept}$(پذیرش قطری) بصورت $\{x_j = x\}$ و ماشین تورینگ M رشته x را می پذیرد|x} از L_{accept} نتیجه می شود. هر ماشین تورینگ که پذیرنده $L_{diagonal\_accept}$(پذیرش قطری) هست دلالت بر این دارد که داده های سطر M_k در T_{accept} با مکانهای مشابه در قطر مکمل می شود.(شکل ٤-٥-١ (b))

بطور دقیقتر، k امین شماره در سطر M_k باید مکمل k امین شماره در قطر باشد . پس k امین شماره سطر ، همچنین ، k امین شماره قطر می باشد در نتیجه متوجه می شویم که هیچ ماشین تورینگی پذیرنده زبان $L_{diagonal\_reject}$(عدم پذیرش قطری) نیست.

مباحث بالا را در قسمت بعد دسته بندی میکنیم.

در اثبات فرض کنید $L_{diagonal\_reject}$(عدم پذیرش قطری) توسط ماشین تورینگ M پذیرفته می شود پس داریم : پس اندیسی مانند k وجود دارد که $M = M_k$.

حالا رشته x_k را در نظر بگیرید، برای x_k دو حالت ممکن است:

حالت اول

x_k در $L_{diagonal\_reject}$(عدم پذیرش قطری) وجود دارد. در این حالت، فرض می شود که ماشین تورینگ M_k پذیرنده زبان $L_{diagonal\_reject}$ است که بمعنی پذیرش رشته x_k توسط M_k می باشد. متناوبا، در تعریف $L_{diagonal\_reject}$ داریم که M_k نمی تواند x_k را بپذیرد بنابراین این حالت را نمی تواند برقرار باشد.

حالت دوم

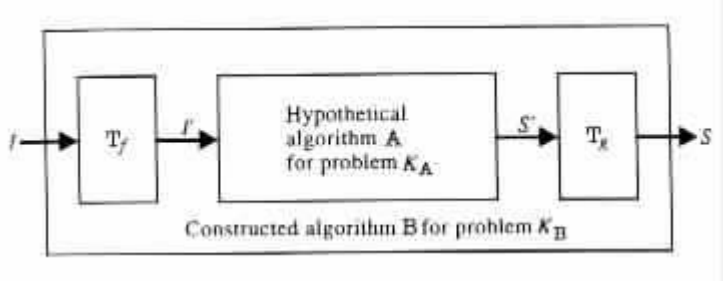
x_k در زبان $L_{diagonal\_reject}$(عدم پذیرش قطری) وجود ندارد. بطور مشابه در حالت دوم فرض می شود که M_k پذیرنده زبان $L_{diagonal\_reject}$ باشد بدین معنا که رشته x_k توسط M_k پذیرفته نمی شود و متعاقبا با توجه به تعریف $L_{diagonal\_reject}$، M_k پذیرنده x_k است بنابراین حالت دوم هم نمی تواند برقرار باشد. در نتیجه و با توجه به فرض که ماشین تورینگ M پذیرنده زبان $L_{diagonal\_reject}$ هست متوجه می شویم که یکی از دو حالت ۱ یا ۲ با ید نتیجه شود. با توجه به تز چرچ یک مسئله تصمیم گیری، قابل تصمیم گیری پاره ای است اگر و فقط اگر برای آن یک ماشین تورینگ وجود داشته باشد نمونه هایی از مسئله را که جواب آن "بله" است ، بپذیرد.به طور مشابه ، مسئله ای تصمیم پذیر است اگر و فقط اگر ماشین تورینگی وجود داشته باشد که نمونه هایی از مسئله که جواب آنها "بله " است بپذیرد و همچنین روی تمام نمونه های با پاسخ "خیر" متوقف شود .

از اثبات قضیه ٤-٥-١ و تز چرچ قضیه زیر را می توان نتیجه گیری کرد. اهمیت این قضیه اینجاست که این قضیه پایه ای برای مسئله تصمیم پذیری است که در مسائل کاهش پذیری مفید واقع می شود.

قضیه ٤-٥-٢ مسئله عضویت غیر قابل تصمیم گیری است و در عمل برای $L_{diagonal\_reject}$ به صورت پاره ای تصمیم پذیر نیست.

اثبات با کاهش پذیری

در زیر اثبات تصمیم ناپذیری را با قضیه کاهش انجام داده ایم.(رجوع شود به شکل ٤-٥-٢ و بخش ٥-١)



شکل ٤-٥-٢ کاهش از K_B به K_A

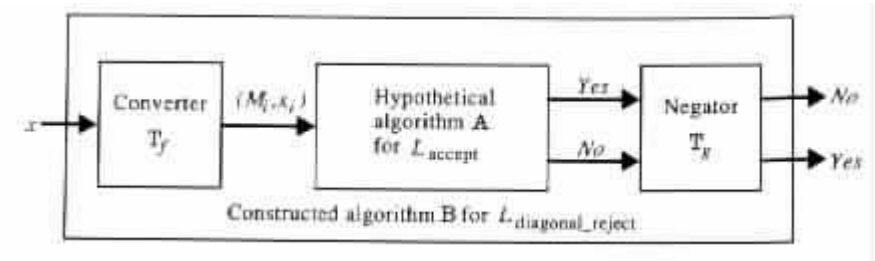
برای اثبات، فرض می کنیم مسئله K_A توسط الگوریتم A تصمیم ناپذیر است، الگوریتم های T_f و T_g را پیدا می کنیم و با الگوریتم A با هم الگوریتم B را نتیجه می گیریم. برای جواب تصمیم ناپذیری مسئله K_B در ادامه داریم، B ، با در دسترس بودن نومنه I ،از T_f استفاده میکند تا یک نمونه I' از K_A بدست بیاورد. از A روی I' استفاده می شود تا خروجی S' که A برای I' فراهم می کند بدست بیاید و آن وقت S' بصورت T_g بیان می شود تا خروجی S را از B بدست آوریم. K_A تصمیم ناپذیر است زیرا در غیر اینصورت از مسئله K_B که می دانیم تصمیم ناپذیر

است یک مسئله تصمیم پذیر نتیجه شده است.(تناقض)

اثبات قضیه زیر یک مثال برای کاهش در حل مسائل تصمیم نا پذیری است.

قضیه ٤-٥-٣ مسئله عضویت برای ماشینهای تورینگ یا برای L_{accept} تصمیم ناپذیر است.

اثبات برای اثبات فرض می کنیم که مسئله با یک الگوریتم فرضی A تصمیم پذیر است.(رجوع شود به شکل ٤-٥-٣)



شکل ٤-٥-٣ کاهش در مسئله عضویت برای $L_{diagonal\_reject}$ به مسئله عضویت برای L_{accept}

در نتیجه مسئله عضویت برای $L_{diagonal\_reject}$ می تواند توسط الگوریتم B به فرم زیر تصمیم پذیر باشد. . الگوریتم B بر روی ورودی x از مبدل T_f استفاده کرده تا خروجی زوج مرتب (M_i, x_i) را بدست آورد که $x = x_i$. T_f می تواند I را برای x در لیست مرتب شده رشته های باینری $\varepsilon, 0, 00, \ldots, x$ پیدا کندو اندیس x، را در لیست تعیین کند.

T_f می تواند M_iرا در لسیت مرتب شده رشته های باینری $\varepsilon, 0, 1, 00, \ldots$ پیدا کند تا i امین ماشین تورینگ با نمایش استاندارد باینری حاصل شود.

خروجی (M_i, x_i) از T_f توسط الگوریتم B برای قسمت A فراهم می شود، در نهایت B از T_f استفاده می کند و تعیین می کند که x در $L_{diagonal\_reject}$ هست اگر و تنها اگر A تعیین کند که x در L_{accept} نیست و x در $L_{diagonal\_reject}$ نیست اگر و تنها اگر A تعیین کند که x در L_{accept} هست.

در نتیجه می فهمیم که مسئله عضویت برای زبان $L_{diagonal\_reject}$ تصمیم ناپذیراست.
از قضیه بعدی می فهمیم که زبانهای غیر بازگشتی بطور بازگشتی قابل شمارشند.

قضیه ٤-٥-٤ مسئله عضویت برای ماشینهای تورینگ یا L_{accept} بطور پاره ای تصمیم پذیرند.

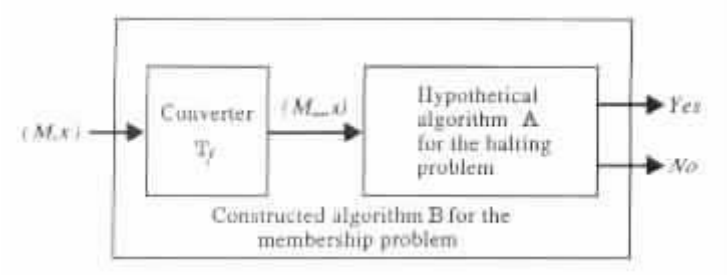
اثبات : L_{accept} توسط یک ماشین تورینگ غیر قطعی پذیرفته می شود شبیه ماشین تورینگ عمومی M_2 در اثبات قضیه ٤-٤-١. مسائل زیادی هست که تصمیم ناپذیری آنها می تواند توسط کاهش روی مسئله عضویت در ماشینهای تورینگ نشان داده شوند.

قضیه ٤-٥-٥ مسئله توقف برای ماشینهای تورینگ تصمیم ناپذیراست.

اثبات : ماشین تورینگ M با ورودی x متوقف نمی شود اگر وتنها اگر M پذیرنده x نباشد و ورودی M می تواند دنباله ای نامتناهی ازحرکات داشته باشد.

پاسخ منفی در مسئله توقف برای (M,x) بطور نمونه جواب مشابهی را در مسئله عضویت برای (M,x) بیان می کند. اگرچه جواب مثبت در مسئله توقف برای (M,x) می تواند معادل با هر دو جواب مثبت یا منفی در مسئله توقف باشد. اثبات قضیه از لحاظ نظری بیان می کند که هر ماشین تورینگ M می تواند تغییر یابد تا ورودی را در یک حالت توقف موفقیت آمیز رد نکند. با چنین تغییراتی یک جواب مثبت برای مسئله توقف برای (M,x) جواب مشابهی را برای مسئله عضویت روی (M,x) بیان می کند.

برای فهم اثبات ،فرض کنید که مسئله توقف برای ماشینهای تورینگ با الگوریتم A تصمیم ناپذیر باشد، آنگاه الگوریتم در مورد مسئله عضویت برای ماشینهای تورینگ تصمیم می گیرد، برای این منظور از یک مترجم T_f و الگوریتم A استفاده کند.(شکل ٤-٥-٤)



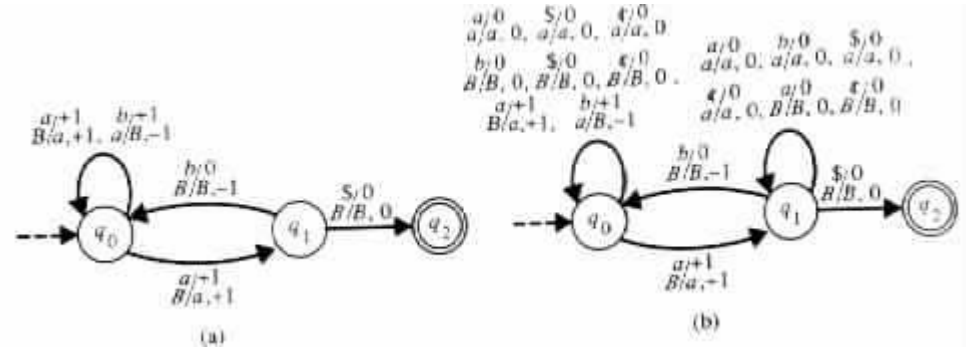
شکل ٤-٥-٤ کاهش از مسئله عضویت برای ماشینهای تورینگ به مسئله توقف برای ماشینهای تورینگ

B نمونه (M,x) را برای T_f آماده می کند. T_f با توجه به نوا کاری ماشین تورینگ M_∞ را معادل با M می سازد، که برای یک ورودی متوقف خواهد شد اگر و تنها اگر پذیرنده آن باشد. در حالت خاص، M_∞ همان M است تنها با اضافه شدن یک قانون انتقال حلقه به صورت $(q, a, b_1, \ldots, b_m, q, 0, b_1, 0, \ldots, b_m, 0)$ برای هر حالت غیر پذیرش q ، هرسمبل ورودی a و هر ترکیبی از نشانه های نوار کار $b_1, \ldots, b_m$ که در M حرکت بعدی وجود ندارد .

B ، (M,x) را به A واگذار کرده و خروجی را از A می گیرد.

مسئله بعدی با قضیه ٤-٥-٣ نشان می دهد که مسئله عضویت برای L_{accept} تصمیم پذیر است.

مثال ٤-٥-١ ماشین تورینگ را مطابق شکل ٤-٤-٥ (a) در نظر بگیرید



شکل ٤-٥-١ (a) ماشین تورینگ M (b) ماشین تورینگ M_∞ معادل با M

اگر هد ورودی و هد نوار کار هر دو a را پیمایش کنند. ماشین تورینگ M در یک حالت q_0 با پذیرش نا موفق متوقف می شود.M با ایجاد تغییراتی می تواند وارد حلقه نامتناهی ، در پیکربندی شود به طوری که ماشین تورینگ حرکتی انجام دهد که در آن پیکربندی تغییر نکند . این کار با معرفی قاعده انتقالی به فرم $(q_0, a, a, q_0, 0, a, 0)$ انجام می شود . به اثبات قضیه ٤-٥-٥ توجه کنید، شکل ماشین تورینگ M_∞ در شکل ٤-٥-١ (b) آمده است.

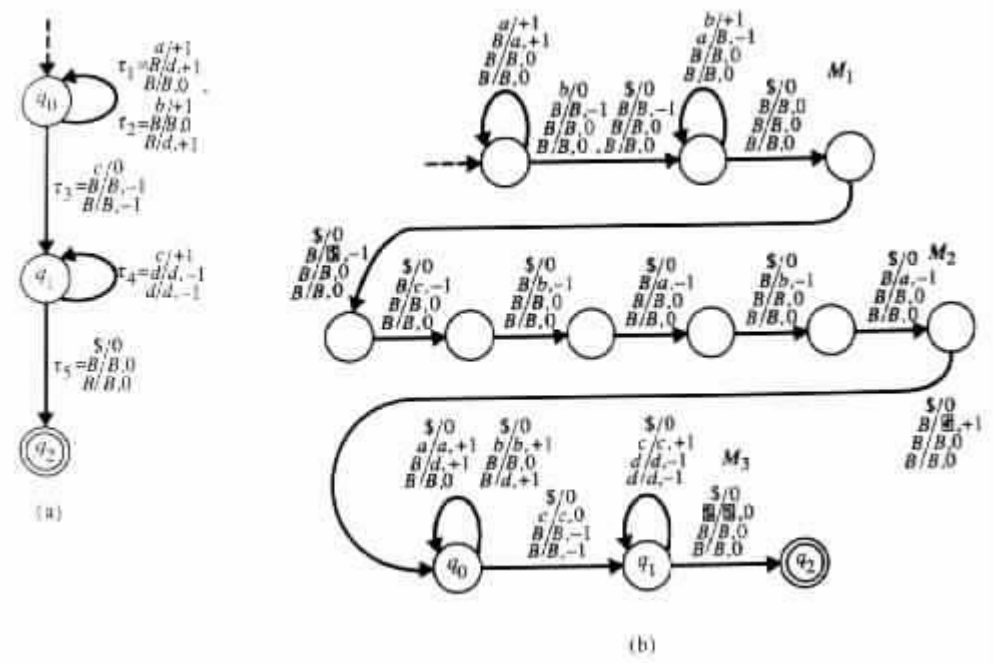
قضیه بعدی مثالهای دیگری را از اثبات تصمیم ناپذیری به وسیله کاهش نشان می دهد.

قضیه ٤-٥-٦ مسئله تصمیم گیری برای هر ماشین تورینگ که آیا پذیرنده یک زبان منظم است یا نه ،تصمیم ناپذیر است.

اثبات : برای (M,x) در مسئله عضویت برای مشینهای تورینگ مشاهده شد که ماشین تورینگ M_x برا ی فرم (M,x) اگر پذیرنده x باشد مجمو عه $\{a^i b^i | i \geq 0\}$ را میپذیردو اگر پذیرنده x نباشد مجموعه تهی را می پذیرد. بطور دقیقتر M_x روی هر ورودی w عبارت $w = a^i b^i$را برای $i \geq 0$ بررسی می کندکه اگر عبارت برای $i \geq 0$ درست نباشد w پذیرفته نمی شود و اگر نه M_x محاسبات M را بر روی ورودی x انجام می دهد. در نهایت، M_x پذیرنده w است اگر M ورودی x را بپذیرد و w را نمی پذیرد اگر x، M را نپذیرد.

نتیجتا، M_xپذیرنده یک زبان منظم است اگر وتنها اگر M پذیرنده x نباشد. در ادامه تصمیم ناپذیری برای مسئله عضویت برای ماشینهای تورینگ را خواهیم داشت.(قضیه ٤-٥-٣)

مثال ٤-٥-٢ ماشین تورینگ M را مطابق شکل ٤-٥-٦ (a) فرض کنید



شکل ٤-٥-٦ (a) ماشین تورینگ M و(b) ماشین تورینگ $M\,ababc$ متناظر با M که پذیرنده زبان $\{a^i b^i | i \geq 0\}$ است اگر و تنها اگر M پذیرنده ababc باشد .

فرض کنید x=ababc آنگاه M_x در قضیه ٤-٥-٦ می تواند به شکل ٤-٥-٦(b) باشد که M_x نسبت به M یک نوار کار بیشتر دارد که شامل سه زیرمولفه M_1, M_2, M_3 هست.

M_1 ورودی داده شده را برای $a^i b^i$ به ازای $i \geq 0$ بررسی می کند.

M_2 رشته ¢x$ را در اولین بلاک نوار ذخیره می کند.

M_3 فقط تغییرات را روی M می دهد تا آن بتواند از اول نوار کار بخواند.نشانه های ¢ و $ در اول نوار کار به ازای حروف پایانی ¢ و $ استفاده می شوند.

مثالهای بیشماری برای مسئله تصمیم ناپذیری وجوددارد.برای خیلی از این مسائل اثبات تصمیم ناپذیری خیلی دشوار است.

٤-٦ ماشینهای تورینگ و زبان نوع O

فهمیدیم که مجموعه زبانهایی که توسط اتاماتای متناهی و اتماتای پشته ای پذیرفته می شوند بترتیب زیانهای نوع سوم و دوم هستند. در دو قضیه بعدی خواهیم دید که زبانهای پذیرفته توسط ماشینهای تورینگ زبانهای نوع صفر هستند.

قضیه ٤-٦-١ هر زبان نوع ۰ یک زبان بازگشتی قابل شمارش است.

اثبات : گرامر نوع 0 ، $G=<N,\Sigma,P,S>$را در نظر بگیرید. برای G یک ماشین تورینگ دو نواره ساخته می شود که برای ورودی غیر قطعی x رشته w را که متعلق به L(G) است ، تولید می کند،و0020 را میپذیرد اگر و تنها اگر x=w .

ماشین تورینگ M_G رشته w را با یک اشتقاق در G از w از S تولید می کند.

M_G با قراردادن فرم جمله ای s در روی نوار شروع بکار می کند. سپس M_G مکررا فرم جمله ای ذخیره شده در ابتدای نوار کار را با رشته اشتقاق شده جایگزین می کند.

نوار کار دوم از یک حافظه میانی استفاده کرده و فرم جمله ای های جایگزین را مشتق می کند.فرم جمله ای جایگزین ٦ توسط یک جستجوی غیر قطعی برا ی زیررشته α بدست می اید ، بطوریکه $\alpha \to \beta$ بعنوان یک قاعده تولید در G حاصل می شود و β در ٦ جایگزین α می شود.

M_Gاز مولفه M_1برا ی کپی کردن پیشوندی از ٦ که قبل ازα بود در نوار دوم استفاده می کند.

M_Gاز مولفه M_2برا ی خواندن α از نوار کار اول و جایگذاری β در نوار کار دوم استفاده می کند

M_Gاز مولفه M_3 برا ی کپی کردن پسوندی از ٦ که بعد از α در نوار دوم آمده استفاده می کند.

M_Gاز مولفه M_4 برا ی کپی کردن فرم جمله ای که در نوار اول ایجاد شده در نوار دوم استفاده می کند. در ادامه M_Gاز مولفه M_4 برای تعیین اینکه آیا فرم جمله ای جدید در L(G) است یا نه استفاده می کند.

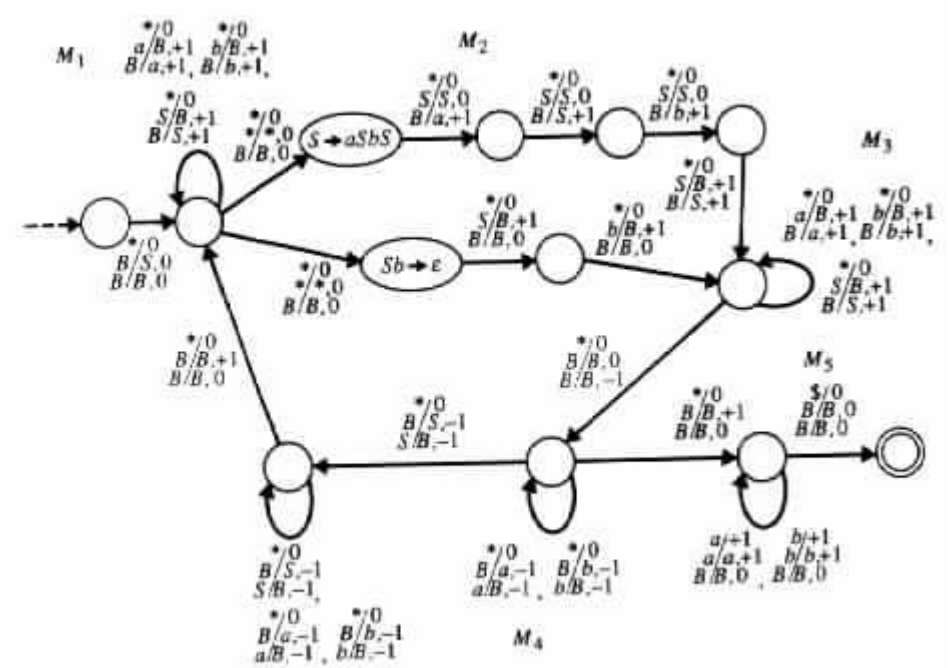
اگر w در L(G) باشد کنترل به M_5میرود وگرنه کنترل به M_1 می رود.M_Gاز

مولفه $M5$ برای تعیین اینکه آیا ورودی رشته x با رشته ی w که در نوار اول ذخیره شده برابر است یا نه استفاده می شود.

مثال ٤-٦-١ گرامر G را بشکل زیر فرض کنید

$$S \rightarrow aSbS$$
$$Sb \rightarrow \varepsilon$$

زبان L(G) که توسط ماشین تورینگ M_G پذیرفته می شود در زیر آمده است(شکل ٤-٦-١)



شکل ٤-٦-١ ماشین تورینگ M_G که برای گرامر G با قواعد تولید $S \rightarrow aSbS$ و $Sb \rightarrow \varepsilon$ ساخته شده

مولفه های $M1, M2, M3$ فرم جمله ای ذخیره شده در نوار کا اول را از چپ به راست پیمایش می کنند، همچنانکه مولفه ها نوار را پیمایش می کنند محتویات نوار پاک می شود.

مولفه $M2$ از MG از دو رشته مختلف برای قواعد تولید $S \to aSbS$ و $Sb \to \varepsilon$ استفاده می کند. دنباله ای از قواعد تولید که با قاعده $S \to aSbS$ متناظر است S ها را از نوار اول پاک می کندو aSbS را در نوار دوم ذخیره می کند.
دنباله ای از قواعد تولید که با قاعده $Sb \to \varepsilon$ متناظر است Sb هار اا‌ز نوار اول پاک کرده و در نوار دوم چیزی ذخیره نمی کند.

مولفه $M4$ فرم جمله ای که در نوار کار دوم قرار دارد را از چپ به راست پیمایش می کندو محتویات نوار در مدت پیمایش پاک می شود.
$M4$ بررسی فرم جمله ای ها را از اولین موقعیت آغاز کرده اگر عنصر خالی B در اولین موقعیت باشد نشان دهنده اینست که فرم جمله ای یک رشته از عناصر پایانی است در این حالت $M4$ کنترل را به $M5$ منتقل می کند.اگر عناصر غیر پایانی باشند نشان دهنده اینست که فرم جمله ای یک رشته از عناصر پایانی نیست در این حالت $M4$ به موقعیت دوم (بعدی) می رود.
قضیه ٤-٦-٢ هر زبان بازگشتی قابل شمارش یک زبان نوع 0 است.
اثبات : اثبات از ترکیب یک ماشین تورینگ M و گرامری که محاسبات M را نشان می دهد ساخته می شود.که گرامر G از سه گروه ساخته می شود:

هدف کاری گروه اول تعیین سه مورد می باشد:

۱) پیکربندی اولیه M برای ورودی ها
۲) بخش هایی برای هر نوار کارکمکی M که هر بخش باید شامل مکان زیر هد از نوار متناظر باشد .
۳) دنباله ای از قواعد انتقال M، دنباله قواعد انتقال باید با حالت ابتدایی شروع و در یک حالت پذیرش پایان یابد و انتقالات بین حالت ها سازگار باشد.

گروهی از قواعد تولید می توانند هر پیکربندی اولیه M ، هر بخش از نوار کار که شرایط بالا را ، و هر دنباله ای از قواعد انتقال که شرایط بالا را ارضا کنند، را تعیین

سازند.

هدف کاری گروه دوم از قواعد تولید،شبیه سازی محاسبات M است. شبیه سازی باید در یک پیکربندی تعیین شده که توسط گروه اول تعیین می شود آغاز شود. در ادامه، شبیه سازی باید بر اساس دنباله ای از قواعد انتقال و درون بخش نوار های کار کمکی تعیین شده به وسیله گروه اول ،باشد .

هدف کاری گروه سوم از قواعد تولید ،استخراج ورودی هایی ست که محاسبه پذیرش روی آنها شبیه سازی شده است و عناصر غیر پایانی را رها کرده در نتیجه گرامر رشته ای تولید می کند که ماشین تورینگ M پذیرنده آن است.

هر ماشین تورینگ $M = < Q, \Sigma, \Gamma, \delta, q0, B, F >$ را در نظر بگیرید .بدون از بین رفتن کلیت ،می توان فرض کرد که M یک ماشین تورینگ دو نواره است .(رجوع شود به قضیه ٤-۳-۱ و فرضیه ٤,۳۱) که هیچ قاعده انتقالی موجب پذیرش نشود چنانچه $N = \Gamma \cup \{ \tau \mid \tau \text{ is in } \delta \} \cup \{ [q] \mid q \text{ is in } Q \}$ {¢, $, 0, 1, 2, #, S, A, C, D, E, F, K} یک مجموعه چند گانه است که همه نشانه های آن مجزا هستند.

از M گرامر $G = < N, \Sigma, P, S >$ ساخته می شود که L(G) را تولید می کند،این کار با با دنبال کردن اشتقاق ها، پیکربندی هایی که M در آنها پذیرش دارند، انجام می شود. قواعد تولید در P به فرم زیر هستند:

۱) قواعد تولیدی برای تولید هر فرم جمله ای به شکل

$$¢0a_1 \cdots a_n\$B \cdots B1B \cdots B\#B \cdots B2B \cdots B\#\tau_{i_1} \cdots \tau_{i_t}$$

هر فرم جمله ای که این چنین باشد ، متناظر است با پیکربندی اولیه $(¢q_0a_1 \ldots a_n\$, q_0, q_0)$ از M و دنباله قوانین انتقال $\tau_{i_1} \ldots \tau_{i_t}$. ترتیبی است که قوانین انتقال دنباله ای از حالتهای سازگار را تعریف می کنند که در حالت اولیه شروع شده و در یک حالت پذیرش به پایان می رسد .

0 هد ورودی را مشخص می کند، 1 هد نوار کار اول را مشخص میکند، 2 هد

نوار کار دوم را. رشته B...B1B...B مر بوط به بخشی از نوار اول و B...B

B...B2 مربوط به بخشی از نوار دوم می باشد.

یک رشته در یک زبان، از فرم جمله ای ها قابل اشتقاق است اگر و تنها اگر به یکی از سه فرم زیر باشد:

۱)رشته ای مثل $a1...a_n$ باشد .

۲)M ، $a_1...a_n$ را در محاسباتی می پذیرد که از دنباله قوانین انتقال $\tau_{i_1}...\tau_{i_t}$ استفاده کنند.

۳)B...B i B...B متناظر با بخشی از نوار کار i ام است که برا ی محاسبه در نظر گرفته شده از M به اندازه کافی بزرگ باشد و $1 \le i \le 2$.

موقعیت i مکان اولیه هد نوار کار کمکی متناظر را مشخص می کند .

قواعد تولید به شکل زیر می باشد:

S	→	¢0A	
A	→	aA	For each input symbol a in Σ.
	→	$C	
C	→	BC	
	→	1D	
D	→	BD	
	→	#E	
E	→	BE	
	→	2F	
F	→	BF	
	→	#[q_0]	For the initial state q_0.
[q]	→	τ[p]	For each pair q and p of states, and for each transition rule τ, such that τ leaves state q and enters state p.
[q_f]	→	ϵ	For each accepting state q_f.

قوانین تولید برای عناصر غیر پایانی S و A می توانند رشته به فرم ¢0a1…an$C برای هر ورودی ممکن $a_1...a_n$ برای M تولید کنند.

قواعد تولید برای عناصر غیر پایانی C و D می توانند برای هر قسمت ممکن B...B1B...B از نوار کار اول که حاوی مکان هد متناظر است ، یک

رشته به فرم $B\ldots B\boxed{1}B\ldots B\#E$ تولید کنند.

قواعد تولید برای عناصر غیر پایانی E و F می توانند برای هر قسمت ممکن $B\ldots B\boxed{2}B\ldots B$ از نوار کار دوم که حاوی مکان هد متناظر است ،یک رشته به فرم $B\ldots B\boxed{2}B\ldots B\#[q0]$ تولید کنند. قواعد تولید برای عناصرغیر پایانی که متناظر با حالت هایی از M هستند می توانند هر دنباله ای $\tau_{i_1}\ldots\tau_{i_t}$ از قوانین انتقال M که با پیکربندی اولیه شروع و در یک پیکربندی پذیرش پایان می یابند را تولید کنند که با حالات میانی نیز سازگاری داشته باشد.

۲) قواعد تولید برای اشتقاق از فرم جمله ای های به شکل

$$\tau_{i_1}\cdots\tau_{i_{j-1}}\,u\boxed{0}v\$B\cdots B\,u_1\boxed{1}v_1\,B\cdots B\#B\cdots B\,u_2\boxed{2}v_2\,B\cdots B\#\tau_{i_j}\cdots\tau_{i_t},$$

که متناظرند با پیکربندی $\tau=(uqv\$, u_1qv_1, u_2qv_2)$ است و فرم جمله ای

$$\tau_{i_1}\cdots\tau_{i_j}\,\hat{u}\boxed{0}\hat{v}\$B\cdots B\,\hat{u}_1\boxed{1}\hat{v}_1\,B\cdots B\#B\cdots B\,\hat{u}_2\boxed{2}\hat{v}_2\,B\cdots B\#\tau_{i_{j+1}}\cdots\tau_{i_t},$$

که متناظر با پیکربندی $\hat{\tau}=(\hat{u}\hat{q}\hat{v}\$, \hat{u}_1q\hat{v}_1, \hat{u}_2q\hat{v}_2)$ است. γ و $\hat{\gamma}$ دو پیکربندی از M هستند که $\hat{\gamma}$ با یک حرکت با استفاده از قانون انتقال τ_{i_j} از γ بدست می آید.

برای هر قانون انتقال γ مجموعه ای از قواعد تولید وجود دارد:

۱) یک قاعده تولید از فرم $X\tau\to\tau X$ برای هر x متعلق به $\Sigma\cup\Gamma\cup\{¢,\$,\#\}$

۲) یک قاعده تولید از فرم $\boxed{0}\tau a\to\tau\boxed{0}a$ برای هر a متعلق به $\Sigma\cup\{¢,\$\}$ بطوریکه: γ قاعده تولیدی است که a را بدون حرکت هد پیمایش می کند.

۳) یک قاعده تولید از فرم $\boxed{0}\tau a \to \tau\boxed{0}a$ برای هر a متعلق به $\Sigma Y\{¢,\$\}$ بطوریکه: $\Im$ قاعده تولیدی است که a را با یک حرکت هد به راست پیمایش می کند.

٤) یک قاعده تولید از فرم $a\boxed{0}\tau b \to \tau\boxed{0}ab$ برای هر زوج مرتب a و b متعلق به $\Sigma Y\{¢,\$\}$ بطوریکه: $\Im$ قاعده تولیدی است که b را با یک حرکت هد به چپ پیمایش می کند.

۵) یک قاعده تولید از فرم $\boxed{i}\tau X \to \tau\boxed{i}Y$ برای هر $1 \le i \le 2$ و زوج مرتب X و Y متعلق به Γ بطوریکه: $\Im$ قاعده تولیدی است که X در I امین نوار بدون تغییر موقعیت هد با Y جایگزین می شود.

٦) یک قاعده تولید از فرم $\boxed{i}\tau X \to \tau Y\boxed{i}$ برای هر $1 \le i \le 2$ و زوج مرتب X و Y متعلق به Γ بطوریکه: $\Im$ قاعده تولیدی است که X در i امین نوار با حرکت هد به راست با Y جایگزین می شود.

۷) یک قاعده تولید از فرم $X\boxed{i}\tau Y \to \tau\boxed{i}XZ$ برای هر $1 \le i \le 2$ و سه تایی X و Y و Z متعلق به Γ بطوریکه: $\Im$ قاعده تولیدی است که Y را در i امین نوار با حرکت هد به چپ با Z جایگزین می کند.

هدف قواعد تولید (1) ، $\Im$ را از راست به چپ روی موقعیت های غیر ابتدایی $\{\boxed{0},\boxed{1},\boxed{2}\}$ منتقل می کند. بصورت فرمی از M به شکل

$$u\boxed{0}v\$B\cdots Bu_1\boxed{1}v_1B\cdots B\#B\cdots Bu_2\boxed{2}v_2B\cdots B\#,$$

$\Im$ نشانه های $\boxed{0},\boxed{1},\boxed{2}$ را با استفاده از یکی هفت حالات قسمت دوم تهیه می کند.هنگامیکه $\Im$ موقعیت های ابتدایی را تولید می کند قواعد تولید توسط یکی از هفت حالت تغییرات را روی نوا M شبیه سازی می کند وهمچنین موقعیت های ابتدایی مربوط به حالات $\Im$.

۳) قواعد تولید برای استخراج از فرم جمله ای

$$u\boxed{0}v\$B\cdots Bu_1\boxed{1}v_1B\cdots B\#B\cdots Bu_2\boxed{2}v_2B\cdots B\#,$$

که متناظر با پیکربندی پذیرش از M می شود برای ورودی که M پذیرنده آن است قواعد تولید به شکل زیر است:

$\tau_f ¢$	$\rightarrow$	K	For each transition rule τ that enters an accepting state.
τK	$\rightarrow$	K	For each transition rule τ that does not enter an accepting state.
Ka	$\rightarrow$	aK	For each input symbol a in Σ.
KX	$\rightarrow$	K	For each symbol X in $\Gamma \cup \{0, 1, 2, \$, \#\}$.
K	$\rightarrow$	ϵ	

مثال ٤-٦-٣ ماشین تورینگ M مانند شکل ٤-٥-٦(a) فرض شده است.L(M) از گرامر G بدست آمده است که مرکب از حالات زیر است:

۱) قواعد تولید که فرم جمله ای را پیدا می کنند که متناظر با پیکربندی اولیه برای M است مطابق با (a) در اثبات قضیه ٤-٦-٢.

S	$\rightarrow$	$¢0A$		D	$\rightarrow$	BD		$[q_0]$	$\rightarrow$	$\tau_1[q_0]$
A	$\rightarrow$	aA			$\rightarrow$	$\#E$			$\rightarrow$	$\tau_2[q_0]$
	$\rightarrow$	bA		E	$\rightarrow$	BE			$\rightarrow$	$\tau_3[q_1]$
	$\rightarrow$	cA			$\rightarrow$	$2F$		$[q_1]$	$\rightarrow$	$\tau_4[q_1]$
	$\rightarrow$	$\$C$		F	$\rightarrow$	BF			$\rightarrow$	$\tau_5[q_2]$
C	$\rightarrow$	BC			$\rightarrow$	$\#[q_0]$		$[q_2]$	$\rightarrow$	ϵ
	$\rightarrow$	$1D$								

۲) "انتقال" قواعد تولید که متناظر با (b,1) در اثبا ت قضیه ٤-٦-٢ است و $1 \le i \le 5$

$a\tau_i$	$\rightarrow$	$\tau_i a$		$B\tau_i$	$\rightarrow$	$\tau_i B$
$b\tau_i$	$\rightarrow$	$\tau_i b$		$¢\tau_i$	$\rightarrow$	$\tau_i ¢$
$c\tau_i$	$\rightarrow$	$\tau_i c$		$\$\tau_i$	$\rightarrow$	$\tau_i \$$
$d\tau_i$	$\rightarrow$	$\tau_i d$		$\#\tau_i$	$\rightarrow$	$\tau_i \#$

۳) "شبیه سازی" قواعد تولیدی که متناظر با (b-2,b-4) در اثبات قضیه ٤-٦-٢ است

$$
\begin{array}{llll}
[0]\tau_1 a \rightarrow \tau_1 a[0] & [0]\tau_2 b \rightarrow \tau_2 b[0] & [0]\tau_3 c \rightarrow \tau_3[0]c \\
[0]\tau_4 c \rightarrow \tau_4 c[0] & [0]\tau_5\$ \rightarrow \tau_5[0]\$ &
\end{array}
$$

۴) "شبیه سازی" قواعد تولیدی که متناظر با (b.5-b.7) در اثبات قضیه ۴-۶-۲

$$
\begin{array}{lll}
[1]\tau_1 B \rightarrow \tau_1 d[1] & [1]\tau_2 B \rightarrow \tau_2[1]B & d[1]\tau_3 B \rightarrow \tau_3[1]dB \\
B[1]\tau_3 B \rightarrow \tau_3[1]BB & d[1]\tau_4 d \rightarrow \tau_4[1]dd & B[1]\tau_4 d \rightarrow \tau_4[1]Bd \\
[1]\tau_5 B \rightarrow \tau_5[1]B & & \\
[2]\tau_1 B \rightarrow \tau_1[2]B & [2]\tau_2 B \rightarrow \tau_2 d[2] & d[2]\tau_3 B \rightarrow \tau_3[2]dB \\
B[2]\tau_3 B \rightarrow \tau_3[2]BB & d[2]\tau_4 d \rightarrow \tau_4[2]dd & B[2]\tau_4 d \rightarrow \tau_4[2]Bd \\
[2]\tau_5 B \rightarrow \tau_5[2]B & &
\end{array}
$$

۵) "استخراج" قواعد تولیدی که متناظر با (c) در اثبات قضیه ۴-۶-۲

$$
\begin{array}{lll}
\tau_5 \$ \rightarrow K & & \\
\tau_i K \rightarrow K & \text{For } i = 1, \ldots, 4. & \\
Ka \rightarrow aK & & \\
Kb \rightarrow bK & & \\
Kc \rightarrow cK & & \\
Kd \rightarrow K & & \\
KB \rightarrow K & & \\
K[i] \rightarrow K & \text{For } i = 0, 1, 2. & \\
K\$ \rightarrow K & & \\
K\# \rightarrow K & & \\
K \rightarrow \epsilon & &
\end{array}
$$

چپ ترین اشتقاق رشته abc از G به فرم زیر است :

$$
\begin{array}{lll}
S \Rightarrow & ¢\boxed{0}A & \\
\Rightarrow^* & ¢\boxed{0}abc\$C & \\
\Rightarrow^* & ¢\boxed{0}abc\$B\boxed{1}BB\#E & \\
\Rightarrow^* & ¢\boxed{0}abc\$B\boxed{1}BB\#B\boxed{2}BB\#[q_0] & \\
\Rightarrow^* & ¢\boxed{0}abc\$B\boxed{1}BB\#B\boxed{2}BB\#\tau_1\tau_2\tau_3\tau_4\tau_5 & \text{The sentential forms} \\
\Rightarrow^* & \tau_1¢a\boxed{0}bc\$Bd\boxed{1}B\#B\boxed{2}BB\#\tau_2\tau_3\tau_4\tau_5 & \text{in the derivation,} \\
\Rightarrow^* & \tau_1\tau_2¢ab\boxed{0}c\$Bd\boxed{1}B\#Bd\boxed{2}B\#\tau_3\tau_4\tau_5 & \text{that correspond to} \\
\Rightarrow^* & \tau_1\tau_2\tau_3¢ab\boxed{0}c\$B\boxed{1}dB\#B\boxed{2}dB\#\tau_4\tau_5 & \text{the configurations} \\
\Rightarrow^* & \tau_1\tau_2\tau_3\tau_4¢abc\boxed{0}\$\boxed{1}BdB\#\boxed{2}BdB\#\tau_5 & \text{in the simulated} \\
\Rightarrow^* & \tau_1\tau_2\tau_3\tau_4\tau_5¢abc\boxed{0}\$\boxed{1}BdB\#\boxed{2}BdB\# & \text{computation of } M. \\
\Rightarrow^* & \tau_1\tau_2\tau_3\tau_4K abc\boxed{0}\$\boxed{1}BdB\#\boxed{2}BdB\# & \\
\Rightarrow^* & Kabc\boxed{0}\$\boxed{1}BdB\#\boxed{2}BdB\# & \\
\Rightarrow^* & abcK\boxed{1}BdB\#\boxed{2}BdB\# & \\
\Rightarrow^* & abcK\boxed{2}BdB\# & \\
\Rightarrow^* & abcK & \\
\Rightarrow & abc &
\end{array}
$$

از قضیه ٤-٦-٢ بهمراه قضیه ٤-٥-٣ نتایج زیر گرفته می شود:

نتیجه ٤-٦-١مسئله عضویت برای گرامر نوع ۰ تصمیم ناپذیر است یابه طور معادل ،برای

{G یک گرامر نوع ۰ است و x متعلق است به L(G)|{(G,x)} تصمیم ناپذیر است .

یک گرامر وابسته به متن یک گرامر نوع ۰ است که هر قاعده تولید فرمی از $\tau_1 A \tau_2 \to \tau_1 \alpha \tau_2$ برای عنصر غیر پایانی a می باشد. بطور واضحتر، یک قاعده تولید با فرم $\tau_1 A \tau_2 \to \tau_1 \alpha \tau_2$ نمایش داده می شود که $A \to \alpha$ فقط زمانی می تواند استفاده شود که τ_1 در سمت چپ و τ_2 در سمت راستش قرار بگیرند.

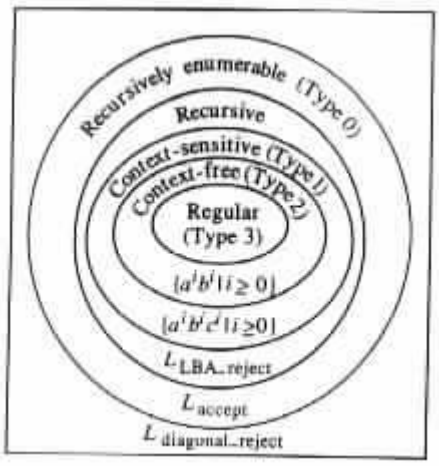
یک زبان را وابسته به متن می گوییم اگر بتواند یک گرامر وابسته به متن تولید کند. یک زبان وابسته به متن است اگر و تنها اگر زبان نوع ۱ باشد.(تمرین ٤٦٤) و اگر

و تنها اگر توسط یک اتاماتای محدود خطی پذیرفته شود.(تمرین ٤٦٤)

با تعریف و قضیه ۳-۳-۱ هر زبان مستقل از متن وابسته به متن هم هست ولی عکس این مسئله در ست نمی باشد زیرا زبانی که مستقل از متن نیست ولی وابسته به متن است.این نشان می دهد که هر زبان وابسته به متن بازگشتی است.(تمرین ٤-٤-۱) بطوریکه در مورد زبان بازگشتی

$$L_{LBA-reject} = \{ x \mid x = x_i \text{ and } M_i \text{ does not have accepting computations on input } x_i \text{ in which at most } |x_i| \text{ locations are visited in each auxiliary work tape}\}$$

داریم که وابسته به متن نیست.



شکل ٤-٦-۲ سلسله مراتبی از مجموعه زبانها . هر کدام از زبان های نشان داده شده ، به کلاس متناظر متعلق هستندولی به کلاس زیرین در سلسله مراتب متعلق نیستند .

٤-۷ مسئله تناظر پست

مسئله تناظر پست یا بطور مخفف PCP، شامل دامنه و سوالات زیر است:

دامنه:

$\{ <(x_1, y_1), \ldots, (x_k, y_k)> \mid k \geq 1$ رشته هایی $x_1, \ldots, x_k, y_1, \ldots, y_k$ روی الفبا هستند$\}$

سوال:

آیا به ازای $n \geq 1$ و اندیس های $i_1, \ldots, i_n$ برای $<(x_1, y_1), \ldots, (x_k, y_k)>$ تساوی $x_{i1} \ldots x_{in} = y_{i1} \ldots y_{in}$ وجود دارد؟ هر دنباله $i_1, \ldots, i_n$ که جواب درست را در بر گیرد شاهدی برای راه جواب مثبت یک نمونه از PCP است .

این مسئله می تواند به صورت مسئله "دومینو" به شکل زیر فرمول بندی شود :

دامنه:

$\{$یک کارت دومینو با رشته x_i در بالا و رشتهy_i در پایین $\left[\begin{matrix} x_i \\ y_i \end{matrix}\right]$ ، $1 \leq i \leq k. \} \mid k \geq 1\}$ $\left\{\left(\left[\begin{matrix} x_1 \\ y_1 \end{matrix}\right], \ldots, \left[\begin{matrix} x_k \\ y_k \end{matrix}\right]\right)\right.$

سوال:

با فرض $k \geq 1$ دسته های زیادی از کارت های

$\left[\begin{matrix} x_1 \\ y_1 \end{matrix}\right], \ldots, \left[\begin{matrix} x_k \\ y_k \end{matrix}\right]$

با تعداد زیادی کارت در هر دسته وجود دارند، که می تواند نشان دهنده ی دنباله ای $n \geq 1$ از کارت ها به ازای

$\left[\begin{matrix} x_{i_1} \\ y_{i_1} \end{matrix}\right], \ldots, \left[\begin{matrix} x_{i_n} \\ y_{i_n} \end{matrix}\right]$

برای هر دسته باشد،آیا رشته با فرم $x_{in} \ldots x_{i1}$ در بالای کارت ها مساوی رشته با فرم $y_{in} \ldots y_{i1}$ در پایین است؟

مثال ۱-۷-٤ یک جواب مثبت برای نمونه به شکل PCP $<(01, 0), (110010, 0), (1, 1111), (11, 01)>$

و به طور معادل ، برای نمونه مسئله دومینو می باشد.

چند تایی $(i_1, i_2, i_3, i_4, i_5, i_6) = (1, 3, 2, 4, 4, 3)$ شاهدی از جواب مثبت است زیرا داریم :

$x_1x_3x_2x_4x_4x_3 = y_1y_3y_2y_4y_4y_3 = 01111001011111$

جواب مثبت نیز نشاندهنده ی $(1, 3, 2, 4, 4, 3, 1, 3, 2, 4, 4, 3), (1, 3, 2, 4, 4,3,1,3,2,4,4,3,1,3,2, 4, 4, 3)$ است.از طرف دیگر ، PCP یک جواب منفی برای $<(0, 10), (01, 1)>$ دارد.

تصمیم ناپذیری در مسئله تناظر پست

مسئله تناظر پست مسئله ای کاربردی برای نشان دادن تصمیم ناپذیری خیلی از مسائل با تعریف کاهش است.

تصمیم ناپذیری آن نشان دهنده مناسب بودن آن برای شبیه سازی محاسبات ماشین تورینگ است، بطوریکه نتایج زیر در مورد گرامر نوع 0 را بطور غیر مستقیم نتیجه می دهد.

قضیه ٤-٧-١ PCP یک مسئله تصمیم ناپذیر است.

اثبات: باتوجه به نتیجه ٤,٦,١ مسئله عضویت برای گرامر های نوع 0 تصمیم ناپذیر است، بطوریکه برای نشان دادن هر نمونه مثل (G,w) از مسئله عضویت گرامر های نوع 0 کافی است داشته باشیم:

یک PCP دارای جواب مثبت است اگر و تنها اگر w در $L(G)$ وجود داشته باشد.

مثالی برای اثبات اینکه تمامی گرامر نوع 0 مثل $G = <N,\Sigma , P, S>$ و تمامی رشته

های w که در Σ^* وجود دارند با فرض اینکه # و ¢ و $ نشانه هایی متعلق به $\Sigma \cup N$ نبوده و با $I = <(x_1, y_1), \ldots, (x_k, y_k)>$ از PCP رابطه دارند که در فرم های زیر بیان می شوند.

PCP یک جواب مثبت در I دارد اگر و تنها اگر بتوان اشتقاق هایی را نتیجه گرفت که با S شروع شده و به w ختم می شوند.برای هر اشتقاق در G به فرم $S \Rightarrow \gamma_1 \Rightarrow \gamma_m \Rightarrow w \Rightarrow \ldots$ که نشان ای از جواب مثبت است بطوریکه

$$x_{i_1} \cdots x_{i_n} = y_{i_1} \cdots y_{i_n} = ¢\$\#\gamma_1\#\gamma_2\#\gamma_3\#\gamma_4 \cdots \gamma_m\#w\$$$

یا

$$x_{i_1} \cdots x_{i_n} = y_{i_1} \cdots y_{i_n} = ¢\$\#\$\#\gamma_1\#\gamma_2\#\gamma_3\#\gamma_4 \cdots \gamma_m\#w\$,$$

که به ترتیب اعداد زوج و فرد را در بر می گیرند.از طرف دیگر هر نشانه از جواب مثبت برای PCP بطوریکه I کوچکترین عدد صحیح $t \geq 1$ کوچکترین است بطوریکه $x_{i1} \ldots x_{it} = y_{i1} \ldots y_{it}$. از طرفی ،

$S \Rightarrow x_{i1}x_{i2} \ldots x_{it} = y_{i1} \ldots y_{it}\#\gamma_1\#\gamma_2\#\gamma_3\#\gamma_4 \ldots \gamma_m\#w\$$ برای بعضی اشتقاق های $S \Rightarrow \gamma_1^* \Rightarrow \gamma_2^* \Rightarrow \gamma_3^* \Rightarrow \gamma_4^* \Rightarrow w_m^* \Rightarrow w^*$ وجود دارد.

I مرکب از زوج مرتب هایی به فرم زیر است :

۱)زوج مرتبی به فرم $(¢S^{\#}, ¢)$

۲) زوج مرتبی به فرم $(\$, \#w\$)$

۳) زوج مرتبی به فرم $(X, X^{\#})$ و یک زوج مرتب به فرم $(X^{\#}, X)$ برای هر X که متعلق است به

$\Sigma \cup N \cup \{\#\}$

٤) زوج مرتبی به فرم (β, α) و یک زوج مرتب به فرم $(\beta^{\#}, \alpha)$ برای هرحالت $\alpha \to \beta$ که متعلق است به

G بطوریکه $\beta \neq \varepsilon$

٥) زوج مرتبی به فرم $(X, X\alpha)$ و یک زوج مرتب به فرم $(X^{\#}, X\alpha)$ برای

هرحالت $\alpha \rightarrow \gamma$ که متعلق است به G بطوریکه $\{\#\} \cup N \cup$

نماد آندرلاین نشان دهنده اینست که فقط نمونه هایی قابل پذیرش هستند که با زوج مرتب ($\not\subset S\#,\not\subset$) شروع و ($\$,\#w\$$) آخرین زوج مرتب آن است. زوج مرتب ($\not\subset S\#,\not\subset$) در (a) استفاده می شود تا اشتقاق ها با S شروع شوند. زوج مرتب ($\$,\#w\$$) در (b) استفاده می شود تا اشتقاق ها با w خاتمه یابند.

زوج مرتب های دیگر استفاده می شوند تا هر فرم جمله ای γ رابه فرم جمله ای γ' نتیجه دهند، بطوریکه

$\gamma \Rightarrow^* \gamma'$

تمامی جواب ها ممکن است زیرا هر زوج مرتب (x_i, y_i) توسط یک y_i برای نشان دهنده دریچه γ و x_i برای ضبط کردن جایگذاری ها برای y_i در γ' تعریف می شوند.

زوج مرتب های به فرم (β,α) و (α,α) در (d) استفاده می شوند برای جایگذاری زیر رشته های α در γ توسط زیر رشته های β در γ .

زوج مرتب هایی به فرم ($X,X\alpha$) و ($X\alpha,X$) در (e) استفاده می شوند برای جایگذاری زیر رشته α در γ توسط رشته تهی ε در γ.

دریچه وجو دارد زیرا برای هر $1 \le i_1,\ldots,i_j \le k$ رشته های و را با مشخصات زیر داریم:

۱) اگر x یک پیشوند از y باشد آنگاه x=y، وگرنه آن کمترین l است بطوریکه $x_{i1}\ldots x_{il}$ یک پیشوند مناسب از $y_{i1}..y_{il}$ هست.که در این مورد (x_{il}, y_{il}) با (v, uvv') به ازای رشته های غیر تهی v و v' برابر میشوند، که بنا بر تعریف هیچ زوج مرتبی با چنین در I موجود نیست.

۲) اگر y یک پیشوند مناسب از x باشد که جمع تعداد نمادهای # و $\not\subset$ در x یکی بیشتر از تعداد آنها در است از طرف دیگر آن کمترین $l > 1$ است برای هر $x_{il}=x_{i1}$ و $y_{il}=y_{i1}$ که با این مشخصات در این مورد راضی کننده نیست.

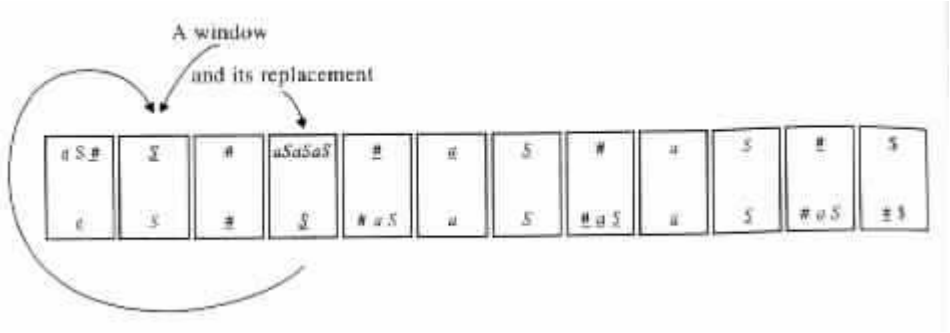
درستی این طلب می تواند توسط قیاس تعداد قواعد تولید در اشتقاق های دردست ساخت یا تعداد زوج مرتب هایی از نوع (d) و (e) استفاده شده در نمونه ی جواب مثبت نشان داده شود.

مثال ٤-٧-٢ اگر G یک گرامر با قواعد تولید { S→aSaSaS, aS→ε } باشد آنگاه (G,ε) بنا بر اثبات قضیه ٤-٧-١ PCP است و داریم:

<(¢S#, ¢), (aSaSaS, S), (aSaSaS, S), (#, #aS), (#, #aS), (a, aaS), (a, aaS), (S, SaS), (S, SaS), (#, #), (#, #), (a, a), (a, a), (S, S), (S, S), ($, #$)>

نمونه ای متناظر با مثال در شکل ٤-٧-١ نمایش داده شده

Error!



شکل ٤-٧-١ یک ترتیب از کارت های PCP برای توصیف کردن یک اشتقاق برای ε در گرامر که قواعد تولید S→aSaSaS, aS→ε را دارد .

بعلاوه ،این شاهد متناظر با اشتقاق های S ⇒S ⇒*aSaSaS ⇒aSaS ⇒aS در G نیز می باشد.

کاربرد مسئله تناظر پست

نتیجه زیر نشان می دهد که مسئله تناظر پست می تواند در مسئله تصمیم ناپذیری در

کاهش استفاده شود.

نتیجه ٤-۷-۱ مسئله هم ارزی برای مبدل های قطعی محدود تصمیم ناپذیر است.

اثبات : ملاحظه می کنید برای هر $<(x_1, y_1), \ldots, (x_k, y_k)>$ از PCP ، Δ الفبای حداقل است که

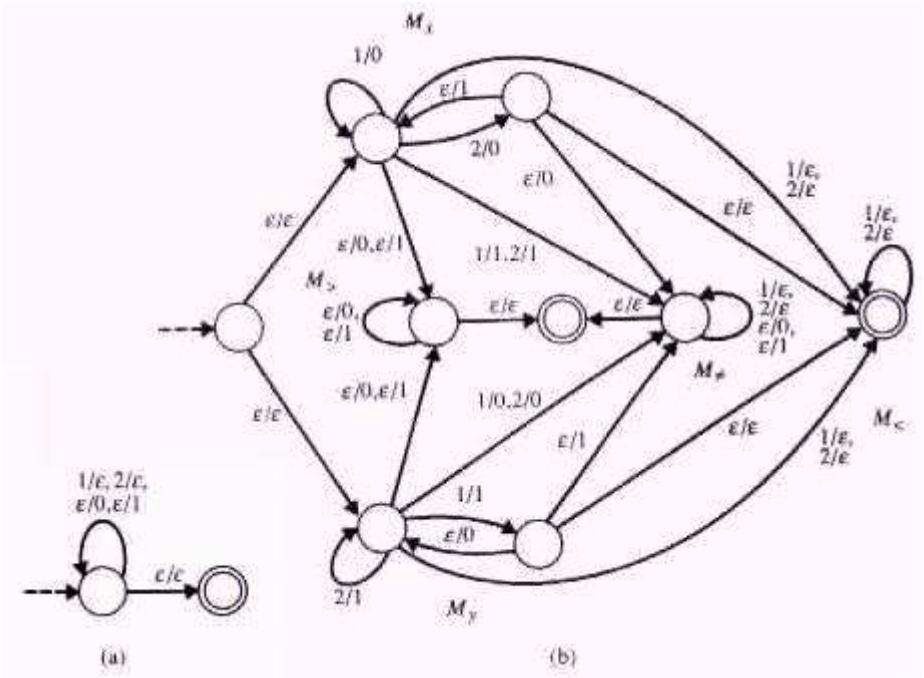
$x_1, \ldots, x_k, y_1, \ldots, y_k$ همه در Δ^* وجو دارند. با فرض اینکه $\{1, \ldots, k\} = \Sigma$ حروف ها هستند.

$M_1 = < Q_1, \Sigma, \Delta, \delta_1, q_0, F_1 >$ یک مبدل های قطعی با حالت متناهی است که محاسبات مربوط به $\Sigma^* \times \Delta^*$ را محاسبه می کند و پذیرنده همه وروردی های روی Σ و هر ورودی که بتواند خروجی روی Δ تولید کند.

$M_2 = < Q_2, \Sigma, \Delta, \delta_2, q_0, F_2 >$ یک مبدل های قطعی با حالت متناهی است که روی ورودی های $i_1 \ldots i_n$ خروجی های شبیه w را دارد که $w \neq x_{i_1} \cdots x_{i_n}$ یا $w \neq y_{i_1} \cdots y_{i_n}$. M_2 روی ورودی $i_1 \ldots i_n$ می تواند رشته خروجی روی Δ^* داشته باشد اگر $x_{i_1} \cdots x_{i_n} \neq y_{i_1} \cdots y_{i_n}$ ، از طرف دیگر، اگر $x_{i_1} \cdots x_{i_n} = y_{i_1} \cdots y_{i_n}$ باشد آنگاه M_2 برای ورودی $i_1 \ldots i_n$ خروجی روی Δ^* دارد، بعلاوه برای $x_{i_1} \cdots x_{i_n}$.

M_1 هم ارز با M_2 است اگر وتنها اگر PCP برای $<(x_1, y_1), \ldots, (x_k, y_k)>$ یک جواب منفی باشد.

مثال ٤-۷-۳ داریم $<(x_1, y_1), (x_2, y_2)> = <(0, 10), (01, 1)>$ با توجه به نتیجه ٤-۷-۱ $\Delta = \{0, 1\}$ و $\Sigma = \{1, 2\}$. مبدل قطعی M_1 میتواند به شکل ٤-۷-۲(a) باشد و M_2 به شکل (b).



شکل ٤-۷-۲ مبدل قطعی حالت متناهی (a) هم ارز مبدل (b) است اگر و تنها اگر PCP یک جواب مثبت برای $<(0, 10), (01, 1)>$ داشته باشد

M_2 روی ورودی $i_1 \ldots i_n$ مجبور به انتخاب M_x و M_y است در M_x خروجی های یک پیشوند از $x_{i_1} \cdots x_{i_n}$ و در M_y خروجی های یک پیشوند از $y_{i_1} \cdots y_{i_n}$ قرار می گیرند.

M_2 به طور غیر قطعی به $M_<$، $M_>$ یا $M_=$ سوییچ می کند.

M_2 از M_x به $M_>$ تغییر می کند تا خروجی مناسب $x_{i_1} \cdots x_{i_n}$ فراهم شود.M_2

از M_x به $M_<$ تغییر می کند تا برای خروجی پیشوند مناسب $x_{i_1} \cdots x_{i_n}$ فراهم شود.M_2 از M_x به $M_=$ تغییر می کند تا خروجی از $x_{i_1} \cdots x_{i_n}$ فراهم شود.

M_2 از M_y هم در حالات مشابه استفاده میکند.

نتیجه ٤-۷-۲ مسئله هم ارزی برای آتاماتای پشته ای (pushdown)تصمیم ناپذیر

است.

اثبات : داریم $<(x_1, y_1), \ldots, (x_k, y_k)>$ از PCP که Σ_1 کمترین الفباست بطوریکه $x_1, \ldots, x_k, y_1, \ldots, y_k$ در Σ_1^* هستند.بافرض اینکه $\Sigma_2 = \{1, \ldots, \Sigma k\}$ حروف هستند که Σ_1 و Σ_2 هر دوگسسته هستند و Z_0 یک عنصر جدید است و متعلق به Σ_1 نیست.

$M_1 = < Q_1, \Sigma_1 \cup \Sigma_2, \Sigma_1 \cup Z_0, \delta_1, q_0, Z_0, F_1 >$ یک آتاماتای پشته ای است که همه رشته های درون $(\Sigma_1 \cup \Sigma_2)^*$ را می پذیرد.

$M_2 = < Q_2, \Sigma_1 \cup \Sigma_2, \Sigma_1 \cup Z_0, \delta_2, q_0, Z_0, F_2 >$ یک آتاماتای پشته ای است که ورودی w را می پذیرد اگر و تنها اگر به فرم $i_n \ldots i_1 u$ باشد و به ازای $i_1 \ldots i_n$ متعلق به Σ_1^* و u متعلق به Σ_2^* بطوریکه $u \neq x_{i_1} \ldots x_{i_n}$ یا $u \neq y_{i_1} \ldots y_{i_n}$.نتیجه می شود که M_1 و M_2 هم ارزند اگر وتنها اگر PCP یک جواب منفی داشته باشد.

آتاماتای پشته ای M_2 در اثبات نتیجه ٤-٧-٢ روی ورودی ها یی متوقف می شود که پذیرنده آنها باشد. آتاماتای پشته ای روی تمامی ورودی ها متوقف می شود اگر وتنها اگر PCP یک جواب منفی داشته باشد.نتیجه زیر دلالت بر تصمیم ناپذیری PCP ها دارد.

نتیجه ٤,٧,٣ مسئله توقف یکنواخت برای آتاماتای پشته ای تصمیم ناپذیر است.

PCP یک مسئله تصمیم پذیر پاره ای است زیرا نمونه داده شده $<(x_1, y_1), \ldots, (x_k, y_k)>$ از مسئله می تواند جزییات کاملی را برای مدرکی از جواب درست جستجو کند.برای مثال در مجموعه استاندارد $\{1, \ldots, k\}^*$ ، متعاقبا اگر جواب منفی باشد جستجو هرگز پایان نمی یابد.

فصل پنجم

محاسبات با منابع محدود

اهداف

در پایان فصل دانشجو با مفاهیم زیر آشنا خواهد شد

زمان و مکان روی مبدلهای تورینگ

سلسله مراتب زمان

زمان چند حمله ای غیر قطعی

مسایل NP

مقدمه

تاکنون، زمان در نظر گرفتن برنامه ها و مسائل ، ما فرض می کردیم که حد و مرزی برای منابعی (مانند زمان و مکان) که در طول محاسبات از آنها استفاده می کنیم وجود ندارد.این فرض ما را قادر می سازد که تعدادی از سوالات مفید در مورد مسائل وبرنامه ها را امتحان کنیم.برای نمونه ما راجع به مسائلی بحث می کردیم که ،صرفنظر از تعداد منابع در اختیار، نمی توانیم باهیچ برنامه ای آنها را حل کنیم .بعلاوه ما راههای گسترش برای شناسایی مسائل غیر قابل حل را جستجومیکردیم .به هر حال مطالعات ما هیچ اشاره ای در مورد امکان اینکه آن مسائل قابل حل هستند فراهم نمی کند.

یک نتیجه طبیعی از مطالعات محاسبات نا محدود، بررسی محاسبات منابع محدود است .هدف این فصل چنین مطالعه ای است و در موارد بسیاری نتیجه آن با تصحیح جزئی از بررسی فصل چهارم به دست می اید.

بخش اول این فصل ما را با مدلهایی از ماشین های دسترسی تصادفی به عنوان انتزاعی برای کامپیوترها آشنا می کند. همچنین نظریه زمان و مکان برای ماشین های دسترسی تصادفی و مبدل تورینگ و ارتباط منابع مورد نیاز از مدلهای متفاوت را معرفی می کند. دومین بخش وجود سلسله مراتبی از مسائل را نشان می دهد که بر اساس زمان مورد نیاز برای حل آنهامی باشد.علاوه بر این بخش دو درباره امکان پذیری محاسبات" زمان چند جمله ای " و امکان پذیر نبودن محاسبات" زمان نمائی " و اهمیت ساده تر شدن مسائل سخت بحث می کند. بخش سوم و چهارم به جایگاه مسائل با زمان چند جمله ای غیر قطعی در سلسله مراتب اهمیت می دهد و هدف آن آسان کردن بعضی مسائل مشکل است . بخش پنجم به محاسبات مکان محدود رسیدگی می کند و بخش ششم با سخت ترین مسائل از میان مسائلی که می تواند در زمان چند جمله ای حل شود سرو کار دارد.

۵-۱ زمان و مکان

زمان و مکان روی مبدل های تورینگ عمومی

زمان ومکان مورد نیاز برای برنامه ، به خود برنامه و واسطی که ان را اجرا می کند،

بستگی دارد .

هر واسط مجموعه ای از داده های اولیه و عملگرهای اولیه خودش را دارد. هر داده اولیه گرفته شده از واسط نیاز به مقداری ثابت از مکان حافظه دارد. همینطور هر عملگر اولیه نیاز به مقدار مشخص ثابتی از زمان اجرا دارد.

بعلاوه هر جفت ازواسطه هایی که برنامه های مشابه را اجرا میکنند نسبتا اولیه هستند. هر داده اولیه از یک واسط نشانگر بعضی مقادیر ثابت از مقادیر داده اولیه از واسطه دیگری باشد. همینطور هرعملگراولیه از یک واسط میتواند مانند بعضی مقادیر ثابت از عملگر اولیه از واسطه دیگری باشد. درزمان اجرای یک برنامه ، یک واسط عناصر پردازش برنامه را با مقادیر داده اولیه خودش نمایش می دهد. بطور مشابه واسط دستورالعملهای برنامه مورد استفاده را با عملگر های اولیه خودش شبیه سازی میکند. درنتیجه هر محاسبه از یک

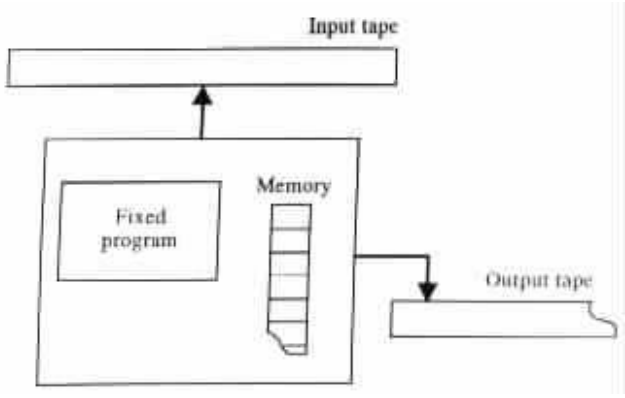
برنامه داده شده نیاز دارد به تعدادی $c_1 s$ مکانی و تعدادی $c_2 t$ زمانی . که s,t فقط به برنامه وابسته اند و c_1,c_2 فقط به واسط وابسته اند. c_1 نشانگر توان فشرده سازی واسط است و c_2 نشانگر سرعت واسط و توان شبیه سازی عملگرهای آن .

به دلیل تفاوت در واسط ها، ثابت های غیر مستقیم c_1,c_2 نیز با هم تفاوت دارند و از آنجا که اهداف مطالعه محاسبات توسعه یک تئوری کلی است ،بدون از دست رفتن کلیت ، مطالعه زمان و مکان را به تحلیل رفتاری محدود می کنیم. یعنی ،تحلیل اینکه ،دقت لازم تا حدود ضریب خطی از مقدار زمان و حافظه لازم برای یک واسط واقعی به دست می آید . چنین تحلیلی می تواند با به کار گیری مدلهایی از ماشینهای محاسبات مانند ماشینهای دستیابی تصادفی و مبدلهای تورینگ مورد استفاده در اینجا انجام شود.

ماشینهای دستیابی تصادفی

در اصل برنامه ها برای اجرا روی کامپیوترها نوشته شده اند .بنابراین کامپیوتر های انتزاعی ،زمان در نظر گرفتن منابع مورد نیاز برای برنامه ها، در مرکز توجه قرار می گیرند . یک کامپیوتر معمولی می تواند به صورت یک نوار ورودی و یک نوار خروجی

, یک برنامه ثابت و یک حافظه دیده شود .(شکل۵-۱-۱ را ببینید)



نوار ورودی و خروجی ،نوار های یکطرفه ترتیبی هستند که ، به ترتیب ،برای نگهداری مقادیر ورودی و خروجی استفاده می شوند . حافظه شامل سلولهایی است که می تواند در هر ترتیبی موردسترسی قرار گیرند.هر سلول می تواند مقادیری را، از دامنه دارای نمایش دودویی ، نگهداری کند.تعداد سلولها می تواند نامحدود فرض شود زیرا در دسترس بودن حافظه های عظیم برای کامپیوترها امکان پذیر است. به طور مشابه ، به دلیل تنوع زیاد مقادیری که می تواند در هر سلول ذخیره شود، اندازه هر سلول نامحدود فرض می شود . برنامه ثابت می تواند شامل هر نوع استانداردی از دستورالعملهای قطعی باشد . (read, write , add, substract , goto)

چنین کامپیوترهای انتزاعی به ماشینهای با دسترسی تصادفی یا RAM نام گذاری شده اند. در ادامه، RAM ها با برنامه های قطعی با خصوصیات مشابه شناسایی می شوند.به ویژه ، فرض می شود برنامه ها دامنه ای از متغیرها دارند که مساوی با مجموعه ای از اعداد طبیعی هستند و متغییرهایی که می توانند به عنوان ارایه های یک بعدی دیده شوند . هر ورودی A(1)از یک ارایه A به یک عملگر اندیس گذاری دست می یابد که پارامترهایش Aو۱است .

مثال ۵-۵-۱

read K

```
for i := 1 up to K do read A(i)
for i := 2 up to K do
   for j := i down to 2 do
      if A(j) < A(j - 1) then A(j) ↔ A(j - 1)
for i := 1 up to K do write A(i)
```

RAM نشان داده شده در شکل ،هر مجموعه از اعداد طبیعی را مرتب می کند .این کار به وسیله یک برنامه قطعی مستقل از قالب و با به کارگیری متغیر های i وj وk و A انجام می شود .

RAM تعداداعضای مجموعه N که باید مرتب شوند را در متغیر k می خواند و عناصر N داخل A(1).....A(n) ذخیره می شوند . RAM کار خود را از زیرمجموعه بدیهی و مرتب که فقط شامل عنصر A(1) است شروع می کند. در هر مرحله عنصر در درایه بعدی A(1) از A به زیر مجموعه مرتب اضافه می شود ودر مکان مناسب جای می گیرد . در مورد RAM دو نوع رایج از معیار ارزش برای تحلیل مکان و زمان وجود دارد: معیار ارزش یکنواخت و لگاریتمی .

تحت معیار ارزش لگاریتمی فرضهای زیر ساخته شده است . یک ایتم داده اولیه قطعه ای از نمایش دودویی از اعداد طبیعی استفاده شده هستند .عملگرها ی اولیه قطعه ای از عملگرهای مورد نیاز برای اجرای دستور العملهایی از RAM هستند . حافظه مورد نیاز در یک محاسبه روی یک RAM مفروض مساوی است با نیاز درایه های متغییرها .

حافظه مورد نیاز برای یک درایه از متغییر از مساوی است با طول نمایش دودویی از بزرگترین مقدار V ذخیره شده در آن که (log(v+1))است اگر v≠0 و یک است اگر v=0 .زمان مورد نیاز برای یک محاسبه برابر است با تعداد قطعه عملگرهایی مورد نیاز برای اجرای دستور العملها. معیار ارزش یکنواخت یک انحطاط از معیار ارزش لگاریتمی است که فرض های زیر ساخته شده اند . هر مقدار یک آیتم داده اولیه است ,حافظه مورد نیاز برای متغییر مفروض مساوی است با تعداد درایه ها در آرایه ای است که آن را نمایش می دهد .

حافظه مورد نیاز یک RAM برابر است با مجموع حافظه های مورد نیاز برای متغییر های ان و زمان مورد نیاز یک RAM برابر است با تعداد دستور العملهای در حال اجرا .

مثال ۵-۱-۲

RAM را از مثال ۵-۱-۱ ملاحظه کنید (شکل ۵-۱-۲ را ببینید)RAM روی ورودی $v_1.....v_n$, N یک واحداز مکان برایk نیازدارد و یک واحد برای i ویک واحد برای j وN واحد برای A تحت معیار ارزش یکنواخت .از سوی دیگر، تحت معیار ارزش لگاریتمی RAM به log N واحد از مکان برای k ، log N واحد برای i , log N برای j و(Nmax(log v1 ….log vn واحد از مکان برای A نیاز دارد.(در این مثال log a مساوی با log (a+1)فرض شده اگر a≠0 باشد و مساوی یک است اگر a=0 .) دستور read k یک واحد از زمان را تحت معیار ارزش یکنواخت می گیرد و log N واحد را تحت معیار ارزش لگاریتمی .

اگر i مقدار یک رادر خود نگه دارد آنگاه دستور read A(i) دو واحد از زمان را تحت معیار ارزش یکنواخت میگیرد: یک واحد برای دستیابی به مقدار یک از i و یک واحد برای دستیابی به مقدار A(1.) تحت معیار ارزش لگاریتمی این دستور به log v_1 + log۱ واحد از زمان نیاز دارد.

.به طور مشابه در چنین موردی دستور writeA(i) تحت معیار ارزش یکنواخت دو واحد از زمان را میگیردو تحت معیار ارزش لگاریتمی log 1+log(the smallest value in {v1…..vn}) واحد را نیاز دارد.

قطعه کد for i:=1 up to k do read A (i) and for i:=1 up to k do write A(i) تحت معیار ارزش یکنواخت زمان خطی در N را میگیرد و تحت معیار ارزش لگاریتمی زمان خطی در (logv_1)+(log2+log v_2)+……+(logN + +۱) $(\log v_N) \leq N\log N + \log(v_1 \ldots v_N)$ را نیاز دارد .

RAM به فضای خطی N تحت معیار ارزش یکنواخت و خطی Nlog m تحت معیار ارزش لگاریتمی نیاز دارد کهm به معنی بزرگترین مقدار ورودی است .RAM به

زمانی نیاز دارد که تحت معیار ارزش یکنواخت نسبت به N^2 خطی است و تحت معیار ارزش لگاریتمی نسبت به خطی $N^2 \log m$ می باشد .

به طور کلی هم مکان و هم زمان مورد نیاز برای پیدا کردن راه حل مسئله در یک نمونه داده شده، با اندازه طول نمایش نمونه ، افزایش می یابد .در نتیجه زمان ومکان مورد نیاز ماشینهای محاسبه به وسیله توابعی از اندازه ورودی مشخص می شوند.در ادامه n به عنوان نماینده طول نمونه ها، در سؤال استفاده خواهد شد .

مثال ۵-۱-۳

یک عدد طبیعی بزرگتر از یک و قابل تقسیم فقط بر یک و خودش ، یک عدد اول نامیده می شود .مسائل اول بودن، برای هر عدد صحیح مثبت m سوال می کند که ایا m اول است یا نه؟

```
read x
if x < 2 then halt with answer no
if x = 2 then halt with answer yes
  √x y :=
do
    if x is divisible by y then halt with answer no
   y := y - 1
until y = 1
halt with answer yes
```

شکل ۵-۱-۳ RAM برای حل مسایل فدیمی

RAM نشان داده شده در شکل قبل ، به وسیله یک برنامه قطعی مستقل از فرمت مسائل اول بودن را با استفاده از روش brute –force حل می کند .

یک ورودی m میتواند به RAM با نمایش یکانی یا دودویی داده شود .در حالیکه متغییرها تنها ارزش دودویی را می توانند بگیرند. یک نمایش یکانی برای m انداره n=m را دارد و یک نمایش دودویی برای mاندازه $n=\log(m+1)$ دارد اگر $m\neq 0$ و اندازه $n=1$ اگر $m=0$.

با یک نمایش یکانی برای یک نمونه m از مسئله، تحت معیار ارزش یکنواخت ، RAM به یک فضای ثابت و یک زمان خطی n نیاز دارد . از سوی دیگر، تحت معیار ارزش لگاریتمی RAM به فضای خطی $\log n$ و زمان خطی $n(\log n)^k$ برای بعضی $k>0$ احتیاج دارد. یک زمان خطی n برای خواندن ورودی m و ذخیره آن درفرم دودویی مورد نیاز است و یک زمان چند جمله ای در $\log n$برای چک کردن بخش پذیری m به یک عدد صحیح i که ($2\leq i\leq\sqrt{2}$) مورد نیاز است .

با یک نمایش دودویی برای یک نمونه m از مسئله، RAM به یک فضای ثابت تحت معیار ارزش یکنواخت و فضای خطی در n تحت معیار ارزش لگاریتمی نیازمند است. اما الگوریتم به زمان چند جمله ای در m یا 2^n تحت هر دو معیارارزش یکنواخت و لگاریتمی نیازمند است.

زمان ومکان در مبدل تورینگ

در مورد مبدل تورینگ در ادامه مفروضات زیر را داریم: عملگرهای اولیه، قاعدهای انتقال هستند و آیتم های داده اولیه، کارکترهایی از حروف الفبا هستند.هر حرکت یک واحد از زمان را می گیرد وزمان یک محاسبه برابر است با تعداد حرکتهای انجام شده در طول محاسبه.مکان لازم برای یک محاسبه برابر است با تعداد مکان های دیده شده در نوار کار کمکی که بزرگترین تعداد آن در نظر گرفته می شود.(یک جایگزین ممکن برای اندازگیری مکان ،جمع تعداد مکان های ملاقات شده در همه نوار های کار کمکی می باشد. از آنجا که تعداد نوارهای کاری کمکی برای مبدل تورینگ داده شده ثابت هستند واز آنجایی که فاکتورهای ثابت در انجام تجزیه نادیده گرفته می شوند، از تعاریف قبلی استفاده می کنیم .)

مبدل تورینگ M، یک مبدل تورینگ زمان محدود $T(n)$ یا از پیچیدگی زمانی $T(n)$ نام دارد اگر هر محاسبه ممکن از M روی هر ورودی از xاز طول n زمانی نه بیشتر از $T(n)$ مصرف کند.M زمان محدود ی چند جمله ای یااز پیچیدگی زمانی چند جمله ای نامیده می شوداگر $T(n)$یک چند جمله ای درn باشد.مبدل تورینگ M ، یک مبدل تورینگ مکان محدود$s(n)$ یا مبدل تورینگ از پیچیدگی مکانی $s(n)$ نام دارد اگر

همه محاسبات ممکن از M بر روی هر ورودی x از طول n ، مکانی بیشتر از s(n) نگیرد. M یک مکان محدود چندجمله ای یا از پیچیدگی مکان چندجمله ای است اگر S(n) چندجمله ای از n باشد .M مکان محدود لگاریتمی یااز پیچیدگی مکانی لگاریتمی نام دارداگرs(n)= c log n برای برخی ثابت c باشد. تعاریف مشابهی برای ماشین های تورینگ,RAM هاو دیگر طبقات در ماشین های محاسبات برقرار می باشند.

جملات زیر یک تصحیح به قضیه Church اضافه می کند. در مورد قضیه اصلی ,اثبات درستی این امکان پذیر نیست . اگرچه، دراینجا می توان به طور شهودی، با نشان دادن وجود ترجمه ای بین کلاس های مختلف از مدل های محاسباتی ، شخص را به درستی این مطلب متقاعد کرد. ترجمه بین RAM ها ومبدل تورینگ قطعی می تواند به آنچه در بخش ٤,٢ ارایه شد،شبیه باشد.

قضیه محاسبات ترتیبی

یک تابع محاسبه پذیر (یا به ترتیب , محاسبه پذیر پاره ای) به وسیله یک الگوریتم A است تنها اگر آن محاسبه پذیر (یا به ترتیب محاسبه پذیر پاره ای) بوسیله مبدل تورینگ قطعی باشند که شرایط زیر را دارا باشند :الگوریتم A ورودی داده شده محاسباتی داشته باشد که T(n) زمان وs(n) مکان را می گیرد تنها اگر در چنین ورودی مبدل تورینگ محاسبه ای کهP(T(n)) زمان و P(S(n)) مکان را می گیرداشته باشد که P() یک چند جمله ای ثابت است که به ورودی بستگی ندارد.

پیچیدگی مسائل

بدون ازدست دادن کلیت ،درآنچه که در زیر می آید، فرض می کنیم که زمان محدودT(n) برابر است با max(T(n),n) ، یعنی، برابر است باکمترین زمان مورد نیاز ی برای خواندن همه ورودیها.بعلاوه فرض می کنیم یک مکان محدود یs(n) برابر است باmax(1,s(n)). log 0 برابربا ۱فرض می شود.

کاربرد نماد O بزرگ f(n) = (O g(n)) برای این است که: وجود دارد یک ثابت c>0 وn_0 کهf(n)≤g(n)برای همه $n=>n_0$.در چنین مواردی f(n) از مرتبه g(n)

خوانده میشود.

مساله ای دارای پیچیدگی زمانی$T(n)$ خواهد بود اگر آن بوسیله یک مبدل تورینگ قطعی زمان محدود $T(n)$ قابل حل باشد. ($T(n$ مساله ای دارای پیچیدگی زمانی غیر قطعی $T(n)$خواهد بود اگر آن بوسیله یک مبدل تورینگ زمان محدود $T(n)$ قابل حل باشد. مساله ای دارای پیچیدگی مکانی $s(n)$ خواهد بود اگر آن بوسیله مبدل تورینگ قطعی مکان محدود $s(n)$ حل شود. مساله ای دارای پیچیدگی مکانی غیر قطعی ($s(n$خواهد بود اگر آن بوسیله مبدل تورینگ مکان محدود $s(n)$ حل شود.

به طور مشابه، یک زبان دارای پیچیدگی زمانی ($T(n$خواهد بود اگر انها بوسیله ماشین توریینگ غیر قطعی زمان محدود ($T(n$پذیرفته شود.زبان دارای پیچیدگی زمانی غیر قطعی $T(n)$ خواهد بود اگر ان بوسیله ماشین تورینگ غیر قطعی زمان محدود ($T(n$ پذیرفته شود.زبان دارای پیچیدگی مکانی $s(n)$ خواهد بود اگر بوسیله ماشین تورینگ قطعی مکان محدود $s(n)$ پذیرفته شود. زبان دارای پیچیدگی مکانی غیر قطعی $s(n)$خواهد بود اگر انها بوسیله ماشین تورینگ غیر قطعی مکان محدود $s(n)$ پذیرفته شود.

طبقه های پیچیدگی

طبقه های زیرین برای مطالعه ما درمورد زمان ومکان مهم هستند.

$DTIME(T(n))$

طبقه ای از زبان ها که پیچیدگی زمانی$O(T(n))$ دارد.

$NTIME(T(n))$

طبقه ای از زبان ها که پیچدگی زمانی غیر قطعی$O(S(n))$ دارد.

$DSPACE(S(n))$

طبقه ای از زبان ها که پیچیدگی مکانی قطعی$O(S(n))$ دارد.

$NSPACE(S(n))$طبقه ای از زبان ها که پیچیدگی مکانی غیر قطعی$O(S(n))$ دارد.

P

طبقه ای از مسائل عضویت برای زبان ها در

$$\bigcup_{p(n)} DTIME(p(n))$$

(P(n) چند جمله ای از n است)

NP

طبقه ای از مسائل عضویت برای زبانهای در

$$\bigcup_{p(n)} NTIME(p(n))$$

EXPTIME

طبقه ای از مسائل عضویت برای زبان ها در

$$\bigcup_{p(n)} DTIME(2\hat{}\ p(n))$$

PSPACE

طبقه ای از مسائل عضویت برای زبان ها در

$$\bigcup_{p(n)} DSPACE(p(n))$$

NLOG

طبقه ای از مسائل عضویت برای زبان ها در NSPACE(log n)

DLOG

طبقه ای از مسائل عضویت برای زبان ها در DSPACE(log n)

بنابراین تحلیل ما از پیچیدگی مسائل معنی دار خواهد بود ،فقط نمایش های طبیعی برای مورد های خودشان فرضی هستند.طبیعی بودن نسبت به منابع تحلیل شده در نظر گرفته شده اشت .

مثال ۵-۱-٤. مساله اول بودن بوسیله مبدل تورینگ قطعی در زمان چند جمله ای حل می شود اگر نمونه ها در نمایش یکانی داده شوندو در زمان نمایی اگر نمونه ها در نمایش غیر یکانی نمایش داده شوند(مثال ۵-۱-۳ را ببینید) بهر حال برای هر نمونه داده شده mهر دو شیوه به زمان چند جمله ای از m نیاز دارند .

هنگام ملاحظه پیچیدگی مساله اول بودن، یک نمایش غیر یکانی برای نمونه های مطرح شده طبیعی ویک نمایش یکانی برای نمونه های مطرح شده غیر طبیعی به حساب آمد . انتخاب ویژه اندازه d از یک نمایش غیر یکانی مهم نیست زیرا درجه چنین تفاوتهای نمایشی از یک m با فاکتور ثابت برابر است. مخصوصا اندازه $\log_{d_1} m$ و اندازه $\log_{d_2} m$ برای جفت نمایش ها از m رابطه $\log_{d_1} m = (\log_{d_1} d_2)\log_{d_2} m$ را وقتی که d_1 و d_2 بزرگتر از ۱ باشند ارضا می کند .

در نتیجه ،RAM در شکل ۵-۱-۳ وتز محاسبات ترتیبی نشان می دهد مساله اول بودن دارای پیچیدگی زمانی نمایی می باشد .

زمان ومکان در مبدل تورینگ عمومی

تحلیل اثبات تئوری ٤-٤-۱ لم زیر را فراهم می کند .

لم : مبدل تورینگ عمومی U از تئوری ٤-٤-۱ بر روی داده ی داده شده (M,x) از مبدل تورینگ قطعی M و ورودی x برای M

a: در $c_M t^2$ حرکت توقف می کند اگر M در t حرکت بر روی ورودی x که $t \geq |x|$توقف کند .

b: حداکثر $c_M s$ مکان در هر نوار کاری کمکی ملاقات می کند. اگر M کمتر از s مکان در هر نوار کاری کمکی ملاقات کند و $s \geq \log |x|$ باشد.

c_M چند جمله ای از طول نمایش M فرض می شود .(چند جمله ای به M بستگی ندارد.)

اثبات : نماد گذاری در اثبات ٤-٤-۱ را فرض میکنیم. تعداد حرکات مورد نیاز U

برای بررسی ورودی صحیح (M,x) ، حداکثر مقدار ثابتی ضرب در |x| (اندازه x) است که این مقدار ثابت تنها به طول نمایش از M بستگی دارد.

به ویژه U به $|E(M)| + 3$ حرکت برای یافتن $E(M)$ نیاز دارد . $|E(M)|$ حرکت برای پیمایش $E(M)$و ۳ حرکت برای معلوم کردن ۰۱ که به دنبال پسوند ۰۱ از $E(M)$ می آید.

بررسی صحت برای نمایش $E(M)$از یک مبدل تورینگ M ، به تعدادی حرکت احتیاج دارد که نسبت به $|E(M)|$ خطی است ، یعنی ،

a: $|E(M)|$ حرکت برای معلوم کردن عدد m از نوار کاری کمکی از M و برای بررسی اینکه هر قاعده انتقال $\tau=(q,a,b_1,...,b_m,p,d_0,c_1,d_1,...,c_m,d_m,\rho)$ از M که شامل $3m+5$ درایه است.هر قاعده انتقال τ در $E(M)$ به وسیله زیر رشته $E(\tau)$ نمایش داده شده است که بین دو جدا ساز ۰۱ قرار گرفته است . زیر رشته $E(\tau)$ باید دقیقاشامل $3m+5$ صفرباشد.

b:$|E(M)|$ حرکت برای معلوم کردن اینکه جابجایی هد d_i در قاعده انتقال . بوسیله رشته های دودویی به فرم $E(0)=0111$,$E(-1)=01$, $E(+1)=01111$ نمایش داده شده است .

c: $|E(M)|$ حرکت برای معلوم کردن اینکه قاعده انتقال به سمبل خالی B ازM تنها در نوار کاری کمکی و به نشانه انتهایی سمت چپ c ونشانه انتهایی سمت راست $ تنها در نوار ورودی رجوع می کند.

d:$|E(M)|$ برای معلوم کردن اینکه هیچ حالتی از M با رشته باینری ۰ نمایش داده نمی شود.

اینکه M در تعداد حرکت هایی که نسبت به $|E(M)|^2$ خطی هستند قطعی است ، بررسی می شود.این بررسی می تواند با کپی کردن $E(M)$به یک نوار کاری کمکی از U انجام شود و سپس هر قاعده انتقال τ در نوار کاری کمکی از U با هر قاعده انتقال که به دنبال τ در نوار ورودی U می آید مقایسه می شود.

بررسی صحت ورودی x برای مبدل تورینگ M به زمان خطی نسبت به $|E(M)|$ ($|E(M)| + |x|$) نیاز دارد. به ویژه U در زمانی که نسبت به $|E(M)|^2$ خطی است، سمبل های ورودی M را تعیین می کند و آنها را در نوار کار کمکی ذخیره می کند. سپس U در $|x|$. $|E(M)|$ زمان این که تنها نشانه ها از نوار کاری کمکی در x هستند را چک می کند.

U به $\tilde{c}\, s \leq 3+2m+(ms+1)|E(M)|+\log |x|$ مکان در نوارهای کار کمکی نیاز دارد تا رشته $\#(E(q)\#|u|\#E(u_1)\#E(v_1)\#...\#E(u_m)\#E(v_m)\#$ را ثبت کند که

پیکربندی $(uqv, v_1qv_1, ..., u_mqv_m, w)$ از M را نمایش می دهد . که $\tilde{c}=8|E(M)|m$ است .U به $\log |x|$ مکان برای $|u|$, $|E(M)|$مکان برای q، $|E(M)|$ مکان برای هر سمبل در هر u_i و $|E(M)|$ مکان برای هر سمبل در هر v_i و $2m+3$ مکان برای نشانه های # نیاز دارد .

رشته داده شده $\#E(q)\#\#E(u_1)\#E(v_1)\#...\#E(v_m)\#$ مبدل تورینگ عمومی U می تواند حداکثر در $\tilde{c}\tilde{c}(s+|x|) \leq c\tau$ حرکت $m+2$ عنصر ابتدایی $q,a,b_1,b_2,...,b_m$ از قاعده انتقال $\tau=(q,a,b_1,b_2,...,b_m,p,d,c_1,d_1,...,c_m,d_m)$ که در m حرکت بعدی از Mاستفاده می شود را معلوم می کند. که $\tilde{c}$ ثابتی است که اندازه آن نسبت به $|E(M)|$ خطی است و b_i به اولین نشانه در v_iB اشاره می کند .مبدل حداکثر $\tilde{c}s$ حرکت برای استخراج $|u|$و $E(q)$و $E(b_1),...,E(b_m)$ (,) از $\#(E(q)\#|u|\#E(u_1)\#E(v_1)\#...\#E(u_m)\#E(v_m)\#$ انجام می دهد. به ویژه $|u|$٦ حرکت روی نمایش رشته $|u|$ برای شمارش از $|u|$ تا ۰ نیاز دارد. (ببینید تمرین ۵,۲,۱) و $|E(M)|+|01|+|E(M)|$ حرکت برای استخراج یک سمبل از ورودی نوار نیاز دارد.

با در دست بودن $m+2$ عنصر ابتدایی $(q,a,b_1,...,b_m)$ درτ، مبدل تورینگ عمومی U می تواند چند تایی $(p,d_0,c_1,d_1,...,c_m,d_m,\rho)$ در حرکت واحد روی نوار

ورودی مشخص کند .با داشتن چنین چند تایی ، U همچنین می تواند پیکربندی ثبت شده از M را در یک حرکت واحد تغییر دهد.

در نتیجه تعداد کل حرکات U برای شبیه سازی کردن حرکات M بزرگتر از ct^2 نیست .c چند جمله ای (مستقل از M) به طول نمایش باینری استاندارد از M می باشد.

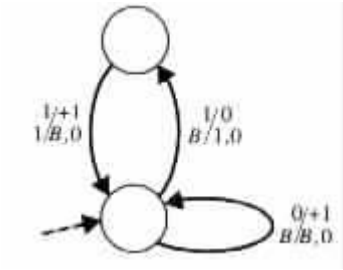
۵-۲ سلسله مراتب زمان

به طور واضح اشکار به نظر می رسد که تعدادی از مسائل به زمان بیشتری برای حل ،نسبت به دیگر مسائل نیاز دارند. نتیجه زیر،این برآورد شهودی را تایید می کند که باید سلسله مراتب زمانی برای کلاس مسائل تشخیص زبان وجود داشته باشد .

تعاریف: تابع $T(n)$ قابل ساخت زمانی گفته می شود اگر یک ماشین تورینگ قطعی با محدودیت زمان $T(n)$ وجود داشته باشد که برای هر n یک ورودی به طول n که دقیقاً $T(n)$ حرکت انجام دهد. یک تابع دارای کاملاً قابل ساخت زمانی گفته می شود اگر یک ماشین تورینگ قطعی که دقیقا $T(n)$حرکت روی هر ورودی به طول n انجام دهد ، وجود داشته باشد . یک تابع $s(n)$ قابل ساخت مکانی گفته می شود اگر ماشین تورینگ قطعی با محدودیت مکانی $s(n)$ وجود داشته باشد که برای هرn ورودی به طول n داشته باشد که روی آن دقیقاً به $S(n)$ مقدار از فضا نیاز داشته باشد . یک تابع کاملاً قابل ساخت مکانی گفته می شود اگر یک ماشین تورینگ قطعی وجود داشته باشد که

روی هر ورودی به طول n دقیقا به $S(n)$ فضا نیاز داشته باشد.

مثال ۵-۲-۱ ماشین تورینگ قطعی M در شکل ۵-۲-۱



شکل ۵-۲-۱ زمان محدود $T(n) = 2n$ برای ماشین تورینگ قطعی.

دقیقا(number of 1's in x)+|x|= t(x) حرکت روی ورودی $t(x)= 2|x|$ زمانی که x شامل هیچ ۰ نیست و $t(x)< 2|x|$ وقتی که x شامل ۰ است .

وجود M مشخص می کند که $T(n)=2n$ یک تابع قابل ساخت زمانی است زیرا

a : M دارای محدودیت زمان 2n می باشد

b: برای هر n ورودی 1^n از طول n وجود دارد که M روی آن دقیقا 2n حرکت انجام می دهد.

وجود یک ماشین تورینگ قطعی M به اینکه ۲n یک تابع کاملاً قابل ساخت زمانی است دلالت نمی کند. زیرا M دقیقا 2n حرکت را روی هر ورودی از طول n انجام نمی دهد . هر چند M میتواند با انجام اصلاحاتی نشان دهد که 2n یک تابع کاملاً قابل ساخت زمانی است .

قرار داد : در این بخش M_x ماشین تورینگی است که با رشته x و به فرم زیر نمایش داده میشود.اگر $x=1^jx_0$ برای بعضی $j \geq 0$ و برای بعضی نمایش دودویی استاندارد x_0 از یک ماشین تورینگ قطعی M آنگاه M_x مشخص کننده M است. در غیر اینصورت M_x مشخص کننده یک ماشین تورینگ قطعی است که هیچ ورودی را نمی پذیرد.

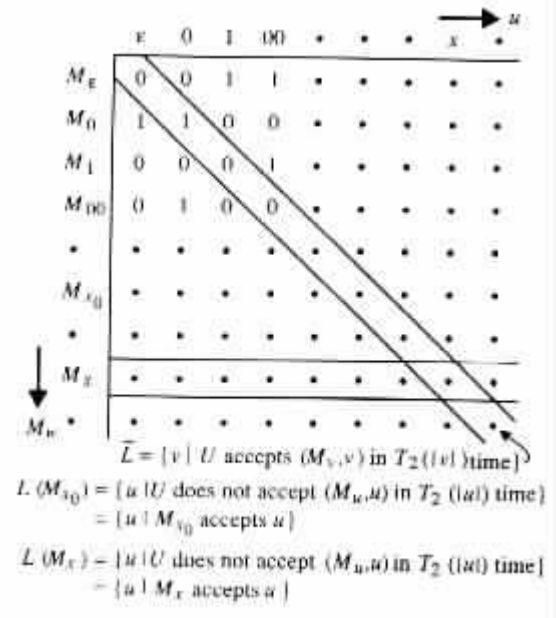
.به رشته x یک نمایش دودویی لایه ای از M_x گفته میشود.

قضیه ۵-۲-۱ : هر تابع $T_1(n)$ و هر تابع کاملاً قابل ساخت زمانی $T_2(n)$ رادر نظر

بگیرید که برای هر $c>0$ یک n_c دارد به طوری که $T_2(n)\geq c\,(T_1(n))^2$ برای همه $n>n_c$. سپس زبانی وجود دارد که در $DTIME(T_2(n))$ هست اما در $DTIME(T_1(n))$ نیست.

اثبات : $T_1(n)$ و$T_2(n)$ به عنوان آنچه در قضیه آمده در نظر بگیرید. U یک ماشین تورینگ عمومی شبیه مبدل تورینگ عمومی در اثبات لم ۵-۱-۱ می باشد. تفاوت اصلی در اینجاست که برای U ورودی (M,x) که در M با نمایش دودویی لایه ای ، به جای نمایش دودویی استاندارد، نشان داده شده است ، فرض شده است. U هر محاسبه را با رفتن بر روی لایه 1^j شروع می کند تا زمانی که به اولین صفر در ورودی برسد .سپس U محاسبه را با روش معمولی با نادیده گرفتن لایه ادامه میدهد. U از یک نوار کار کمکی سوم برای نگهداری مسیرفاصله هد ورودی از انتهای لایه استفاده می کند.نتیجه با قطرسازی روی زبان $L=\{ v \mid v \text{ is in } \{0,1\}^* \}$ و(M_v,v) U را درزمان $T_2(|v|)$ نمی پذیرد.} نمایش داده می شود.

L از قطر جدول $T_{universal}$ گرفته می شود (شکل ۵-۲-۲ را ببنید)



شکل ۵-۲-۲ : جدول فرضی $T_{universal}$ نشان دهنده عضویت در زبان $\{(M_v,v)\ l\ U\ (M_v,v)\}$ را دارد $\{$ (accepts (M_w,u) in $T_2(|u|)$ time $\}$

در جدول $T_{universal}$ درایه سطر M_w و ستون u مساوی با یک اگر $U\ (M_x,u)$ را در زمان $T_2(|u|)$ بپذیرد و مساوی صفر است اگر آن را نپذیرد .اثبات بر این مشاهده تکیه می کند که هر ماشین تورینگ قطعی زمان محدود $O\ (T_1(n))$, M_x که زبان L را می پذیرد دارای نمایش لایه ای x برای U است که می تواند تمام محاسبات M_x روی x را در زمان $T_2(|x|)$ شبیه سازی کند.

در نتیجه $x\ M_x$ را می پذیرد اگر وتنها اگر $U\ (M_x,x)$ را نپذیرد و یا به طور معادل اگر وتنها اگر $x\ M_x$ را نپذیرد.

خصوصا, نمایش این که L در $DTIME(T_1(n))$ نیست , فرض میکنیم که L در آن کلاس باشد.تحت این فرض یک ماشین تورینگ قطعی زمان محدود $M\ dT_1(n)$ وجود دارد , که زبان L را می پذیرد برای بعضی ثابت d .

x_0 یک نمایش دودویی استاندارد از M است و c ثابت متناظر c_M که با لم ۵,۱,۱برای نمایش x_0 از M به کار گرفته می شود . بگذارید $x=1^jx_0$ برای j هایی که رابطه $j+c(dT_1(j+|x_0|))^2 \le T_2(j+|x_0|)$ ارضا می کند ،یعنی ، $x=1^jx_0$ برای j های به اندازه کافی بزرگ که به U زمان کافی $T2(|x|)$ را برای شبیه سازی تمام محاسبات از M_x روی ورودی x می دهد.چنین مقداری برای j وجود دارد زیرا برای j های به اندازه کافی بزرگ نامساوی های زیر برقرار هستند :

$$^{2}(dT_1(j + |x_0|))j + c \leq {}^{2}(dT_1(j + |x_0|))(j + |x_0|) + c$$

$$\leq {}^{2}(dT_1(j + |x_0|))T_1(j + |x_0|) + c$$

$$\leq {}^{2}(T_1(j + |x_0|))(1 + cd^2)$$

$$\leq T_2(j + |x_0|)$$

رشته $x=1^jx_0$ رادر نظر بگیرید .با تعریف $j+c(dT_1(|x|))^2 \leq T_2(|x|)$ و $|x|=j+|x_0|$.به علاوه x یک نمایش دودویی لایه ای از M هست.برای رشته x یکی از دو مورد زیر برقرار هستند. تا تناقض مطلوب با فرض وجود L در $DTIME(T_1(n))$ به دست آید .

.

مورد ۱: x در L هست .این فرض به همراه $L=L(M_x)$ مشخص می کند که x M_x را در زمان $dT_1(|x|)$ می پذیرد.در چنین موردی با لم ۵-۱-۱, (M_x,x) U را در زمان $j+c(dT_1(|x|))^2 \leq T_2(|x|)$ می پذیرد. از سویی دیگر، x در L است و با تعریف از L ،معلوم می شود که x U را در زمان $T_2(|x|)$ نمی پذیرد .تناقض مشخص می کند که این مورد نمی تواند برقرار باشد .

مورد ۲:x در L نیست . این فرضبه همراه $L=L(M_x)$ مشخص می کند که x M_x را نمی پذیرد.در چنین موردی U همچنین (M_x,x) را نمی پذیرد. از سویی دیگر x در L نیست به همراه تعریف L معلوم می کند که (M_x,x) U را می پذیرد.تناقض نشان می دهد که همچنین این مورد نمی تواند برقرار باشد .

برای نشان دادن اینکه L در $DTIME(T2(n))$ هست ، یک ماشیت تورینگ قطعی M با ٤ نوار کار کمکی در نظر بگیرید که روی ورودی x بر اساس الگوریتم زیر پیش می رود :

مرحله ۱

M، (M_x,x) را در اولین نوار کار کمکی ذخیره می کند. M رشته x را , که در ادامه آن جداساز ۰۱ آمده است و در ادامه با نمایش ۰۱۱ از نماد پایانی چپ c در ادامه با x در ادامه با نمایش ۰۱۱۱ از نماد پایانی راست s ذخیره می کند.به علاوه M دنباله رشته

ها را که در بالا ذکر شد بین نماد پایانی چپ و نماد پایانی راست محدود می کند.

مرحله ۲

M مقدار $T_2(|x|)$ را محاسبه کرده و آنرا در دومین نوار کار کمکی ذخیره می کند .

مرحله ۳

M حرکات U را روی محتوای اولین نوار کار کمکی یعنی (M_x,x) دنبال می کند.

M سومین وچهارمین نوار کار کمکی برای ثبت محتوای دو نوار کار کمکی U استفاده می کند . M در پیکربندی پذیرش توقف می کند اگر مشخص شود که U در $T_2(|X|)$ حرکت به حالت پذیرش نمی رسد .در غیر این صورت Mدر پیکربندی غیرپذیرش متوقف می شود. با این روش ساخت، ماشین تورینگ M دارای پیچیدگی زمانی $O(T_2(|X|))$می باشد .

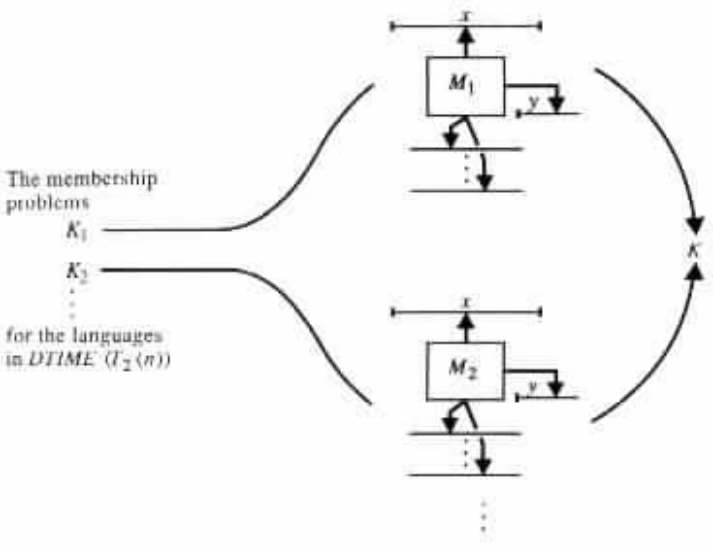
قابلیت ساخت کامل از $T_2(n)$ برای مرحله دوم مورد نیاز می باشد.

مثال ۵-۲-۲

توابع $T_1(n)=n^k$ و $T_2(n)=2^n$ را در نظر بگیرید. $T_1(n)$ و $T_2(n)$در شرایط قضیه ۵-۲-۱ صدق می کنند. بنا بر این کلاس $DTIME(2^n)$ کلاس $TIME(n^k)$راکاملاً شامل می شود .

حدود پایین در پیچیدگی زمان

علاوه بر این که قضیه ۵-۲-۱ به وجود سلسله مراتب زمانی برای مسائل شناخت زبان دلالت می کند ،می تواند حدود پایین روی پیچیدگی زمان بعضی مسائل را نشان دهد. هر دو تابع $T_1(n)$ و $T_2(n)$ را در نظر بگیرید که در شرایط قضیه ۵-۲-۱ صدق می کنند.فرض کنید هر مسئله عضویت K_i برای زبان در $DTIME(T_2(n))$ می تواند به وسیله مبدل تورینگ قطعی M_i با محدودیت زمانی $T_3(n)$ به برخی مسائل ثابت Kکاهش پیدا کنند.(شکل ۵-۲-۳)



شکل ۵,۲,۳

مجموعه ای از مبدل های تورینگ $M_1, M_2, \ldots$ برای کاهش مسائل $K_1, K_2, \ldots$ در $DTIME(T_2(n))$ به مسئله تشخیص زبان K. هر M_iروی نمونه xاز K_i، نمونه y از Kرا فراهم می کند که K پاسخ بله برای y دارد

اگر و تنها اگر K_iبرای x پاسخ بله داشته باشد .

علاوه بر این فرض می کنیم که هر M_iروی ورودی x به طول n خروجی y با حداکثر طول f(n) را فراهم می کند.آنگاه مسائل عضویت برای زبانها در $DTIME(T_2(n))$در زمان $T_3(n)+T(f(n))$ قابل تصمیم گیری میباشند اگر K بتواند درزمان T (n) حل شود.

در چنین موردی حد پایین برای پیچیدگی زمان T(n)ازKبیان شده بنا بر قضیه ۵-۱-۲

کلاس DTIME(T2(n)) از مساله ای تشکیل شده است که به بیش از $cT_1(n)$ زمان برای هر نمونه c نیاز دارد.

نا مساوی $T_3(n)+T(f(n)) > cT_1(n)$ که باید برای تعداد نامتناهی از n بر قرار باشد.حد پایین از جانشین کردن m برای f(n) برای بدست آودن نامساوی T(m) $cT_1(f^{-1}(m)) - T_3(f^{-1}(m))$ > یا بطور معادل

نامساوی $T(n) > cT_1(f^{-1}(n)) - T_3(f_{-1}(n))$ ، به دست می آید.

مثال ۵-۲-۳

حدود زمانی $T_1(n) = 2^{an}$ و $T_2(n) = 2^{bn}$ برای $a2 < b$ و $T_3(n) = f(n) = n$ را در نظر بگیرید. برای هر انتخاب $\log n$ مشخص $cT_1(n) > T(f(n)) + T_3(n)$ می کند که $c2^{an} > \text{nlogn} + T(\text{nlogn})$. بوسیله جانشین کردن $n \log n$ برای m نتیجه می شود که

$T(m) > c2^{an} - m = c2^{am/\log n} - m \geq c2^{am/\log m} - m \geq 2^{dm/\log m}$، یا به طور معادل ،برای بعضی مقادیر d بیان می کند $T(n) > 2^{dn/\log n}$.

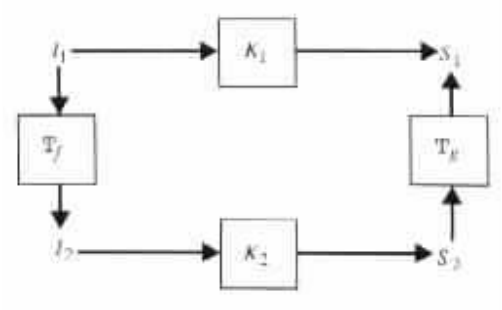
روش بالا برای بدست آوردن حدود پایین روش موردعلاقه ای برای شناسایی مسائل رام نشدنی است . یعنی مسائلی که برای حل شدن به میزان غیر ممکنی از منابع نیاز دارند. چنین شناسایی می تواند در تلاش قابل توجهی که ممکن است برای حل مسائل رام نشدنی هدر رود صرفه جویی کند.

رام شدنی و رام نشدنی

بطور کلی ،مسئله ای به عنوان رام شدنی در نظر گرفته می شود که اگر از پیچیدگی زمان چند جمله ای باشد.این به این دلیل است که زمان مورد نیاز آن با طول ورودی به آهستگی رشد می کند.از طرف دیگر مسائل با پیچیدگی زمان نمایی ،رام نشدنی در نظر گرفته می شوند زیرا زمان مورد نیاز آنها با طول ورودی به سرعت رشد می کندو بنابر این تقریبا" فقط برای ورودی های کوچک قابل حل می باشند.به عنوان نمونه افزایش به وسیله عامل ۲ در n ،مقدار چند جمله ای $p(n)$ ازمرتبه k را حداکثر بوسیله عاملی از 2^k افزایش می دهد. به عبارت دیگر چنین افزایشی حداقل مقدار $2^{p(n)}$ را مجذور می کند.

استفاده از روش بالا برای شناسایی مسائل رام نشدنی ،کاهش های با زمان محدود چند جمله ای را بکار می گیرد.

مسئله K_1 *قابل کاهش در زمان چند جمله ای* به مسئله K_2گفته می شود اگر مبدل تورینگ قطعی زمان محدود چند جمله ای , T_fو T_g وجود داشته باشند که برای هر نمونه I_1از K_1در شرایط زیر صدق کند (شکل ۵-۲-٤)



شکل ۵-۲-٤

کاهش بوسیله مبدل تورینگ قطعی زمان محدودچند جمله ای T_g, T_f.,

1. T_fروی ورودی I_1 نمونه I_2 از K_2را می دهد.

2.K_1راه حل S_1 برای I_1را دارد اگر وتنها اگر K_2راه حل S_2 برای I_2را داشته باشد, که S_1 خروجی T_g روی ورودی S_2میباشد.

در صورتیکه K_1 و K_2 مسائل تصمیم گیری باشند، بدون از دست دادن کلیت ،می توان فرض کرد که T_g تابع همانی $g(S) = S$ را محاسبه کند که T_g روی ورودی S_2 خروجی $S_1 = S_2$ را تولید کند.

کلاس پیچیدگی داده شده C از مسائل می توانندبرای نشان دادن رام نشدنی بودن مسئله K استفاده شوند. این کار با نشان دادن برقراری دو شرط زیر انجام می شو:

1. Cبرخی مسائل رام نشدنی را در بر دارد.

2.هر مسئله در C زمان چند جمله ای قابل کاهش به K می باشدکه K به سختی حل هر مسئله درC می باشد.

زمانی که مشخص شد مسئله K رام نشدنی است ممکن است برای نشان دادن رام نشدنی بودن برخی مسائل دیگر $\acute{K}$ بوسیله نشان دادن اینکه K درزمان چند جمله ای قابل کاهش به $\acute{K}$ می باشد استفاده شود. در چنین حالتی K آسانتر , از میان تمام کاهشها , کاهش آسانتر و گروه بزرگتر از مسائل قابل اجرا $\acute{K}$ می باشد.

مشاهده بالا نظر ما را به " آسانترین" مسائل رام نشدنی K و کلاس های پیچیدگیCکه همگی "آسانترین" مسائل رام نشدنی هستند جلب می کند.

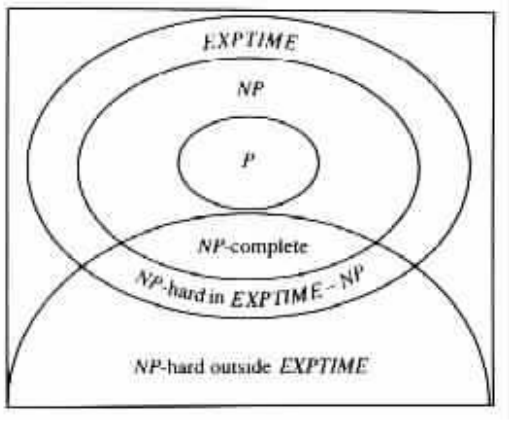
در ادامه مسئله K که یک مسئله سخت C نسبت به کاهش های زمان چند جمله ای گفته می شود, یا فقط مسئله سختC ،زمانی که کاهش های زمان چند جمله ای معلوم

باشند گفته می شود, اگر هر مسئله در گروه Cدر زمان چند جمله ای قابل کاهش به مسئله K باشد.مسئله K ، Cکامل گفته می شود اگرK یک مسئله سخت C در C باشد.

آنچه در اینجا مورد نظر ما می باشدC = $PSPACE$ and C =NP است .

۵-۳ زمان چند جمله ای غیر قطعی

زیر کلاس, NP از کلاس مسائلی است که میتوانند بصورت غیر قطعی در زمان چند جمله ای حل شوند , به نظر می رسد که نقش مرکزی در بررسی رام نشدنی بودن ایفا می کند. طبق تعریف ،NP شامل کلاس P از آن دسته مسائلی است که به صورت قطعی در زمان چند جمله ای می توانند تصمیم گیری شوند, و بر طبق نتیجه ۵-۳-۱ کلاس NP در کلاس EXPTIME از آن مسائلی که بصورت قطعی در زمان نمایی میتوانند تصمیم گیری شونددر بر گرفته شده است.علاوه بر این بر طبق قضیه ۵-۲-۱ P,کاملا" در EXPTIME قرار دارد.(شکل ۵-۲-۱)



شکل ۵-۳-۱

طبقه بندی مسائل قابل تصمیم گیری

اگر چه مشخص نیست که آیاP کاملا" درون NP میباشد یا نه و آیا NP کاملا" درون EXPTIME می باشد یا نه. در نتیجه بنا براین اهمیت NP از آنجا است به عنوان مرز میان رام شدنی بودن رام نشدنی بودن مسائل پیشنهاد شده است.

به خصوص اگر مشخص شود که NP با P برابر نیست , همانگونه که خیلی احتمال دارد , پس NP انتظار می رود برخی از آسانترین مسائل (به نام , مسائل کامل NP)را برای نشان دادن رام نشدنی بودن مسائل جدید به کمک قابلیت کاهش را بیان کند . از سوی دیگر اگر مشخص شود NP با P برابر است آنگاه بسیاری ازمسائل مهمی که کار روی آنها برای دهه ها بدون هیچ موفقیتی بودند در زمان چند جمله ای قابل حل می شوند.

از زمان غیر قطعی به زمان قطعی

تحلیل اثبات قضیه ۲-۳-۱ به افزایش نمایی در تعداد حالات اتوماتای حالت متناهی قطعی مورد نیاز برای شبیه سازی اتوماتای حالت متناهی غیر قطعی دلالت دارد. نتیجه زیر به افزایش نمایی مشابهی در تعداد حرکات که ماشین تورینگ قطعی برای شبیه سازی ماشین تورینگ غیر قطعی نیاز دارد اشاره می کند.

نتیجه ۵-۳-۱

برای هر مبدل تورینگ غیر قطعی M_1 مبدل تورینگ قطعی معادل M_2 با خصوصیات زیر وجود دارد.اگر M_1 روی ورودی x در t حرکت توقف کند , آنگاه M_2 روی چنین ورودی در ct^2 حرکت توقف می کند, که c فقط به M_1 بستگی دارد . علاوه بر این ,در چنین موردی M_2 حداکثر $2t$ مکان از روی هر نوار کار کمکی را ملاقات می کند.

اثبات

M_1 و M_2 را از قضیه ۴-۳-۱ در نظر می گیریم.فرض کنید M_1 روی ورودی x در t مرحله توقف می کند. آنگاه M_2 نیازمند است که فقط رشته های α درون $\{r,.....,1\}^*$ را که طولشان بیشتر از t یا $t+1$ نباشد را بررسی کند, بسته به آنکه آیا M_1 رشته x را به تر تیب می پذیرد یا رد می کند.تعداد چنین رشته های α بیشتر از $(r+1)^{t+1}$ نمی باشد.

برای هر رشته $\alpha=i_1\ldots i_j$ در $\{r,....,1\}^*$ مبدل تورینگ M_2 از تعدادی از حرکات که نسبت به j خطی هستند برای بدست آوردن α استفاده می کند و تلاش می کند دنباله حرکات به فرم $C_\varepsilon \vdash \ldots \vdash C_\alpha$ را شبیه سازی کند.در نتیجه M_2 تعدادی از حرکات خطی در $(r+1)^{t+1}t$ را نیاز دارد.بنا بر این ct^2 حدی برای تعداد حرکت ها است , زیرا برای هر مقدار صحیح مثبت v داریم: $v=2^{\log v}$.

به طور مشابه ، برای هر رشته $\alpha=i_1\ldots i_j$ مبدل تورینگ M_2 به j مکان در اولین نوار کار کمکی برای ذخیره رشته α نیاز دارد, و حداکثر به j مکان در هر یک از نوار های کار کمکی دیگر برای ثبت محتوای نوارهای متناظر از M_1 نیاز دارد. با قرار دادن هدهای نوارهای کار کمکی روی نخستین محل پیش از شروع به شبیه سازی M_1 روی α، مطمئن می شود که هد بیش از t محل از اولین محل آنها نمی رود.

مسائل صدق پذیری

قضیه در ذیل آمده وجود مسائل کامل NP را با مثال نشان می دهد.

تعاریف

عبارت منطقی عبارتی است که در زیر به صورت استقرایی تعریف شده است :

۱- ثابت های ۰(نادرست) و ۱(درست) عبارات منطقی می باشند.

۲- هر متغیر x یک عبارت منطقی میباشد.

۳- اگر E_1 و E_2 عبارات منطقی باشند نقیض $\neg E1$ و ترکیب فصلی $E1 \vee E2$,ترکیب عطفی $E1 \wedge E2$ و (E)عبارات منطقی هستند.

هر انتساب ۰و۱ به متغییر ها از عبارات منطقی مقداری را برای عبارت فراهم می کند.اگر E عبارت منطقی باشد آنگاه مقدار (E) همان مقدار E خواهد بود.اگر ارزش E,0باشد ارزش $\neg E$ برابر با ۱ خواهد بود واگرارزش E ،یک باشد ارزش نقیض آن ۰ خواهد بود.ارزش ترکیب فصلی $E1 \vee E2$ در صورتی ۱ می شود که ارزش یکی از آنها ۱ باشد ودر صورتی که ارزش هر دو آنها ۰ باشد ارزش ترکیب فصلی آنها نیز ۰ میباشد.ارزش ترکیب عطفی $E1 \wedge E2$، ۱ می باشد اگر هر دو آنهادارای ارزش ۱ باشند در غیر اینصورت ارزش آن ۰ خواهد بود.فرض می شود که میان عملگرهای منطقی نقیض , عطف و فصل عملگر نقیض بالا ترین اولویت را نسبت به عملگرهای عطف و فصل دارا میباشد.

عبارت منطقی صدق پذیر گفته می شود اگر متغیر های آن بتوانندبا گرفتن مقادیر ۰ یا۱ ارزش ۱ را برای عبارت تولید کنند.مسائل صدق پذیری هر عبارت منطقی داده شده را بررسی میکنند که آیا عبارت صدق پذیر می باشد یا نه ..

مثال ۵-۳-۱

عبارت منطقی $E = x_2 \wedge x_3 \wedge (\neg x_1 \wedge x_2)$ با هر انتساب $x_2=1$ و $x_3=1$ و یا هر انتساب $x_1=0$ و $x_2=1$ صدق پذیر است .. تمامی انتسابهای دیگر ارزش ۰ را

برای E فراهم می کند. $(x_2 \wedge x_3) \vee ((\neg x_1) \wedge x_2)$ یک نمونه کاملاً پرانتزگذاری شده از E می باشد.

$x \wedge \neg x$ مثالی ازیک عبارت منطقی صدق ناپذیر است.

اثبات قضیه زیر از شیوه کلی (عمومی) استفاده می کند.

قضیه ۵-۳-۱

مسئله صدق پذیری complete-NP است.

اثبات : صدق پذیری هر عبارت منطقی می توا ند در زمان چند جمله ای با انتساب غیر قطعی برخی مقادیر به متغیر های عبارت داده شده و سپس ارزیابی عبارت برای چنین انتسابی کنترل شود. بنابراین مسئله در NP است.برای نشان دادن اینکه مسئله صدق پذیری NPسخت (hard - NP) می باشد کافی است ثابت شود که هر مسئله K در NP یک مبدل تورینگ زمان محدود چند جمله ای T_k دارد . . به طوری که T_k مسئله Kرا به مسئله صدق پذیری کاهش می دهد. برای هدف اثبات هر مسئله K را در NP در نظر می گیریم .فرض می کنیم $M =< Q, \Sigma, \Gamma, \delta, q0, B, F >$ ماشین تورینگ غیر قطعی باشد که برای M در زمان $T(n) = O(n^k)$ تصمیم گیری می کند

m نشان دهنده تعداد نوارهای کارکمکی M باشد آنگاه T_k می تواند مبدل تورینگی باشد که روی ورودی x عبارت منطقی E_x را با ساختار زیر به خروجی می دهد.

ساختار E_x

عبارت منطقی E_x چگونگی محاسبه پذیرش M روی ورودی x را توصیف می کند. E_xبا انتساب داده شده صدق پذیر می باشد اگر وتنها اگر انتساب با محاسبه پذیرش $C_0 \vdash C_1 \vdash \ldots \vdash C_{T(|x|)}$ از M روی ورودی x متناظر باشد. عبارت ساختار زیر را دارد که $T(|x|) = t$.

$$E_{conf\,0} \wedge \ldots \wedge E_{conf\,t} \wedge E_{init} \wedge E_{rule\,1} \wedge \ldots$$
$$\wedge E_{rule\,t} \wedge E_{accept} \wedge E_{follow\,1} \wedge \ldots \wedge E_{follow\,t}$$

$E_{conf\,0} \wedge \ldots \wedge E_{conf}$ بیان کننده یک محاسبه پذیرش است که شامل دنباله پیکربندی های $C_0 , \ldots , C_t$ می باشد .

E_{init} پیکربندی اولیه .C را نشان می دهد.

$E_{rule_1} \wedge \ldots \wedge E_{rule_t}$ محاسبه پذیرشی را نشان می دهد که از دنباله Ψ از t قانون انتقال استفاده می کند . E_{accept} آخرین قاعده انتقال در Ψ را که به یک حالت پذیرش وارد می شود نشان می دهد.بدون از دست رفتن عمومیت ،فرض می کنیم که قاعده انتقال هم می تواند "NULL" باشد,یعنی ، قاعده انتقال روی M می تواند حرکتی بدون تغییر در پیکربندی داشته باشد.چنین فرضی به ما اجازه می دهد توجهمان را فقط به محاسباتی که دقیقا" شامل $T(|x|)$ حرکت هستند محدود کنیم. E_{follow_i} با استفاده M i امین قاعده انتقال در Ψ از پیکربندی C_{i-1} به پیکربندی C_i که $t \geq i \geq 1$ می رسد.

متغیر های E_x

عبارت منطقی E_x متغیرهایی به فرم $w_{i,r,j,X}$ و متغیرهایی به فرم $w_{i,\tau}$ را استفاده می کند.هر متغیر توصیفی درباره خصوصیت محاسبه پذیرش فراهم می کند. انتسابی که E_x را صدق پذیری می کند مقدار ۱ را برای آن متغیر هایی که جملاتشان برای محاسبه در سوال بر قرار است فراهم می کند و مقدار ۰ را برای آن متغیرهایی که جملاتشان برای محاسبه برقرار نیستند فراهم می کند.

$w_{i,r,j,X}$ بیان می کند که X در jامین کاراکتر از rامین نوار در i امین پیکربندی است که $1 \leq r \leq m$. $r = 0$ به نوار ورودی و $1 \leq r \leq m$ به r امین نوار کار کمکی اشاره می کند.

$w_{i,\tau}$ بیان می کند که τ قاعده انتقال در i امین حرکت از محاسبه می باشد.

ساختار E_{conf_i}

عبارت E_{conf_i} عطف عبارات منطقی زیر است.

آ . $\vee \{w_{i,0,j,X} \mid X \text{ is in } \Sigma Y \{c, \$\} Y Q\}$ برای $1 \leq j \leq |x| + 3$

این عبارت بیان می کند که یک پیکربندی دارای بخش ورودی با $|x|+3$ درایه است. و هر درایه حداقل یک نماد از $\Sigma Y \{c, \$\} Y Q$ دارد .

ب

.

$$\wedge \{ (w_{i,0,j,X} \wedge w_{i,0,j,Y}) \mid X \text{ and } Y \text{ are in } \Sigma Y \{c, \$\} Y Q \text{ and } X \neq Y \}$$

برای . $1 \le j \le |x|+31 \le$ این عبارت بیان می کند که هر درایه در بخش ورودی حد اکثر یک نماد دارد

پ . $\vee \{ (w_{i,0,j,X} \mid X \text{ is in } \Gamma\ Y\ Q \}$ برای $1 \le r \le m$ و $1 \le j \le t+1$

این عبارت بیان می کند که یک پیکربندی دارای m بخش نوار کار کمکی می باشد که هر بخش دارای $t+1$ درایه و هر درایه دارای حداقل یک نماد از $\Gamma\ Y\ \Sigma$ می باشد .

ت .

$$\wedge \{ (w_{i,r,j,X} \wedge w_{i,r,j,Y}) \mid X \text{ and } Y \text{ are in } \Gamma\ Y\ \ Q \text{ and } X \neq Y \}$$

برای $1 \le r \le m$ و $1 \le j \le t+1$.

این عبارت بیان می کند که هر درایه در بخش نوار کار کمکی حداکثر یک نماد دارد.

هر انتساب که عبارات در قسمت های آ و ب در بالا را صدق پذیر کنندرشته ای با طول $|x|+3$ می باشد . رشته متناظر با نوار ورودی M شامل سمبل های ورودی ، سمبل های نشانه گذار پایانی ¢ و $ و سمبل های حالت می باشد . به ویژه ، سمبل Xدرمکان j در رشته قرار دارد اگر و فقط اگر $w_{i,0,j,X}$ مقدار یک را بگیرد . بطور مشابه هر انتساب که عبارات در قسمت بالای پ و ت را برای مقدار خاص r صدق پذیر کند رشته ای با طول $t+1$ را فراهم می کند

که با r امین نوار کار کمکی از Mمتناظر است. این رشته از نمادهای نوار کار کمکی و نمادهای حالت تشکیل شده است. بخصوص این رشته حاوی نماد X در محل j است اگر وتنها اگر مقدار ۱ به $w_{i,r,j,X}$نسبت داده شده باشد.

ساختار E_{init}

عبارت E_{init} عطف سه عبارت منطقی زیر است.

آ . $w_{0,0,1,q} \wedge w_{0,0,2,q_0} \wedge \{ w_{0,0,j+2,a_j} \mid 1 \le j \le |x| \} \wedge w_{0,0,|x|+3,\$}$

این عبارت نشان می دهد که در پیکربندی اولیه بخش ورودی از رشته $¢q_0a_1 \ldots a_n\$$ که a_j ، j امین نماد ورودی در x است ، تشکیل شده است .

ب . $\vee \{w_{0,r,j,q_0} \mid 1 \le j \le t+1\}$ برای $1 \le r \le m$.

این عبارت نشان می دهد که در پیکربندی اولیه هر بخش نوار کار کمکی شامل حالت اولیه q_0 می باشد .

پ . $w_{0,r,j,B} \vee w_{0,r,j,q_0} \wedge \{ w_{0,r,s,B} \mid 1 \le s \le t+1 \text{ and } s \ne j \}$ برای $1 \le j \le t+1$ و $1 \le r \le m$.این عبارت نشان می دهد که در پیکربندی اولیه هر بخش نوار کار کمکی از نماد فضای خالی B وحداکثر حضوری از q_0 تشکیل شده است .هر انتسابی که E_{init} را صدق پذیر کند متناظر است با پیکربندی اولیه از M روی ورودی x .به علاوه هرکدام $E_{conf\,0}$ را نیز ارضا می کنند .

ساختار E_{rule_i} ، E_{accpt}

عبارت E_{rule_i} عطف دو عبارت منطقی زیر است.

آ . $\vee \{ w_{i,\tau} \mid \tau \text{ is in } \delta \}$

ب . $\wedge \{ (w_{i,\tau_1} \wedge w_{i,\tau_2}) \mid \tau_1, \tau_2 \text{ are in } \delta \text{ and } \tau_1 \ne \tau_2 \}$

عبارت قسمت آ نشان می دهد که حداقل یکی از متغیر های $w_{i,\tau}$,مقدار ۱ دارد. عبارت قسمت ب نشان می دهد که برای هر انتساب که E_{rulei}را صدق پذیر می کند حداکثر یک متغیر از $w_{i,\tau}$ دارای مقدار یک می باشد . بنابراین هر انتسابی که E_{rulei} را صدق پذیر کند مقدار ۱رابه دقیقاً یک متغر از $w_{i,\tau}$نسبت می دهد. برای مثال به متغیری که با قاعده انتقال τ که در i امین حرکت محاسبه استفاده شده، متناظر است . عبارت E_{accept} به صورت $\vee \{ w_{t,\tau} \mid \tau \text{ takes M into an accepting state} \}$ می باشد.

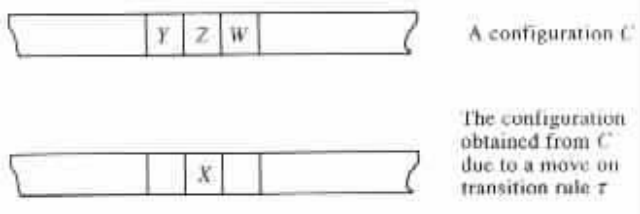
ساختار E_{follow_i}

عبارت E_{follow_i}عطف عبارات منطقی زیر است.

آ . $\vee \{ (w_{i,0,j,X} \wedge w_{i-1,0,j-1,Y} \wedge w_{i-1,0,j,Z} \wedge w_{i-1,0,j+1,W} \wedge w_{i,\tau}) \mid X, Y, Z, W \text{ and } \tau \text{ such that } X = f_0(Y,Z,W,\tau) \}$ برای $1 \le j \le |x|+3$.

ب . $\vee \{ (w_{i,r,j,X} \wedge w_{i-1,r,j-1,Y} \wedge w_{i-1,r,j,Z} \wedge w_{i-1,r,j+1,W} \wedge w_{i,\tau}) \mid X, Y, Z, W \text{ and } \tau \text{ such that } X = f_r(Y,Z,W,\tau) \}$ برای $1 \le r \le m$ و $1 \le j \le t+1$.

که $f_r(Y,Z,W,\tau)$ تابعی است که جانشینی X برای سمبل Z را در پیکربندی تعیین می کند که از به کارگیری قانون انتقال τ نتیجه می شود .(شکل ۵-۳-۲ را ببینید)



شکل ۵-۳-۲ مقدار X از $f_r(Y,Z,W,\tau)$.

فرض می شود که Z بین Y از سمت چپ و W از سمت راست محصور شده است .

و .

$W_{i-1,0,0,Y}$, . . . , $W_{i-1,m,0,Y}$, $W_{i-1,0,|x|+4,W}$, $W_{i-1,1,t+2,W}$, . . . , $W_{i-1,m,t+2,W}$

متغیر های جدید هستند . آنها معرفی شده اند تا با موارد مرزی را که در آنها سمبل Z در $f_r(Y,Z,W,\tau)$ متناظر است با سمبل انتهایی نوار، کار کنند

اگر $\tau =(q, a, b_1, . . . , b_m, p, d_0, c_1, d_1, . . . , c_m, d_m)$ آنگاه مقدار X از تابع $f_r(Y,Z,W,\tau)$ ، $X= p$ را زمانی که موارد زیر برقرار باشند،ارضا می کند.

. a $Z = q$ and $d_r = 0$
.b $Y = q$ and $d_r = +1$
.c $W = q$ and $d_r = -1$

بطور مشابه، $X = c_r$ زمانی ارضا می کند که موارد زیر برقرار باشند: $1 \le r \le m$.

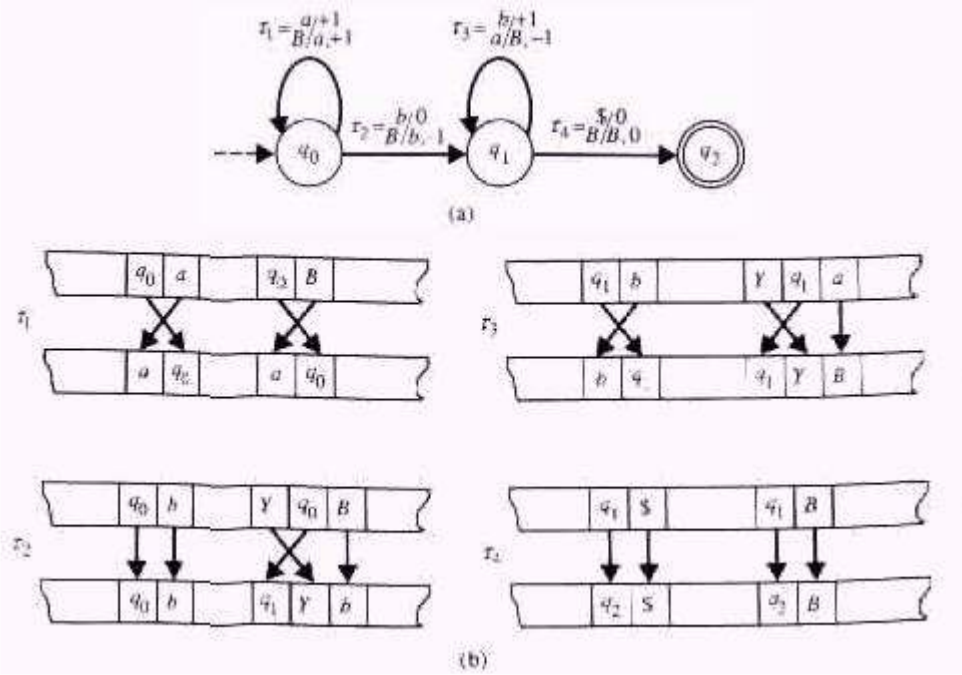
. a $Z = q$, $W = b_r$, and $d_r = +1$
. b $Y = q$, $Z = b_r$, and $d_r = 0$
. c $Y = q$, $Z = b_r$, and $d_r = -1$

به عبارت دیگر

. a $X = W$ whenever $Z = q$, $r = 0$, and $d_0 = +1$
. b $X = Y$ whenever $Z = q$ and $d_r = -1$

در سایر موارد $X = Z$ است زیرا هد r امین نوار از Z خیلی دور است. در این صورت نتیجه به دست می آید زیرا T_k روی ورودیx می تواند محاسبه $t= T(|x|)$ را در زمان چند جمله ای انجام دهد و E_x (رشته ای که آن را نمایش دهد)را به خروجی بفرستد .

مثال ۵-۳-۲ M ماشین تورینگ شکل ۵-۳-۳ است .



شکل ۵-۳-۳ (a یک ماشین تورینگ M (b تاثیری که حالات بالا روی پیکربندی از M دارند.

پیچیدگی زمانی M از T(n) = n+2 است .روی ورودی x=ab ماشین تورینگ M یک محاسبه پذیرش $C_0 \vdash C_1 \vdash C_2 \vdash C_3 \vdash C_4$ از t=4 حرکت را دارد که هر C_i یک پیکربندی(uqv, u'qv')است که $uv = ¢ab\$$ و $|u'v'| \leq t$ را ارضا می کند .

بااستفاده از نمادگذاری در اثبات قضیه ۵-۳-۱ در نا مساوی های زیر برای Mو x در بالا برقرار هستند :

$$E_{init} = W_{0,0,1,c} \wedge W_{0,0,2,q_0} \wedge W_{0,0,3,a} \wedge W_{0,0,4,b} \wedge W_{0,0,5,\$}$$
$$\wedge (W_{0,1,1,q_0} \vee W_{0,1,2,q_0} \vee W_{0,1,3,q_0} \vee W_{0,1,4,q_0} \vee W_{0,1,5,q_0})$$
$$\wedge (W_{0,1,1,B} \vee W_{0,1,1,q_0} \wedge W_{0,1,2,B} \wedge W_{0,1,3,B} \wedge W_{0,1,4,B} \wedge W_{0,1,5,B})$$
$$\wedge (W_{0,1,2,B} \vee W_{0,1,2,q_0} \wedge W_{0,1,1,B} \wedge W_{0,1,3,B} \wedge W_{0,1,4,B} \wedge W_{0,1,5,B})$$
$$\wedge (W_{0,1,3,B} \vee W_{0,1,3,q_0} \wedge W_{0,1,1,B} \wedge W_{0,1,2,B} \wedge W_{0,1,4,B} \wedge W_{0,1,5,B})$$
$$\wedge (W_{0,1,3,B} \vee W_{0,1,4,q_0} \wedge W_{0,1,1,B} \wedge W_{0,1,2,B} \wedge W_{0,1,3,B} \wedge W_{0,1,5,B})$$
$$\wedge (W_{0,1,5,B} \vee W_{0,1,5,q_0} \wedge W_{0,1,1,B} \wedge W_{0,1,2,B} \wedge W_{0,1,3,B} \wedge W_{0,1,4,B})$$

$$E_{rule\,i} = (W_{i,\tau 1} \vee W_{i,\tau 2} \vee W_{i,\tau 3} \vee W_{i,\tau 4})$$
$$\wedge (W_{i,\tau 1} \wedge W_{i,\tau 2})$$
$$\wedge (W_{i,\tau 1} \wedge W_{i,\tau 3})$$
$$\wedge (W_{i,\tau 1} \wedge W_{i,\tau 4})$$
$$\wedge (W_{i,\tau 2} \wedge W_{i,\tau 3})$$
$$\wedge (W_{i,\tau 2} \wedge W_{i,\tau 4})$$
$$\wedge (W_{i,\tau 3} \wedge W_{i,\tau 4})$$
$$F_{accept} = W_{4,\tau 4}$$

$$\left.\begin{array}{lcl} f_0(q_1, b, W, \tau_3) & = & q_1 \\ f_0(Y, q_1, b, \tau_3) & = & b \\ f_0(Y, Z, q_1, \tau_3) & = & Z \\ f_0(Y, Z, W, \tau_3) & = & Z \\ f_1(q_1, a, W, \tau_3) & = & B \\ f_1(Y, q_1, a, \tau_3) & = & Y \\ f_1(Y, Z, q_1, \tau_3) & = & q_1 \\ f_1(Y, Z, W, \tau_3) & = & Z \end{array}\right\} \text{For all } Y, Z, W \text{ in } \{a, b, B, ¢, \$\}$$

شکل ۵-۳-۳ قسمت b تغییرات در پیکربندی های M را نشان می دهد که ناشی از قوانین انتقال ذکر شده در بالاست .

مسئله صدق پذیری سه تایی

یک تغییر ناچیزدر اثبات قبلی ، به NP کامل بودن نسخه محدودشده ای از مسئله صدق پذیری دلالت می کند.

تعریف : یک عبارت بولین ،یک لیترال است گفته می شود اگر به صورت یک متغیر یا منفی یک متغیر باشد.یک عبارت بولین ، یک جمله گفته می شود اگر فصلی ($\vee$) از لیترال ها باشد .یک عبارت بولین به فرم نرمال عطفی است اگر به صورت عطف ($\wedge$) جملات باشد .یک عبارت بولین به فرم نرمال عطفی k تایی گفته میشود اگر به فرم نرمال عطفی باشد و هر یک از جمله های آن شامل دقیقا k لیترال باشد.

مسئله صدق پذیری k تایی این است که آیا برای عبارت بولین داده شده به فرم نرمال عطفی k تایی ،عبارت صدق پذیر است یا نه ؟

بدون از بین رفتن کلیت , در ادامه فرض می شود که متغیر نمی تواند بیشتر از یکبار در هر جمله داده شده ظاهر شود.

قضیه ۵-۳-۲ : مسئله صدق پذیری سه تایی NP کامل است.

اثبات : عبارت E_x در اثبات از قضیه ۵-۳-۱ با تغییرات جزئی به فرم نیاز نرمال عطفی سه تایی تبدیل می شود .

a . به جز عبارت $\underline{E_{follow\ i}}$ و قسمت c از E_{init} , تمام عبارات دیگر می توانند با استفاده از هم ارزی $\neg(w_1 \wedge w_2) \equiv (\neg w_1) \vee (\neg w_2)$ به فرم نرمال عطفی تبدیل شوند .

b .هر عبارت در $\underline{E_{follow\ i}}$ و قسمت c از E_{init} می تواند با استفاده از هم ارزی $w_1 \vee (w_2 \wedge w_3) \equiv (w_1 \vee w_2) \wedge (w_1 \vee w_3)$ به نحوی تغییر کند که به فرم نرمال عطفی سه تایی در آید .

c . هر ترکیب فصلی به فرم $w_1 \vee \ldots \vee w_s$ با $s>3$ جمله می توانند تغییر یابند که در فرم نرمال عطفی سه تایی باشند این کار به وسیله جایگزینی مکرر زیر عبارت ها به فرم $w_1 \vee \ldots \vee w_s$ با زیر عبارتهایی به

فرم $(w_1 \vee w_2 \vee w) \wedge (\neg w \vee w_3 \vee ... \vee w_s)$ انجام می شود که w ها متغیر های جدید هستند .

نتیجه NP کامل بودن مسئله صدق پذیری در مطالعه مسائل به دو دلیل دارای اهمیت است : ۱) این مسئله وجود یک مسئلهNP کامل را نشان میدهد. ۲) این مسئله در نشان دادن NP سخت بودن مسائل دیگر مفید است.

٤-٥ مسائل NP کامل دیگر

مسئله ...NP سخت بودن مسئله صدق پذیری با نشان دادن وجود یک کاهش چندجمله ای از هر مسئله درون NP به مسئله صدق پذیری مشخص می شود .راهکار مشابهی برای نشان دادن NP سخت بودن مسئله صدق پذیری سه تایی استفاده می شود . اگر چه ، به طور کلی ، اثبات NP سخت بودن یک مسئله داده شده نیاز به جامعیت نداردادما می تواند با کاهش زمان چند جمله ای از مسائل NP سخت دیگر انجام شود .یک اثبات به وسیله کاهش امکان پذیر است زیرا ترکیبی از کاهش های با زمان چند جمله ای یک کاهش با زمان چند جمله ای است . یعنی اگر یک مسئله K_a به یک مسئله K_b در زمان $T_1(n)$ قابل کاهش باشد و K_b به یک مسئله K_c در زمان$T_2(n)$ قابل کاهش باشد آنگاه K_a به K_c در زمان $T_2(T_1(n))$ قابل کاهش است . به علاوه $T_2(T_1(n))$ چند جمله ای است اگر $T_1(n)$ و$T_2(N)$ هم چند جمله ای باشند.

مسئله کوله پشتی صفرو یک

اثبات هایی که در پی دو قضیه می آیند , NP سخت بودن مسائل در سؤال ها را به وسیله کاهش نمایش می دهند.

قضیه ٥-٤-١

مسئله کوله پشتی صفر و یک که یک مسئلهNP کامل است و با دوتایی زیر تعریف می شود :

دامنه : $\{(a_1.....a_N,b) \mid N \geq 1 \text{ and } a_1,.....,a_N,b)\}$ اعداد طبیعی هستند.

سؤال :

ایا وجود دارد $v_1.... v_N$ در {۰و۱} چنان که برای نمونه داده شده $(a_1....a_N,b)$، $a_1v_1+.....a_Nv_N=b$ باشد؟

اثبات:

یک ماشین تورینگ M که روی هر نمونه داده شده $(a_1.....a_N,b)$از مسئله غیر قطعی , مقادیری از {۰و۱} را به $v_1.....v_N$منسوب می کند, بررسی می کند که آیا $a_1v_1+.....a_Nv_N=b$ هست یا نه , و ورودی را می پذیرد اگر وتنها اگر در آن تساوی برقرار باشد. M میتواند دارای پیچیدگی زمانی چند جمله ای باشد. بنابراین مسئله کوله پشتی صفر و یک,در NP است.

برای نشان دادن اینکه مسئله کوله پشتی صفرویک NPسخت است هر نمونه E از مسئله صدق پذیری سه نایبرا در نظر بگیرید. بگذارید $x_1.....x_m$ نشان دهنده متغیر ها در عبارت بولین E باشد. E به صورت عطف $c_1 \wedge ... \wedge c_k$از بعضی جمله های $c_1,.....,c_k$میباشد .هر C_i فصل $c_{i1} \vee c_{i2} \vee c_{i3}$ از بعضی لیترال های$c_{i3}\ c_{i1},c_{i2},$ است. هر c_{ij} یک متغیر x_t , یا نقیض $\neg x_t$ از یک متغیر x_t برای بعضی $1 \le t \le m$ است.

از عبارت بولین تا یک سیستم معادلات خطی

از عبارت بولین E می توان یک سیستم S از معادلات خطی به صورت زیر ساخت :

$$x_1 + \overline{x}_1 = 1$$

$$M$$

$$x_m + \overline{x}_m = 1$$

$$c_{11} + c_{12} + c_{13} + y_{11} + y_{12} = 3$$

$$M$$

$$c_{k1} + c_{k2} + c_{k3} + y_{k1} + y_{k2} = 3$$

سیستم S دارای متغیرهای $x_1,\ldots,x_m,\bar{x}_1,\ldots,\bar{x}_m,y_{11},\ldots,y_{k2}$ می باشد . .متغیر x_t در S متناظر است با لیترال x_t در E.متغیر $\bar{x}_m$ در S متناظر است با لییترال $\neg x_t$ در E. c_{ij} نماینده متغیر x_t در S اگر j امین لیترال در C_i باشد. c_{ij} نماینده متغیر $\bar{x}_m$ در S است اگر $\neg x_t$ ، j امین لیترال در C_i باشد.

هر معادله به شکل $x_i+\bar{x}_i=1$ یک راه حلی روی $\{۰و۱\}$ دارد اگر و تنها اگر یا $x_i=1$ $\bar{x}_i=1$ or $x_i=0$ and $\bar{x}_i=0$ and . هر معادله از شکل

$c_{i1}+c_{i2}+c_{i3}+y_{i1}+y_{i2}=3$ یک راه حل روی $\{۰و۱\}$ دارد اگر و تنها اگر حداقل یکی از تساوی های $c_{i1}=1, c_{i2}=1,$ and $c_{i3}=1$ برقرار باشد . این نتیجه به دست می آید که سیستم S یک راه حل روی $\{۰و۱\}$ دارد اگر وتنها اگر عبارت بولین E صدق پذیر باشد.

از سیستم معادلات خطی تا نمونهای از مسئله کوله پشتی ۰-۱

سیستم S می تواند در برداری مثل زیر، نمایش داده شود

$$\begin{pmatrix} a_{11} \\ M \\ a_{m+k\,1} \end{pmatrix} z_1 + \Lambda + \begin{pmatrix} a_{1\,2m+2k} \\ M \\ a_{m+k\;2m+2k} \end{pmatrix}$$

متغییر $z_1,\ldots,z_{2m+2k}$ در بردار ، نماینده متغییرهای $x_1,\ldots,x_m,\bar{x}_1,\ldots,\bar{x}_m,y_{11},\ldots,y_{k2}$ از S به ترتیب بودند.a_{ij}راضریب z_j در i مین معادله ازS فرض می کنیم. b_i را ثابتی در طرف سمت راست i امین معادله در S فرض می کنیم.

به طور مشابه ، سیستمS میتواند همچنین بوسیله معادله H مطابق زیر نمایش داده شود.

$$a_1 z_1 + \ldots + a_{2m+2k}\, z_{2m+2k} = \delta$$

در H و هر a_jنماینده عدد صحیحی است که نمایش اعشاری آن به صورت $a_{1j}\ldots a_{m+k\,j}$می باشد. به طور مشابه، b نماینده عددصحیحی است که نمایش اعشاری آن به صورت $b_1\ldots b_{m+k}$ می باشد. نما یش ممکن هست زیرا جمع $a_{i1},\ldots,a_{i(2m+2k)}$ یا با ۲برابراست یاباه برای $1\le i\le m+k$. یعنی، i امین رقم در جمع $c=a_1+\ldots+a_{2m+2k}$ فقط به i امین رقم از $a_1,\ldots,a_{2m+2k}$ بستگی دارد.نتیجه می شود که S روی $\{0,1\}$صدق پذیر است اگر وتنها اگر H روی $\{0,1\}$ صدق پذیر باشد.به عنوان نتیجه ، نمونه E از مسئله صدق پذیری ، صدق پذیر است اگر و تنها اگر نمونه $(a_1 , \ldots , a_{2m+2k} , b)$ از مسئله کوله پشتی صفر و یک ، یک پاسخ مثبت داشته باشد. بعلاوه یک مبدل تورینگ قطعی با محدودیت زمانی چند جمله ای می تواند نمونه متناظری از مسئله کوله پشتی ۰-۱ برای هر نمونه E از مسئله صدق پذیری سه تایی بسازد. در نتیجه، NPسخت بودن از NP سخت بودن مسئله صدق پذیری سه تایی ، نتیجه می شود .

مثال ۵-٤-۱ عبارت بولیین E مطابق زیر را در نظر بگیرید .

$$(x_1 \vee x_2 \vee \neg x_3) \wedge (\neg x_2 \vee x_3 \vee \neg x_4) \wedge (x_1 \vee x_3 \vee x_4) \wedge (\neg x_1 \vee x_2 \vee x_4)$$

E نمونه ای از مسئله صدق پذیری سه تایی می باشد.عبارت بولین صدق پذیر است اگر وتنها اگر سیستمS زیر از معادلات خطی یک راه حل روی $\{0,1\}$ داشته باشد.

$$\begin{bmatrix}1\\0\\0\\0\\1\\0\\1\\0\end{bmatrix}x_1+\begin{bmatrix}0\\1\\0\\0\\1\\0\\0\\1\end{bmatrix}x_2+\begin{bmatrix}0\\0\\1\\0\\0\\1\\1\\0\end{bmatrix}x_3+\begin{bmatrix}0\\0\\0\\1\\0\\0\\1\\1\end{bmatrix}x_4+\begin{bmatrix}1\\0\\0\\0\\0\\0\\0\\1\end{bmatrix}\bar{x}_1+\begin{bmatrix}0\\1\\0\\0\\0\\0\\1\\0\end{bmatrix}\bar{x}_2+\begin{bmatrix}0\\0\\1\\0\\1\\0\\0\\0\end{bmatrix}\bar{x}_3+\begin{bmatrix}0\\0\\0\\1\\0\\1\\0\\0\end{bmatrix}\bar{x}_4+$$

$$\begin{bmatrix}0\\0\\0\\0\\1\\0\\0\\0\end{bmatrix}y_{11}+\begin{bmatrix}0\\0\\0\\0\\1\\0\\0\\0\end{bmatrix}y_{12}+\begin{bmatrix}0\\0\\0\\0\\0\\1\\0\\0\end{bmatrix}y_{21}+\begin{bmatrix}0\\0\\0\\0\\0\\1\\0\\0\end{bmatrix}y_{22}+\begin{bmatrix}0\\0\\0\\0\\0\\0\\1\\0\end{bmatrix}y_{31}+\begin{bmatrix}0\\0\\0\\0\\0\\0\\1\\0\end{bmatrix}y_{32}+\begin{bmatrix}0\\0\\0\\0\\0\\0\\0\\1\end{bmatrix}y_{41}+\begin{bmatrix}0\\0\\0\\0\\0\\0\\0\\1\end{bmatrix}y_{42}=\begin{bmatrix}1\\1\\1\\1\\3\\3\\3\\3\end{bmatrix}$$

از سویی دیگر، سیستم S یک حل روی {۰،۱} دارد اگر و تنها اگر معادله H مطابق زیر یک حل روی {۰،۱}داشته باشد .

صفرهایی که در ابتدا آمده اند در ثابتهای H نادیده گرفته می شود.

$$\begin{aligned}&10001010\,x_1+1001001\,x_2+100110\,x_3+10011x_4\\&+10000001\,\bar{x}_1+1000100\,\bar{x}_2+101000\,\bar{x}_3+10100\,\bar{x}_4\\&+1000\,y_{11}+1000\,y_{12}\\&+100\,y_{21}+100\,y_{22}\\&+10\,y_{31}+10\,y_{32}\\&+y_{41}+y_{42}\qquad\qquad=11113333\end{aligned}$$

عبارت E صدق پذیر است اگر وتنها اگر نمونه

(10001010,1001001,100110,10011,10000001,1000100,101000,1
0100,1000 1000,100,100,10,10,1,1,11113333)

از مسئله کوله پشتی ۰-۱ یک حل مثبت داشته باشد.

مسئله

مثال های قبلی از مسائل NP کامل با عبارات منطقی و معادلات خطی سرو کار داشتند . مثال زیر به گراف ها مربوط می شود . م

قضیه ۵-٤-۲ : مسئله clique که یک مسئله NP کامل است ، به وسیله جفت زیر تعریف می شود :

دامنه:

{ Gیک گراف و k یک عدد طبیعی است} | (G,K)

سوال:

آیا دسته بندی به اندازهK برای نمونه داده شده (G,K)وجود دارد؟(یک د سته بندی، یک زیر گراف است که بین هر جفت از گره ها آن یک یال وجود داشته باشد. تعداد گره ها در یک دسته بندی ، اندازه دسته بندی نامیده می شود.)

اثبات: ماشین تورینگ M که در نمونه داده شده (G,K) از مسئله دسته بندی به صورت زیر تعریف می شود را در نظر بگیرید . M با انتخاب غیر قطعی k گره از G شروع به کار می کند.سپس تعیین می کند که آیا بین هر جفت از گره های انتخاب شده، یالی وجود دارد یا نه. اگر چنین بود سپس M ورودی را می پذیرد در غیر اینصورت ورودی را رد می کند. M دارای پیچیدگی زمانی چند جمله ای است . در نتیجه مسئله دسته بندی در NP است.

برای نشان دادن NPسخت بودن مسئله دسته بندی ، هر نمونه E از مسئله صدق پذیری سه تایی را در نظر بگیرید . همانطور که در اثبات نتیجه قبلی آمد ، $x_1,\ldots,x_m$ متغییرهای عبارت بولین E را نشان می دهند . E به صورت عطف $c_1 \wedge \ldots \wedge c_k$ از جملات $c_1,\ldots,c_k$ است . هر C_i فصل $c_{i1} \vee c_{i2} \vee c_{i3}$ از لیترال های c_{i1} و c_{i2} و c_{i3} می باشد .هر c_{ij} یک متغیر x_t یا $\neg x_t$ ، برای برخی $1 \leq t \leq m$ می باشد .برای هر عبارت منطقی E یک گراف G با روش زیر می تواند ساخته شود :

گراف G گره ای متناظر با هر جفت $(c_i,(d_1,d_2,d_3))$ از انتساب (d_1,d_2,d_3)دارد که جملهC_i را ارضا می کند . گره ای که متناظر با جفت $(c_i,(d_1,d_2,d_3))$ با مجموعه

$\{x_{i1}=d_1,x_{i2}=d_2,x_{i3}=d_3\}$بر چسب گذاری می شود که x_{i1},x_{i2},x_{i3}متغییرهای

مورد استفاده در c_{i1},c_{i2},c_{i3} فرض می شوند. نتیجه می شود کهبرای هرC_i ، گراف G دارای هفت گره مرتبط است .

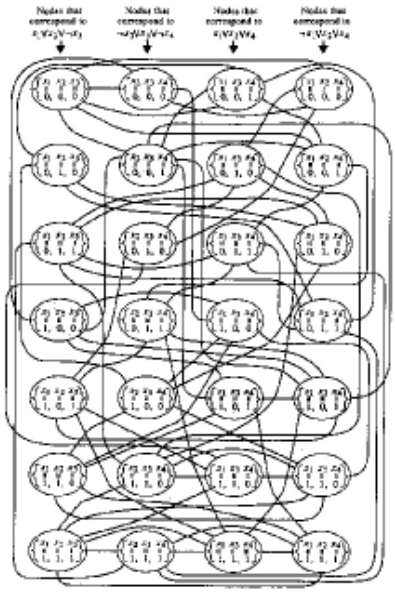
گراف G یک یال بین گره برچسب گذاری شده با مجموعه $\{x_{i1}=d_1,x_{i2}=d_2,x_{i3}=d_3\}$ و گره برچسب گذاری شده با مجموعه $\{x_{j1}=d'_1,x_{j2}=d'_2,x_{j3}=d'_3\}$ دارداگر وتنها اگر x_t دارای انتساب های مخالف در دو مجموعه باشد $1 \le t \le m$.

بین هیچ جفتی از گره ها با جمله مشترک C_i یالی وجود ندارد .به بیان دیگر ، یال های بین گره ها که با هر جمله متناظرند ،انتساب هایی از متغیر ها را به هم مربوط می کنند که هر دو جمله را هم زمان ارضا کنند .در نتیجه ، عبارت منطقی E صدق پذیر است اگر و فقط اگر G دسته ای از اندازه k داشته باشد .

یک مبدل تورینگ قطعی با محدودیت زمانی چند جمله ای می تواند در یک روش مشابه ، نمونه متناظر (G,K) از مسئله دسته بندی را برای هر نمونه E از مسایل صدق پذیری سه تایی معلوم میکند. بنابراین NP سخت بودن مسئله دسته بندی مشخص می شود .

مثال ۵-٤-۲ عبارت منطقی E به صورت زیر است . گراف G در شکل ۵-٤-۱ نشانداده شده است .

$$(x_1 \vee x_2 \vee \neg x_3) \wedge (\neg x_2 \vee x_3 \vee \neg x_4) \wedge (x_1 \vee x_3 \vee x_4) \wedge (\neg x_1 \vee x_2 \vee x_4)$$



شکل ۵-٤-۱ یک گراف G که انتساب هایی که جملات را در عبارت بولین $(x_1 \vee x_2 \vee \neg x_3) \wedge (\neg x_2 \vee x_3 \vee \neg x_4) \wedge (x_1 \vee x_3 \vee x_4) \wedge (\neg x_1 \vee x_2 \vee x_4)$ صدق پذیر می کنند ، به هم متصل کرده است .

بوسیله اثبات آخرین قضیه، E صدق پذیر است اگروتنها اگر (G,4) صدق پذیر باشد.انتساب $(x_1,x_2,x_3,x_4)=(1,0,0,1)$ که E را صدق پذیر می کندمتناظر است بادسته بندی در G که گره های آن هاشور خورده اند .

از تعریف NP کامل بودن برمی آید که P با NP برابر است اگروتنها اگر یک مسئله NP کامل در P وجود داشته باشد .

لازم به ذکر است که الگوریتم های شناخته شده برای مسائل NP کامل براساس جستجوی جامع روی تمام دامنه بنا شده اند .برای مثال ، در مورد مسئله صدق پذیری ، یک جستجوی جامع برای هر انتساب به متغیر ها که عبارت داده شده را ارضا می کنند ، ساخته می شود . در مورد مسئله کوله پشتی صفر و یک ، جستجوی جامع برای زیر مجموعه ای از مجموعه چندگانه $\{a_1,...,a_N\}$ کهمجموع مقادیر به مقدار داده شده b برسد ، ساخته می شود . در مورد مسئله دسته بندی جستجوی جامع برای دسته بندی با اندازه دلخواه ساخته شده است. درهمه این موارد جستجو روی دامنه ای با اندازه نمایی انجام می شود و به نظر می رسد که این روش تاکنون بهترین روش ممکن برای مسائل NP کامل است .

۵-۵ مکان چند جمله ای

با استنباط از نتیجه ۵-۳-۱ $NTIME(T(n)) \subseteq DSPACE(T(n))$ و همچنین PSPACE شامل NP است .

بعلاوه از قضیه ۵-۵-۱ نتیجه می شود که PSPACE در داخل EXPTIME می باشد. این محتویات پیشنهاد می کند که PSPACE به طور مشابه با NP، مورد مطالعه قرار گیرد. به ویژه چنین مطالعاتی برای امکان بعیدمساوی بودن NP و P-- اهمیت دارد .به همین دلیل مطالعهNP در مکان اول اهمیت بود . بهر حال اگرNP متفاوت از P از کار در آید ، مطالعهPSPACE ممکن است بینشی نسبت به عوامل افزایش پیچیدگی مسائل ، فراهم کند .

لم ۵-۵-۱: یک ماشین تورینگ M با محدودیت مکانی $S(n) \geq \log n$ می تواند

حداکثر به $2^{dS(n)}$ پیکربندی در ورودی داده شده به طول n دست یابد. فرض می شود که d ثابتی است که تنها به M بستگی دارد.

اثبات: هر ماشین تورینگ $M=<Q,\Sigma,\Gamma,\delta,q_0,B,F>$ با پیچیدگی مکانی $S(n) \geq \log n$ را در نظر می گیریم. برای ورودی x به طول n ماشین تورینگ می تواند حداکثر $|Q|(n+2)(S(n)|\Gamma|^{S(n)})^m$ پیکربندی مختلف داشته باشد. |Q| به تعداد حالتهای M اشاره می کند و m به تعداد نوارهای کاری کمکی از M اشاره می کند و $|\Gamma|$ به اندازه الفبای نوار –کاری کمکی Γ از M اشاره می کند.

عامل |Q| دیده می شود زیرا در پیکربندی های $(uqv,u_1qv_1,...,u_mqv_m)$ که uv= x$c را ارضا می کند حالت q از مجموعه Q با اندازه |Q| بدست می آید. عامل n+2 دیده می شود زیرا مکان هد ورودی |u| می تواند در n+2 مکان باشد. S(n) نماینده تعداد مکان های ممکن برای هد نوار کمکی است و $|\Gamma|^{S(n)}$ نماینده تعداد رشته های مختلف که در نوا کاری کمکی ذخیره می شود می باشد.

عبارت $|Q|(n+2)(S(n)|\Gamma|^{S(n)})^m$ که دارای ثابت d است به طوری که

$|Q|(n+2)(S(n)|\Gamma|^{S(n)})^m = 2^{\log|Q|}2^{\log(n+2)}(2^{\log S(n)}2^{S(n)\log|\Gamma|})^m \leq 2^{dS(n)}$

برای همه n اگر $S(n)\geq \log n$.

از مکان غیر قطعی به زمان قطعی

بااستنباط از نتیجه ۵-۳-۱، زمان غیر قطعی وقطعی رابطه $NTIME(T(n)) \subseteq \bigcup_{c>0} DTIME(2^{cT(n)})$ را ارضا می کند. قضیه زیر تصحیحی برای این نتیجه است زیرا $NTIME(T(n)) \subseteq NSPACE(T(n))$.

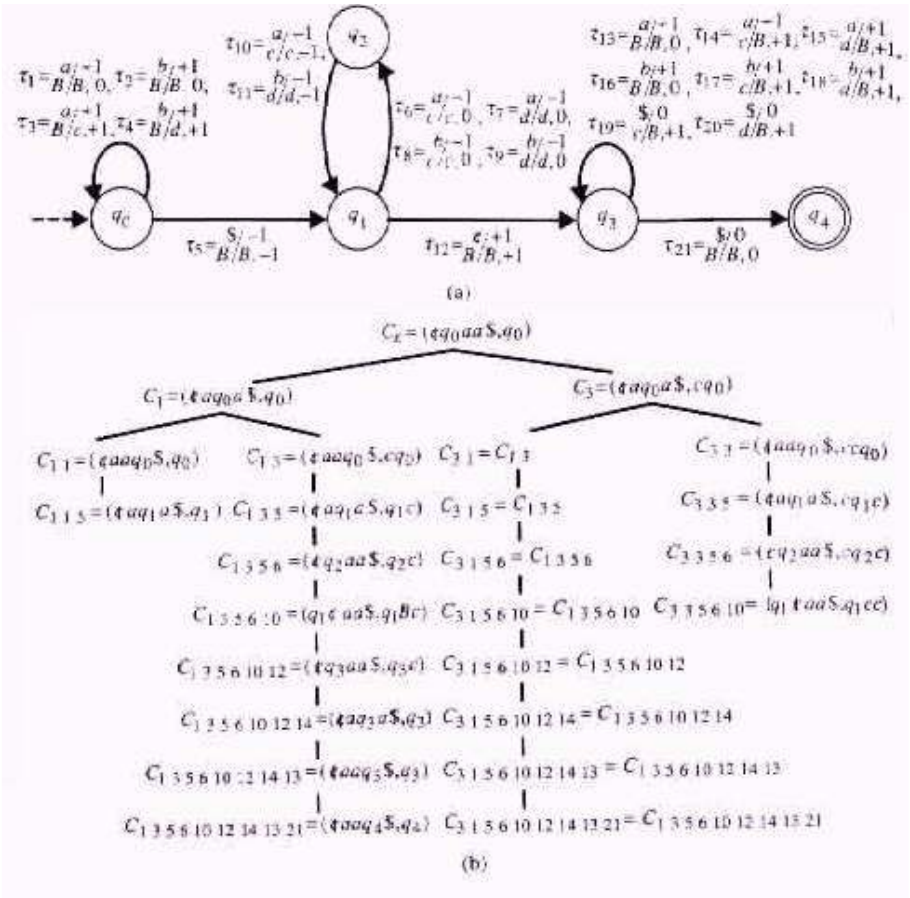
تعریف: درخت پیکربندی های یک ماشین تورینگ M روی ورودی X شاید یک درخت نامتناهی Ω است که به صورت زیر تعریف می شود . ریشه Ω گره ای است که با پیکربندی اولیه M روی ورودی X برچسب خورده است . یک گره در Ω که با پیکربندی C_1 برچسب خورده است ، گره بلافاصله بعدی آن با پیکربندی C_2 برچسب گذاری شده اگر و فقط اگر $C_1 \vdash C_2$.

قضیه ۵-۵-۱: اگر $S(n)\geq \log n$ آنگاه $NSPACE(S(n)) \subseteq \bigcup_{c>0} DTIME(2^{cS(n)})$

اثبات: هر ماشین تورینگ $M_1=<Q,\Sigma,\Gamma,\delta,q_0,B,F>$ با محدودیت مکانی S(n) را در نظر می گیریم . یک ماشین تورینگ قطعی M_2 می تواند تعیین شود اگر M_1

ورودی داده شده x را با تعیین اینکه درخت پیکربندی های Ω از M_1 روی ورودی x ،حاوی پیکربندی پذیرش است یا نه ، بپذیرد. M_2 برای انجام این کار مجموعه A از همه پیکربندی های در Ω را تشکیل داده و سپس بررسی کند که آیا این مجموعه حاوی پیکربندی پذیرش است یا نه . مجموعه A با الگوریتم زیر تولید می شود .

مرحله ۱: مجموعه A در ابتدا فقط شامل پیکربندی اولیه M_1 روی ورودی x است .
مرحله ۲:برای هر پیکربندی C در A که هنوز در نظر گرفته نشده است ، همه پیکربندی هایی که M_1 از C با یک حرکت می تواند به آن برسد را نعیین کرده و به A اضافه می شود .
مرحله ۳: تکرار مرحله ۲ تا زمانی که پیکربندی های بیشتری به A اضافه شوند.
بوسیله لم.۵,۵,۱ ماشین تورینگ M_1 روی ورودی x با طول N حد اکثر $2^{dS(n)}$ پیکربندی متفاوت دارد, برای برخی ثابت که تنها به M_1 بستگی دارد. هر پیکربندی ($uqv, y_1qz_1, \ldots, y_mqz_m$) از M_1 روی ورودی x می تواند بوسیله M_2 در مکان ($\log n+m(S(n)+1)$ نمایش داده می شود. مجموعه A می تواند صریحا در $(\log n+m(S(n)+1))2^{dS(n)} \leq 2^{eS(n)}$ مکان که e ثابت است نمایش داده شود.تعداد دفعاتی که به A دسترسی انجام می شود دارای کران بالای تعداد عناصر درون آن است .در نتیجه ، نتیجه به دست می آید .
مثال ۵-۵-۱: M_1 ماشین تورینگ در شکل .۵-۵-۱(a)می باشد .



درخت پیکربندی های Ω از M_1 روی ورودی aa در شکل ۵-۵-۱ (a) داده شده است . برای ورودی داده شده x از M_1 $C_{i1\text{-}it}$ نماینده پیکربندی است که M_1 از پیکربندی اولیه اش با دنباله ای از حرکات که از قوانین انتقال τ_{1t} ، ... ، τ_{11} استفاده شده ،به آنها می رسد .اگر هیچ دنباله ای از حرکات ممکن نباشند ،آنگاه فرض می شود که $C_{i1\text{-}it}$ یک پیکربندی تعریف نشده است .

الگوریتم اثبات قضیه ۵،۵،۱ در مرحله اول C_ε را به A اضافه می کند .اولین تکرار از مرحله دوم پیکربندی های بلافاصله بعد از C_ε راکه C_1 و C_3 هستند تعیین و به A اضافه می کند .تکرار بعدی یکی از دو پیکربندی C_1 یا C_3 را در نظر می گیرد .

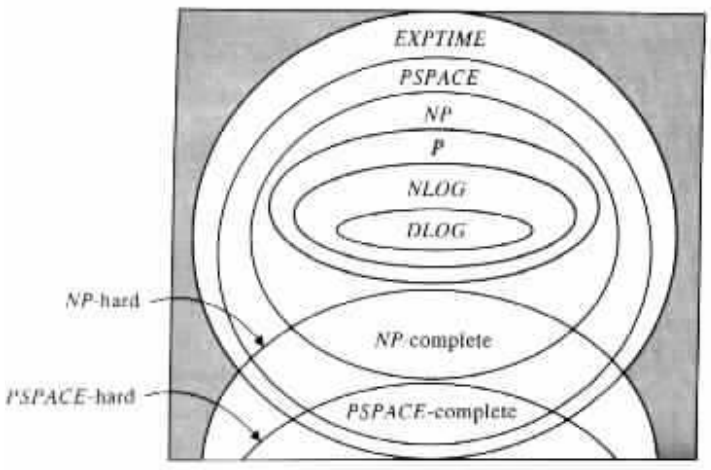
it

.

اگر C_1 قبل از C_3 بررسی شود آنگاه C_{11} و C_{13} پیکربندی هایی هستند که به وسیله C_1 به A اضافه می شوند . در چنین حالتی C_{33}, توسط C_3 به A اضافه می شود .
در زمان تکمیل شدن A شامل پیکربندی های C_ε
C_ε, C_1, C_3, $C_{1\,1}$, $C_{1\,3}$ (= $C_{3\,1}$), $C_{3\,3}$, $C_{1\,1\,5}$, $C_{1\,3\,5}$ (= $C_{3\,1\,5}$), $C_{3\,35}$, . . .
$C_{3\,1\,5\,6\,10\,12\,14\,13\,21}$,است.

از مکان غیر قطعی به مکان قطعی :

ازقضیه قبل به همراه نتیجه ۵,۳,۱ سلسله مراتب
$DLOG \subseteq NLOG \subseteq P \subseteq NP \subseteq PSPACE \subseteq EXPTIME$ به دست می آید .
(شکل ۵-۵-۲ را ببینید)



شکل ۵-۵-۲ : تصحیح از طبقه بندی در شکل ۵-۳-۱

به کمک قضیه بعدی ماشین های تورینگ غیر قطعی از پیچید گی مکانی چند جمله ای میتوانند با ماشینهای تورینگ قطعی با پیچید گی مشابه شبیه سازی شوند . هر چند از تمرین ۵-۲-٦, NLOG کاملا در PSPACE قرار دارد.
در نظر گرفته شده است .گذشته از این ،این شمول کامل و شمول کامل P در EXPTIME ،معلوم نیست که ایا هیچ از یک از شمول های دیگر در سلسه مراتب بالا کامل هست یا نه؟

قضیه بعدی راهکار مناسب تری را در مورد مکان ، نسبت به قضیه قبل فراهم می کند . هر چند بهبود در فضای مورد نیازباعث می شود تابه هزینه شبیه سازی آهسته تری برسیم .

قضیه ۵-۵-۲ : اگر $S(n)$ یک تابع کاملاً قابل مکانی باشد و$S(n)\geq\log n$ آنگاه

$$NSPACE(S(n)) \subseteq DSPACE(S(n)^2)$$

اثبات : هر ماشین تورینگ M_1 با محدودیت مکانی $cS(n)$ را در نظر می گیریم , که $S(n)\geq\log n$ کاملا قابل ساخت مکانی می باشد . بدون از بین رفتن کلیت ، می توان فرض کرد که در ورود به پیکربندی پذیرش نوارهای کمکی M_1 همگی خالی هستند و هد ورودی M_1 روی نماد پایانی راست $ است . علاوه بر این می توان فرض کرد که M_1 دقیقا یک حالت پذیرش q_f را دارد . بنابراین یک محاسبه پذیرش M_1 روی ورودی x داده شده، باید در پیکربندی پذیرش $(¢xq_f$, $q_f, \ldots, q_f)$ پایان پذیرد . بر طبق تعریف , M_1 ورودی داده شده x را میپذیرد اگر وتنها اگر M_1 روی ورودی x دنباله ای از حرکات داشته باشد که از پیکربندی اولیه C_0 از M_1 روی ورودی x شروع شده و در پیکربندی پذیرش C_f از M_1 روی ورودی x پایان می پذیرد. بر اساس لم ۵,۵,۱ ماشین تورینگ M_1 حد اکثر می تواند به $2^{dS(n)}$ پیکربندی مختلف روی ورودی به طول n دست یابد . بر اساس قضیه ۵-۵-۱ هر پیکربندی حداکثر $d's(n)$ فضا احتیاج دارد وقتی که رشته ورودی بیرون نگه داشته می شود. بنا بر این M_1، x رامی پذیرد اگر وتنهااگر حداکثر $2^{dS(|x|)}$ حرکت داشته باشد که از C_0 شروع شده و به C_f ختم می شود .

ماشین تورینگ قطعی M2 می تواند مشخص کند که آیا M_1 از طریق الگوریتم شکل ۵-۵-۳ ورودی x را می پذیرد یا خیر.

C_0 := the initial configuration of M_1 on input x
C_f := the accepting configuration of M_1 on input x
if $R(C_0, C_f, 2^{dS(|x|)})$ then accept
reject
function $R(C_1, C_2, t)$
if $t \leq 1$ then
    if M_1 can in t steps reach configuration C_2
        from configuration C_1 then return (true)

```
   else for each configuration C of M1 on input x
 d's(|x|) do≤            of length
) and t/2       if R(C1, C,
) then return (true)⌊t/2⌋       R(C, C2,
   return (false)
end
```

Figure 5.5.3 A deterministic simulation of a nondeterministic Turing machine M_1

شکل ۵-۵-۳ : شبیه سازی قطعی از ماشین تورینگ غیر قطعی M_1 .

این الگوریتم از تابع بازگشتی $R(C_1, C_2, t)$ استفاده می کند که مشخص کند که آیا M_1 روی ورودی x دارای دنباله ای از حداکثر حرکات که از C_1 شروع شده و به C_2 ختم می شود می باشد یا خیر . این وپژگی دقیقا زمانی که $t \leq 1$ می باشد چک می شود . در غیر اینصورت برای حالت C از M_1 با جستجوی کامل به صورت بازگشتی بررسی می شود به طوریکه هر دو

$R(C_1,C,[t/2])$, $R(C,C_2,[t/2])$ برقرار باشند .

الگوریتم از $O(S(n))$سطوح بازگشتی در $R(C_1,C_2,t)$ استفاده می کند . هر سطح بازگشتی به فضای $O(S(n))$ نیاز دارد . در نتیجه $O((S(n))^2)$, M_2 فضا را استفاده می کند .

وقتی پیکربندی های M_1 استنتاج می شوند M2 به این خصوصیت که $S(n)$ قابل ساخت مکانی است ، تکیه می کند .

مسائل PSPACE کامل

شیوه هایی نظیر انچه برای بیان NP سخت بودن در برخی مسائل داده شده استفاده شده اند می توانند برای نشان دادن PSPACE سخت بودن نیز استفاده شوند . قضیه زیر مثالی از یک مسئله PSPACE کامل است که در آن PSPACE سخت با یک تبدیل کلی نشان داده شده است .

قضیه ۵-۵-۳ : مسئله عضویت برای اتوماتای خطی محدود شده یا بطور مشابه برای $L =(x , M \}=)\mid$ یک اتوماتای خطی محدود شده است که x را می پذیرد $M\}$

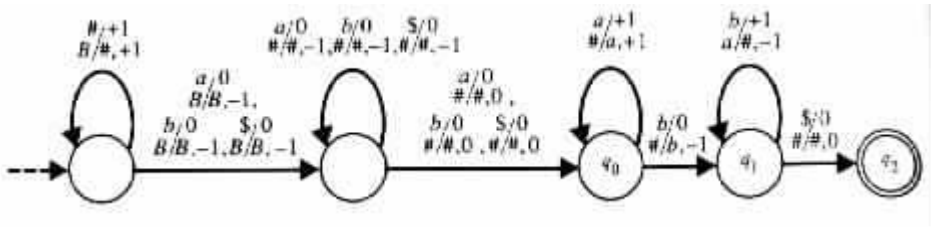
یک مسئله PSPACE کامل می باشد.
اثبات : زبان L با ماشین تورینگ غیر قطعی M_U مانند ماشین تورینگ عمومی در اثبات ٤-٤-١پذیرفته می شود.
M_U به صورت غیر قطعی روی ورودی (M,x) دنباله ای از حرکات M روی x را می یابد . M_U ورودی را می پذیرد اگر و تنها اگر دنباله حرکات از پیکربندی اولیه M روی ورودی x شروع شود و در یک پیکربندی پذیرش پایان پذیرد .محاسبه M_U به روش زیر پیش می رود .

M_U با ساختن پیکربندی اولیه C_0 از M روی ورودی x آغاز میشود .سپس مکررا و بطور غیر قطعی یک پیکربندی C را که M در یک مرحله از آخرین پیکربندی که برای M با M_U تعیین شده است ،بتواند به آن دست یابد.
MU , (M,x) را می پذیرد اگر و وقتی که به پیکربندی پذیرشی از M برسد .

با این ساختار ، M_U به فضای بیشتری از انچه حافظه برای ثبت کردن یک پیکربندی واحد از M نیازمند است احتیاج ندارد . یک پیکربندی واحد $(uqv,u_1qv_1,\ldots..,u_mqv_m)$ از M به فضایی برابر با میزانی که حافظه برای ثبت نماد ناز دارد ضرب در تعداد نمادها در این پیکربندی ، احتیاج دارد. یعنی
$$O(|M|((1+|uv|)+(1+|u_1v_1)+\ldots+(1+u_mv_m))) =،$$
$$O(|M|(m+1)(1+|x|)) = O(|M|^2|x|)$$

که |M| نماینده برای طول نمایش M است . در نتیجه M_U به فضایی که چند جمله ای نسبت به اندازه ورودی(M,x) نیاز دارد . این از قضیه ٥-٥-٢ دنبال می شود که زبان L در PSPACE قرار دارد .
برای اینکه نشان دهیم مسئله عضویت برای L , PSPACE سخت است هر مسئله K در PSPACE را در نظر بگیرید . فرض می کنیم A ماشین تورینگ قطعی از پیچیدگی فضای $S(n) = O(n^k)$ بوده که K را تصمیم گیری می کند .

از((A,S(n))،یک مبدل تورینگ قطعی T_k با محدودیت زمان چند جمله ای می تواند ساخته شود که جفت(M,y) را روی ورودی x به خروجی بفرستد .
فرض می شود که y رشته $x\#^j$ بوده که $j=S(|x|)$ و # یک نماد جدید هستند .
M فرض شده که است یک اتوماتای خطی محدود بوده که روی ورودی $x\#^j$ محاسبه ای از A را روی x با فضای j شبیه سازی می کند که یعنی y,M را می پذیرد اگر وتنها اگر x,A را درفضای j بپذیرد .
مثال ۵-۵-۲ : A را ماشین تورینگ شکل ۵-۳-۳ در نظر می گیریم . با استفاده از اصطلاحات علمی در اثبات قضییه ۵,۵,۳ اتوماتای خطی محدود M می تواند مشابه اتوماتایی که در شکل ۵-۵-٤ امده , باشد.



شکل ۵-۵-٤ : اتوماتای خطی محدود متناظر با ماشین تورینگ شکل ۵-۳-۳

M هر محاسبه را با کپی کردن نمادهای راهنمای # از ورودی به نوار کار کمکی اش شروع می کند . سپس M بصورت غیر قطعی محل هد نوار کار کمکی را روی یکی از نمادهای # تعیین می کند . سرانجام M محاسبه را شبیه A روی ورودی باقیمانده دنبال می کند . تفاوت اصلی این است که M منتظر نماد # است هر وقت که نماد پایانی چپ ¢ یا نمادجای خالیB را پیمایش کند.
.قضیه بعدی مثالی است از مسئله ای که ی PSPACE سخت بودن را با کاهش از مسئله PSPACE سخت دیگری نشان میدهد.
قضیه ۵-۵-٤ : مسئله غیر هم ارز بودن برای اتوماتای حالت متناهی , PSPACE کامل است .
اثبات : (M_1,M_2) را دو تایی از اتوماتای حالت متناهی در نظر می گیریم . ماشین تورینگ M می تواند هم ارز نبودن M_1 و M_2 را با یافتن بصورت غیر قطعی

ورودی $a_1 \ldots a_N$ که دقیقا با یکی از اتوماتای حالت متناهی M_1 و M_2 پذیرفته شده است , مشخص کند .

M محاسبه اش را با تعیین مجموعه S_0 از تمام حالاتی که M_1 و M_2 می توانند روی ورودی پوچ به انها برسند شروع می کند . بدون از بین رفتن کلیت ،می تواند فرض شود که M_1 و M_2 مجموعه های حالات مجزااز هم را دارند . سپس M هم زمان نمادهای درون $a_1 \ldots a_N$ را مشخص می کند. برای هر نماد a_i که M مشخص میکند , M همچنین مجموعه S_i (از حالاتی که در S_{i-1} می باشند) را که M_1و M_2 با مصرف a_i به ان می رسند را می یابد

M روی پیکربندی پذیرش توقف می کند به محض اینکه S_N را بیابد که در هر دو شرط زیر صدق کند:

S_N.a شامل حالت پذیرش M_1 بوده و شامل حالت پذیرش M_2 نباشد .

S_N.b شامل حالت پذیرش M_2 بوده و شامل حالت پذیرش M_1 نباشد .

در هر نمونه از محاسبه ، M نیاز دارد تنها آخرین نماد a_i و مجموعه های S_{i-1} و S_i ی بهم پیوسته شده را که M تعیین می کند , ثبت کند . یعنی M روی ورودی (M_1,M_2) فضای خطی در $|M_1|+|M_2|$ را استفاده می کند . در نتیجه M غیر قطعی از پیچیدگی فضای چند جمله ای می باشد . از قضیه ۵-۵-۲ اینطور نتیجه می شود که مسئله غیر هم ارزی برای اتوماتای حالت متناهی درPSPACE می باشد.

برای نشان دادن اینکه مسئله غیر هم ارزی برای اتوماتای حالت متناهی مسئله PSPACE سخت می باشد, کافی است وجود مبدل تورینگ قطعی T با زمان محدود چند جمله ای , که ویژگی زیر را دارا می باشد را ثابت کنیم :

T روی ورودی (M,x) از یک اتوماتای محدود خطی M و از یک ورودی x برای M , دو تایی (M_1,M_2) از اتوماتای حالت متناهی M_1 و M_2 رابه خروجی می دهد . به علاوه M_2, M_1, غیر هم ارز می باشند اگر وتنها اگر x,M را بپذیرد .

M_1 می تواند اتوماتای حالت متناهی باشد که ورودی داده شده را می پذیرد اگر وتنها اگر ورودی به صورت

$\#C_0\#C_1\# \ldots .\#C_f\#$ نباشد . C_0 پیکربندی اولیه M روی ورودی x در نظر گرفته می شود و C_f پیکربندی پذیرش M روی ورودی x در نظر گرفته می

شود . C_i پیکربندی فرض می شود که M می تواند با یک حرکت از C_{i-1},i=1.....f به آن برسد .طول هر C_i مساوی با (n+3)+m(S(n)+1) در نظر گرفته می شود , m تعداد نوارهای کار کمکی M و $S(n)$ پیچیدگی مکانی M و # نماد جدید فرض می شود .

M_1 می تواند با انتخاب غیر قطعی برای بررسی هر کدام از شروط زیر ، تعیین کند که ورودی از چنین شکلی نیست که:

a. اولین نماد در ورودی # نیست.

b. نماد آخر در ورودی # نیست.

c. C_0 یک پیکربندی اولیه از M روی ورودی x نیست.

d. C_f شامل یک پیکربندی پذیرش از M نیست.

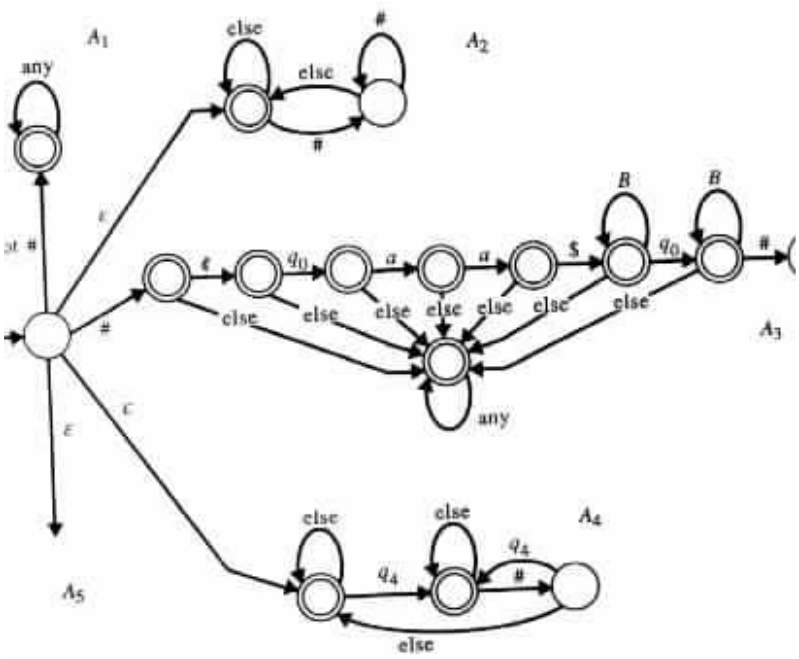
e. C_i با C_{i-1} برای بعضی iها (که بطور غیر قطعی انتخاب میشوند) سازگار نیست.

M_1 می تواند این شرط را با انتخاب غیر قطعی یک j به طوری که j امین سمبل در C_i با j امین سمبل از C_{i-1} ، و دو همسایه اش ،سازگار نباشد، بررسی کند .

M_1 می تواند با استفاده از تعداد چند جمله ای از حالت ها با طول x ، هر کدام از شرایط را بررسی کند .

با چنین ساختاری ،M_1 همه ورودی ها را می پذیرد اگر و تنها اگر x،M را نپذیرد. نتیجه بلافاصله به دست می آید اگر M_2 همه رشته های ورودی روی الفبای M_1 را بپذیرد .

مثال ۵-۳-۳ : ماشین تورینگ M در شکل ۵-۵-۱ ((aپیچیدگی مکانی $S(n)=n+2$ را دارد برای رشته $x=aa$ ماشین تورینگ M در اثبات از قضیه ۵-۵-٤ دارای اتوماتای حالت متناهی متناظر با M_1 است که در شکل ۵-۵-۵ آمده است.



شکل ۵-۵-۵ :یک اتوماتای حالت متناهی که همه ورودی را می پذیرد اگر وتنها اگر ماشین تورینگ در شکل ۵-۵-۱ (a) رشته aaرا نپذیرد.

هر پیکربندی (uqv,u_1qv_1) از M روی ورودی aa فرض می شود که در M_1 با رشته $uqvu_1qv_1$ از طول $(n+3)+(S(n)+1)=2n+6$ نمایش داده شود .

روی یک ورودی داده شده M_1 , بطور غیر قطعی انتخاب می کند که ی یکی از زیر مولفه های A_1,A_2,A_3,A_4,A_5 را اجرا کند .

A_1 بررسی می کند که ورودی با نماد # شروع نمی شود .A_2 بررسی می کند که ورودی با نماد# پایان نمی پذیرد .A_3 بررسی می کند که رشته بین یکی از دو نماد # که پیکربندی اولیه ای از M روی ورودی هستند نباشد.A_4 بررسی می کند که حالت پذیرش q4 در پیکربندی آخر C_f ظاهر نشود .

A_5 نا سازگاری بین پیکربندی های متوالی را بررسی می کند .

خواص بستاری

رده های $NTIME(T(n))$ و $NSPACE(S(n))$ تحت عمل اجتماع و اشتراک بسته هستند اما مشخص نیست که آیا آنها تحت عمل متمم گیری بسته هستند یا خیر .هر چند قضیه زیر برای $NSPACE(S(n))$ برقرار است .

قضیه ۵-۵-۵ : رده NSPACE(S(n)) برای S(n) logn تحت عمل متمم بسته است.

اثبات : ماشین تورینگ غیر قطعی M_1 بافضای محدود S(n) را را در نظر بگیرید . از روی M_1 هم ماشین تورینگ غیر قطعی M_2 با فضای محدود S(n) می تواند ساخته شوند تا متمم $L(M_1)$ را بپذیرد.

به ویژه روی یک ورودی x داده شده ، ماشین تورینگ M_2 تعیین می کند که آیا درخت پیکربندی های M_1 ،Ω،روی x شامل پیکربندی پذیرش هست یا نه؟اگر این طور بود ،آنگاه M_2، x را نمی پذیرد. در غیر این صورت M_2، x را می پذیرد .

M_2 ، Ω را طی مراحلی مطابق با الگوریتمی که در شکل ۵-۵-۶ (a) آمده می پیماید.

```
i:= 0
l := length of the initial configuration C0 of M1 on input x
N := 1
repeat
i do      for each configuration C in
        if C is an accepting configuration then reject
i+1)      l := max(l, length of longest configuration in
i+1     N := number of configurations in
    i := i + 1
until i > (the number of configurations, of M1 on input x, of
 length l)
accept
(a)
count := 0
for each configuration C, of M1 on
 input x, of length l at most
  do
* C in exactly i moves then     if C0
    begin count := count + 1

  end
 N then reject                  if count
(b)
nextN := 0
for each configuration C', of
```

```
M1 on input x, of length l do
  begin countup := false
i do      for each configuration C in
C' then countup := true⊢          if C
    if countup then nextN := nextN + 1
  end
N := nextN
(c)
```

شکل ۵-۵-۶ :

a. یک الگوریتم جستجوی اول سطح برای پذیرش متمم از L(M1)

b. شبیه سازی هر پیکربندی C_i در Ω_i c. ارزیابی تعداد پیکربندی های Ω_{i+1}

در i امین مرحله M_2 پیکربندی های C در i امین سطح از Ω را مشخص می کند یعنی پیکربندی هایی که در مجموعه { $M_1 \mid C$ } دنباله $C_0 \vdash^* C$ از دقیقاً i حرکت که از پیکربندی اولیه C_0 از M_1 روی ورودی x شروع می شود }

M_2 در مرحله i ام در یک پیکربندی غیر پذیرش متوقف می شود اگر پیکربندی پذیرشی در Ω_i تعیین کند . اگرچه ، M_2 در انتهای i امین مرحله در یک پیکربندی پذیرش متوقف می شود اگر معلوم گردد که Ω_i نمی تواند پیکربندی جدیدی را شامل شود. شامل حالت جدیدی باشد لیکن ان در اخر مرحله iام در یک شکل پذیرفته شده می ایستد (با تعیین اینکه i بزرگتر از تعداد پیکربندی های M_1 است که می تواند روی ورودی x به آنها برسد .)

پیکربندی های C که در Ω_i هستند، به صورت غیرقطعی از i و تعداد N از پیکربندی های در Ω_i پیدا شده اند . به ویژه M_2 دستوراتی به فرم " for each configuration C in Ω_i do α " را مطابق با الگوریتمی که در شکل ۵-۵-۶ امده , شبیه سازی می کند . غیر قطعیت برای شبیه سازی دنباله حرکات $C_0 \vdash^* C$ مورد نیاز است .

M2 تعداد پیکربندی های در Ω_{i+1} را با تعیین اینکه کدام پیکربندی می تواند مستقیماً از پیکربندی های Ω_i دستیابی شوند ، مشخص می کند .حال نتیجه به دست می آید زیرا با استفاده از لم ۵,۵,۱ ، ماشین تورینگ M_1 می تواند حداکثر به $2^{O(S(n))}$ i $2^{O(S(n))}$ پیکربندی های مختلف روی ورودی به طول n دست یابد ، یعنی ، M_2 فقط

$\leq$ سطح از Ω را در نظر می گیرد .

رده$DTIME(T(n))$ تحت اجتماع ، اشتراک و متمم بسته است .بسته بودن $DSPACE(S(n))$ تحت اجتماع ، اشتراک و متمم به راحتی با شبیه سازی مستقیم نشان داده می شود .

نظریه.۵,۵,٦ : هر ماشین تورینگ قطعی M_1 با محدودیت مکانی $s(n)$ دارای یک ماشین تورینگ قطعی M_2 با محدودیت زمانی $S(n)$ است که روی همه ورودی ها توقف می کند و با M_1 معادل می باشد .

اثبات: هرماشین تورینگ قطعی M_1 با محدودیت مکانی$s(n)$, که $M_1 = <Q, \Sigma,\Gamma,\delta,q_0,B,F>$ را در نظر می گیریم .M_2 می تواند به فرم زیر باشد.M_2 روی ورودی داده شده x مکان s_x را که M_1روی ورودی x استفاده می کند را تعیین می کند. سپس M_2 بررسی می کند که آیا M_1 دارای یک محاسبه پذیرش روی ورودی x است یا نه. اگر چنین بود سپس M_2 روی پیکربندی پذیرش توقف می کند . درغیر این صورت M_2 در پیکربندی غیر پذیرش توقف می کند.

برای معلوم کردن مقدار s_x ، ماشین تورینگ M_2 آن را با یک مقدار دهی اولیه می کند . سپس M_2 ، s_x را با یک واحد افزایش می دهدتا زمانی که یک پیکربندی با محدودیت مکانی s_x با نام C_1 و یک پیکربندی با محدودیت مکانی $(s_x + 1)$ با نام C_2 پیدا کندبه طوری که شرایط زیر برقرار باشند :

a. $C1 \vdash C2$

b. M_1 دارای یک درخت پیکربندی های با حرکت به عقب و محدودیت مکانیS_x ، Ω باشد . ریشه آن C_1 باشد و شامل پیکربندی اولیه ازM_1 روی x باشد.

M_2 برای پیکربندی با محدودیت مکانیS_x جستجو می کند که شرایط بالا را، بوسیله تولید همه پیکربندی های با محدودیت مکانی S_x در ترتیب استاندارد و بررسی هر کدام از آنها برای شرایط ، ارضا می کند.

بررسی اینکه آیا M_1 یکمحاسبه پذیرش روی ورودی x دارد یا نه ، ماشین تورینگ M_2 برای درخت پیکربندی های با حرکت به عقب و با محدودیت مکانی S_x که شرایط زیر را ارضا کنند ، جستجو می کند :

a. ریشه Ω پیکربندی پذیرش M_1 روی ورودی x باشد .

0- .b– Ω شامل پیکربندی اولیه M_1روی ورودی x باشد.

M_2 الگوریتم در شکل ۵-۵-۷ را پیروی می کند .

```
C := C_root
do
   if C is an initial configuration then
      begin
         while C has a predecessor in s do
            C := predecessor of C in s
         return (true)
      end
   if C is not a terminal node in s then
      C := the leftmost successor of C in s
   else if C has a right neighbor in s then
      C := the right neighbor C in s
   else if C has a predecessor in s then
      C := the predecessor of C in s
   else return (false)
until false
```

شکل ۵-۵-۷ جستجوی اول عمق برای پیکربندی اولیه در Ω

برای تعیین اینکه ایا M_1 دروردی x یک محدودیت مکانی s_x با حرکت به عقب درخت پیکربندی Ω را دارد یا نه ؟ ریشه که C_{root} است و شامل گره ای است متناظر باپیکربندی اولیه می باشد .به محض توقف ، الگوریتم در پیکربندی $C=C_{root}$ می باشد .

الگوریتم تنها روی پیکربندی C_{root} مورد استفادهقرار می گیرد .به طوری که اگر $C_{root} \vdash C'$ در پیکربندی های مکان محدود s نباشد . این خاصیت بوسیله الگوریتم برای تعیین ریشه از Ω عقبگرد مورد استفاده قرار میگیرد .

الگوریتم بر این مشاهده تکیه دارد که قطعیت M_1 خصوصیات زیر را برای هر درخت پیکربندی Ω_s با محدودیت مکانی و حرکت به عقب نتیجه می دهد :

a. درخت Ω_s متناهی هست زیرا هیچ پیکربندی نمی تواند دریک مسیری که ازریشه شروع می شود تکرار شود.

b. قبلی ها , بعدی ها همردیف ها هر یک از گره ها می تواند به سادگی از پیکربندی

منسوب شده به آن گره تعیین شود.

۵-۶ مسائل NP کامل

در برخی موارد مطالعه محدودیت های تعدادی زیرکلاس از p جالب است .انگیزه ممکن است نظری باشد آن چنان که در مورد زیر کلاسهاNLOG از P و یا عملی باشدآن چنان که درمورد زیر کلاسهاU_NC از مسائل در P که بوسیله برنامه های توازی کارا (ببینید بخش ۷,۵)می تواند حل شود.دراین موارد مفهوم "سخت ترین" مسائل درP با معنی دار است .

یک مسئلهK_1 کاهش پذیر با مکان لگاریتمی به مسئله K_2 گفته می شود اگر یک مبدل های تورینگ قطعی T_f و T_g با محدودیت مکانی لگاریتمی وجود داشته باشند, و که برای هر نمونه I_1 از K_1 شرایط زیر را ارضا کند :

a. T_f روی ورودی I_1 نمونهI_2 ازk_2 بدهد.

b . K_1 یک راه حلS_1 در I_1 دارد اگر وتنها اگرK_2 یک راه حل S_2 از I_2 داشته باشد ,کهS_1 خروجی T_g روی ورودی S_2 باشد .

مسئله k یک مسئله P سخت گفته می شود اگر هر مسئله درP کاهش پذیر با مکان لگاریتمی بهk باشد.مسئله ای p کامل گفته می شود - اگر آن یک مسئله Pسخت درP باشد.

بوسیله تعریف بالا, مسائل P کامل سخت ترین مسائل درP هستند و بوسیله بخش۷,۵

$$NLOG \subseteq U\_NC \subseteq P$$

در نتیجهNLOG شامل یک مسئله P کامل است اگر وتنها اگر P=U_NC باشد. و این یک مسئله باز است که آیا p با برابر NLOG یا U_NC است.

قضیه ۵-۶-۱ : مسائل تهی بودن برای گرامرهای مستقل از متن , P کامل است .

اثبات : هر گرامر مستقل از متن $G=<N, \Sigma, R, S>$ را در نظر می گیریم .تهی بودن $L(G)$ می تواند با الگوریتم زیر تعیین شود .

مرحله ۱ : هر نماد پایانی در Σ علامت گذاری شود

مرحله ۲ : مجموعه A برای قاعده تولید $A \to \alpha$ که α فقط شامل نمادهای علامت گذاری شده باشد و A بدون علامت ، جستجو می شود . اگر چنین قاعده تولید $\to \alpha$

A وجود داشت آنگاه A را علامت گذاری کرده و این روند را ادامه می دهیم .

مرحله ۳ : اگر نماد شروع S علامت گذاری نشده باشد آنگاه می گوییم $L(G)$ تهی است . در غیر اینصورت می گوییم $L(G)$ تهی نیست .

تعداد تکرارهای مرحله ۲ در بالا به وسیله تعداد نمادهای غیر پایانی در N محدود شده اند . بنابراین الگوریتم به زمان چند جمله ای نیازمند بوده و مساله در P می باشد.

برای آنکه نشان دهیم مسائل تهی بودن برای گرامرهای مستقل از متن p سخت هستند ، هر مساله kدر Pرا بررسی می کنیم . فرض می کنیم $M = <Q, \Sigma,\Gamma,\delta,q_0,B,F>$ یک ماشین تورینگ قطعی می باشد که K را در زمان $T(n) = O(n^k)$ تصمیم گیری کرده است . فرض می کنیم که m تعداد نوار کار کمکی M و r اندازه δ را مشخص می کند . آنگاه k می تواند توسط مبدل تورینگ T_k با محدودیت مکان لگاریتمی ،به مسئله تهی بودن برای گرامرهای مستقل از متن به روش زیر کاهش یابد .

T_k روی ورودی x , یک گرامر مستقل از متن G_x به شرح زیر را دارد به خروجی می دهد:

a.اگر K دارای جواب مثبت در x باشد آنگاه $L(G_x) =Ø$

b.اگر K دارای جواب منفی در x باشد آنگاه $L(G_x) =\{\varepsilon\}$ T_k گرامر G_x را به منظور توصیف محاسبه M روی ورودی x می سازد .

نماد های غیر پایانی G_x

نمادهای غیر پایانی G_x ویژگی های ممکن محاسبه M روی ورودی x را بیان می کند. G_x نمادهای غیر پایانی زیر را دارد .(فرض می شود که t مقدار , $T(|x|)$ را مشخص می کند)

a. نماد شروع S . S امکان داشتن محاسبه غیر پذیرش M روی ورودی x را نشان می دهد .

b. نماد غیر پایانی $A_{i,\tau}$, برای هر قاعده انتقال τ از M و هر $1 \le i \le t$ ، $A_{i,\tau}$ امکان اینکه i امین جابجایی M روی ورودی x از قاعده انتقال τ استفاده می کند را بیان می کند .

c. نمادغیر پایانی $A_{i,0,j,x}$ ، برای $0 \le i \le t$ و $1 \le j \le |x|+3$ و X در $\{\not\subset, \$\} \cup \Sigma \cup Q$. $A_{i,0,j,x}$ امکان داشتن X در iامین پیکربندی محاسبه در j امین مکان توصیف را بیان می کند .

d .نماد غیر پایانی $A_{i,r,j,x}$، برای هر $0 \le i \le t$ و $1 \le r \le m$ و $1 \le j \le 2t+1$ و X در $\Gamma \cup Q$ ، $A_{i,r,j,x}$ امکان داشتن xدر i امین پیکربندی محاسبه در j امین محل تو صیف r امین نوار کار کمکی را بیان می کند .

e . غیر پایانی $B_{i,r,q,x}$ برای هر $0 \le i \le t$ و $0 \le r \le m$ و q در Q و x در Σ .{$,¢}∪Γ

$B_{i,r,q,x}$ امکان اینکه در i امین پیکربندی , حالت q بوده و نماد زیر هد r امین نوار x می باشد را بیان می کند.

قوا عد تولید G_x

قواعد تولید G_x بیان می کنند که چگونه مشخصات محاسبه M روی ورودی x بیان می شوند .به ویژه , خصوصیتی که با نماد غیر پایانی A نمایش داده می شود برای محاسبه برقرار است اگر و تنها اگر$A \Rightarrow^* \varepsilon$ در G_x باشد.قواعد تولید G_x در ادامه آمده اند .

a. قواعد تواید که بخش ورودی در پیکربندی اولیه را بیان می کند

$$\begin{aligned} A_{0,0,1,q} &\to \epsilon \\ A_{0,0,2,q_0} &\to \epsilon \\ A_{0,0,i+2,i\text{th symbol in } x} &\to \epsilon \qquad \text{for } 1 \le i \le |x| \\ A_{0,0,|x|+3,t} &\to \epsilon \end{aligned}$$

b. قواعد تولیدی که بخش r امین نوار کمکی در پیکربندی اولیه و $1 \ge r \ge m$ را بیان می کند .

$$\begin{aligned} A_{0,r,j,B} &\to \epsilon \qquad \text{for } 1 \le j \le t \text{ and } t+2 \le j \le 2t+1 \\ A_{0,r,t+1,q_0} &\to \epsilon \end{aligned}$$

c. قواعد تولیدی که j امین نماد برای rامین نوار در i امین پیکربندی محاسبه را بیان می کند .

$$A_{i,r,j,x} \to A_{i-1,r,j-1,Y} A_{i-1,r,j,Z} A_{i-1,r,j+1,W} A_{i,r}$$

برای هرX,Y,Z,W,τ به طوری که $f_r(Y,Z,W,\tau) = X$. فرض می شود $f_r(Y,Z,W,\tau)$ تابعی است که جایگذاری X از Z برای r امین نوار زمانی که Y

سمت چپ Z است و w در سمت راست Z و τ قاعده انتقال در حال استفاده را بیان می کند . نمادهای مرزی چپ $A_{i-1,0,-1,Y}$, . . . , $A_{i-1,m,-1,Y}$, $A_{i-1,0,|x|+4,W}$ و نمادهای مرزی سمت راست

$A_{i-1,1,2t+2,W}$, . . . , $A_{i-1,m,2t+2,W}$ فرض می شوند که با رشته خالی ε برابر می باشند .

d. قواعد تولیدی که مشخص می کند که آیا محاسبه پذیرفته نشده است یا خیر؟

$$S \rightarrow B_{i,0,q,a} B_{i,1,q,b_1} \cdots B_{i,m,q,b_m}$$

برای هر $0 \leq i \leq t$ حالت پذیرفته نشده q و $a,b_1, b_2 \ldots\ldots b_m$ به طوری که $(q,a,b_1,\ldots\ldots b_m)=\Phi$

e. قواعد تولیدی که قواعد انتقال استفاده شده در I امین جابجایی محاسبه برای $1 \leq I \leq t$ را بیان می کند

$$A_{i,r} \rightarrow B_{i-1,0,q,a} B_{i-1,1,q,b_1} \cdots B_{i-1,m,q,b_m}$$

برای هر قاعده انتقال= ($q, a, b_1, \ldots, b_m, p, d_o, c_1, d_1, \ldots, c_m, d_m$) از ماشین تورینگ M .

M قطعی است , برای یک i داده شده وجوددارد حداکثر یک τ به طوری که $\Rightarrow^*_{\epsilon} A_{i,}$

f .قواعد تولیدی که برای مشخص کردن حالت M استفاده شده اند و نمادهایی که با هدهای M در i امین پیکربندی برای $0 \leq t \leq i$ بررسی شده اند.

.

$$\begin{aligned} B_{i,0,q,a} &\rightarrow A_{i,0,j_0,q} A_{i,0,j_0+1,a} \\ B_{i,1,q,b_1} &\rightarrow A_{i,1,j_1,q} A_{i,1,j_1+1,b_1} \\ &\vdots \\ B_{i,m,q,b_m} &\rightarrow A_{i,m,j_m,q} A_{i,m,j_m+1,b_m} \end{aligned}$$

برای هر $1 \leq j_0 \leq 1xl+2$ و $1 \leq j_r \leq 2t$ با $1 \leq r \leq m$.

مثال ۵--۱ فرض می کنیم M ماشین تورینگ قطعی شکل ۵-۳-۳ (a) بوده و x= ab و موارد قضیه ۵-۶-۱ را در نظر می گیریم .

قواعد ساخت G_x در زیر بخش ورودی در پیکربندی اولیه M روی x را بیان می کند

$$A_{0,0,1,q} \rightarrow \epsilon$$
$$A_{0,0,2,q_0} \rightarrow \epsilon$$
$$A_{0,0,3,a} \rightarrow \epsilon$$
$$A_{0,0,4,b} \rightarrow \epsilon$$
$$A_{0,0,5,\sharp} \rightarrow \epsilon$$

قواعد تولید که بخش نوار کار کمکی در پیکربندی اولیه را بیان می کند به صورت زیر است :

$$B_{0,0,j,B} \rightarrow \epsilon \quad 1 \leq j \leq 4 \text{and} 6 \leq j \leq 9$$
$$B_{0,0,5,q_0} \rightarrow \epsilon$$

قواعد تولید زیر دومین پیکربندی از اولی را بیان می کند

$$A_{1,0,1,\sharp} \rightarrow A_{0,0,1,q}\ A_{0,0,2,q_0}\ A_{1,\tau_1}$$
$$A_{1,0,2,a} \rightarrow A_{0,0,1,q}\ A_{0,0,2,q_0}\ A_{0,0,3,a}\ A_{1,\tau_1}$$
$$A_{1,0,3,q_0} \rightarrow A_{0,0,2,q_0}\ A_{0,0,3,a}\ A_{0,0,4,b}\ A_{1,\tau_1}$$
$$A_{1,0,4,b} \rightarrow A_{0,0,3,a}\ A_{0,0,4,b}\ A_{0,0,5,\sharp}\ A_{1,\tau_1}$$
$$A_{1,0,5,\sharp} \rightarrow A_{0,0,4,b}\ A_{0,0,5,\sharp}\ A_{1,\tau_1}$$
$$A_{1,1,1,B} \rightarrow A_{0,1,1,B}\ A_{0,1,2,B}\ A_{1,\tau_1}$$
$$A_{1,1,2,B} \rightarrow A_{0,1,1,B}\ A_{0,1,2,B}\ A_{0,1,3,B}\ A_{1,\tau_1}$$
$$A_{1,1,3,B} \rightarrow A_{0,1,2,B}\ A_{0,1,3,B}\ A_{0,1,4,B}\ A_{1,\tau_1}$$
$$A_{1,1,4,B} \rightarrow A_{0,1,3,B}\ A_{0,1,4,B}\ A_{0,1,5,q_0}\ A_{1,\tau_1}$$
$$A_{1,1,5,a} \rightarrow A_{0,1,4,B}\ A_{0,1,5,q_0}\ A_{0,1,6,B}\ A_{1,\tau_1}$$
$$A_{1,1,6,q_0} \rightarrow A_{0,1,5,q_0}\ A_{0,1,6,B}\ A_{0,1,7,B}\ A_{1,\tau_1}$$
$$A_{1,1,7,B} \rightarrow A_{0,1,6,B}\ A_{0,1,7,B}\ A_{0,1,8,B}\ A_{1,\tau_1}$$
$$A_{1,1,8,B} \rightarrow A_{0,1,7,B}\ A_{0,1,8,B}\ A_{0,1,9,B}\ A_{1,\tau_1}$$
$$A_{1,1,9,B} \rightarrow A_{0,1,8,B}\ A_{0,1,9,B}\ A_{1,\tau_1}$$

قاعده تولید قاعده انتقال $A_{1,\tau_1} \rightarrow B_{0,0,q_0,a}\ B_{0,1,q_0,B}$ در اولین حرکت را بیان می کند

قواعد تولید زیر حالت M و نمادهایی که به وسیله هدهای M در اولین پیکربندی

بررسی شده اند را بیان می کند

$$B_{0,0,q_0,a} \rightarrow A_{0,0,2,q_0} A_{0,0,3,a}$$
$$B_{0,1,q_0,B} \rightarrow A_{0,1,5,q_0} A_{0,1,6,B} .$$

رر

فصل ششم

محاسبات احتمالی

اهداف

در پایان فصل، دانشجو با مفاهیم زیر آشنا می‌شود:

برنامه های احتمالاتی بدون خطا

برنامه های احتمالاتی با امکان خطا

مبدل های تورینگ احتمالاتی

زمان چند جمله ای احتمالاتی

مقدمه

در اکثر موارد برنامه هایی راضی کننده هستند که روی هر ورودی رفتار خوبی انجام می دهند .چنین برنامه هایی حتی زمانی که احتمال کمی برای خطا و تولید پاسخ های غلط دارند نیز راضی کننده اند .برنامه هایی با این ساختارمیتوانند در دستوراتشان انتخاب های تصادفی را ایجاد کنند.

به این نوع از برنامه ها احتمالی می گویند.

بخش اول این فصل دستورات احتمالاتی را در برنامه ها معرفی می کنند و بخش دوم فایده چنین برنامه هایی که ممکن است خطا کنند را بررسی می کند.

بخش سوم نظریه مبدل تورینگ احتمالی برای مدل سازی محاسبات برنامه های

احتمالاتی را معرفی میکند. این فصل با بررسی برخی از رده های زمان چند جمله ای احتمالی مسائل خاتمه پیدا می کند.

٦-١ برنامه های احتمالاتی مستقل از خطا

تصادفی سازی (Randomization) یکی از ابزارهای مهم برنامه نویسی است. به طور شهودی قدرت آن برپایه انتخاب استوار است. توانایی انجام " انتخاب های تصادفی " می تواند به عنوان ریشه توانایی انجام " انتخاب های غیرقطعی " در نظر گرفته شود.

در موارد غیر قطعی هر اجرا از یک دستور باید از بین تعدادی گزینه انتخاب کند. بعضی از گزینه ها ممکن است خوب و برخی بد باشند. در صورتی که گزینه خوب موجود باشد باید آنرا انتخاب کنیم. مسئله اینست که عموما به نظر می رسد امکان انجام انتخاب غیرقطعی به طریقه کارآمد وجود نداشته باشد.

گزینه ها در مورد انتخاب های تصادفی شبیه موارد غیر قطعی است. در هر صورت هیچ محدودیتی در ساختار گزینه ی انتخاب شده نیست. در عوض هرکدام از گزینه های خوب و بد بااحتمالهای مساوی برای انتخاب شدن فرض می شود. در نتیجه نبود اریبی و تمایل در میان گزینه های مختلف امکان اجراهای کارآمد از انتخاب ها را افزایش می دهد. مسئولیت افزایش احتمال بدست آوردن انتخاب های خوب بر عهده برنامه نویس است.

اینجا متغیرهای تصادفی با دستورات تعیین تصادفی به برنامه ها معرفی می شوند. به صورت

$x := random(S)$, که S می تواند هرنوع مجموعه متناهی باشد.

یک اجرای دستور انتساب تصادفی $x := random(S)$ عنصری از S را به x نسبت می دهد .که هر عنصر S با احتمال انتخاب شدن مساوی فرض شده. برنامه ها ی با دستورات انتساب تصادفی و بدون دستورات غیر قطعی برنامه های احتمالی نامیده می شوند.

هر دنباله اجرااز یک برنامه ی احتمالی یک محاسبه فرض می شود . برنامه احتمالی

روی ورودی داده شده ممکن است هم محاسبات پذیرش و هم غیر پذیرش داشته باشد.

فرض می شود که اجرای یک دستور انتساب تصادفی به صورت x := random(S), ازیک واحد زمانی تحت معیار ارزش یکنواخت استفاده میکند و زمان |v| + log |S| تحت معیار ارزش لگاریتمی است. |v| طول نمایش مقدارv انتخاب شده از S و |S| نشانگر کاردینالیتی S فرض میشود.

برنامه ی احتمالاتی P دارای پیچیدگی زمانی مورد انتظار $\bar{t}(x)$ روی ورودی x است اگر $\bar{t}(x)$ برابر باشد با

$p_0(x) . 0 + p_1(x) . 1 + p_2(x) . 2 + \ldots$ تابع $P_i(x)$ تابع احتمالی فرض می شود که برای برنامه P با ورودی

x یک محاسبه انجام می دهد که دقیقا i واحد زمانی را می گیرد.

برنامه ی P دارای پیچیدگی زمانی مورد انتظار $\bar{T}(n)$ است اگر $\bar{T}(|x|) \geq \bar{t}(x)$ برای هر x باشد .

مثال زیر نشان میدهد چگونه احتمالات می توانند برای تضمین بهتر شدن رفتار روی هر ورودی (روی معدل) استفاده می شوند.

مثال ۶-۱-۱. برنامه قطعی را در شکل ۶-۱-۱ در نظر یگیرید.

```
call SELECT(k, S)
procedure SELECT(k,S)
    x   := first element in S
    S1 := { y | y is in S, and y < x }
   n1 := cardinality of the set stored in S1
    S2 := { y | y is in S, and y > x }
   n2 := cardinality of the set stored in S2
    n3 := (cardinality of the set stored in
S) - n2
    case
 n1: SELECT(k, S1)≤                    k
           n3 < k   : SELECT(k - n3, S2)
 n3: x holds the desired ≤       n1 < k
element
```

```
    end
end
```

شکل ٦-١-١ برنامه ای که k امین المان کوچک را در S انتخاب مینماید .

(بدون استفاده از بازگشت) برنامه K امین کوچکترین عنصر را در هر مجموعه مفروض S از کاردینالیتی متناهی انتخاب می کند .

T(n) زمانی را (تحت معیار یکسان) که برنامه برای انتخاب یک عنصر از مجموعه با کاردینالیتی n می گیرد، نشان می دهد. T(n) برای چند ثابت c و چند عدد صحیح m<n طبق نامساوی زیر برقرار است.

$$T(n) \leq \begin{cases} T(m) + cn & \text{if } n > 1 \\ c & \text{if } n \leq 1 \end{cases}$$

طبق نامساوی بالا

$$\begin{aligned} T(n) &\leq T(n-1) + cn \\ &\leq T(n-2) + c(n + (n-1)) \\ &\leq T(n-3) + c(n + (n-1) + (n-2) \\ &\leq K \\ &\leq T(1) + c(n + (n-1) + \Lambda + 2) \\ &\leq cn^2 . \end{aligned}$$

پیچیدگی زمانی برنامه $O(n^2)$ میباشد.

زمان مورد نیاز برنامه به ترتیب عناصر در مجموعه ها در سوال حساس است. برای مثال زمانیکه کوچکترین عنصر جستجو می شود، $O(n)$زمان کافی است اگر عناصر مجموعه در ترتیب افزایشی باشند. متناوباً برنامه زمان $O(n^2)$ را نیاز دارد اگر عناصر در ترتیب کاهشی قرار گرفته شده باشند.

این میزان حساسیت به ترتیب عناصر میتواند با انتساب عنصر تصادفی از S به X به جای اولین عنصر S حذف شود. در برخی موارد پیچیدگی زمانی مورد انتظار $\overline{T}(n)$ برنامه نامساوی زیر را برای بعضی ثوابت c ارضا می کند.

$$\overline{T}(n) \leq \begin{cases} \frac{1}{n}\left(\overline{T}(0)+\overline{T}(1)+\cdots+\overline{T}(n-1)\right)+cn & \text{if } n > 1 \\ c & \text{if } n \leq 1 \end{cases}$$

طبق نابرابری بالا

$$
\begin{aligned}
\overline{T}(n) &\le \frac{1}{n}(\overline{T}(0)+\Lambda+\overline{T}(n-1))+cn \\
&\le \frac{1}{n}(\overline{T}(0)+\Lambda\ +\overline{T}(n-2)) \\
&\quad +\frac{1}{n}(\frac{1}{n-1}(\overline{T}(0)+\Lambda\ +\overline{T}(n-2)))+\frac{1}{n}(c(n-1))+cn \\
&\le \frac{1}{n}(1+\frac{1}{n-1})(\overline{T}(0)+\Lambda\ +\overline{T}(n-2))+c(1-\frac{1}{n})+cn \\
&\le \frac{1}{n-1}(\overline{T}(0)+\Lambda\ +\overline{T}(n-2))+c+cn \\
&\le \frac{1}{n-1}(\overline{T}(0)+\Lambda\ +\overline{T}(n-3))+\frac{1}{n-1}(\frac{1}{n-2}(\overline{T}(0)+\Lambda\ +\overline{T}(n-3))) \\
&\quad +\frac{1}{n-1}(c(n-2))+c+cn \\
&\le \frac{1}{n-1}(1+\frac{1}{n-2})(\overline{T}(0)+\Lambda\ +\overline{T}(n-3))+2c+cn \\
&\le \frac{1}{n-2}(\overline{T}(0)+\Lambda\ +\overline{T}(n-3))+2c+cn \\
&\le \frac{1}{n-3}(\overline{T}(0)+\Lambda\ +\overline{T}(n-4))+3c+cn \\
&\le \Lambda \\
&\le \overline{T}(1)+(n-1)c+cn \\
&\le 2cn\ .
\end{aligned}
$$

برنامه اصلاح شده ،احتمالی است و پیچیدگی زمانی مورد انتظار آن $O(n)$ است. برای هر ورودی (K,S) داده شده با S از کاردینالیتی $|S|$ برنامه ی احتمالی ،یافتن K امین کوچکترین عنصر از S را با زمان $O(|S|^2)$ ضمانت می کند. به هر حال به طور متوسط برای هر ورودی زمان $O(|S|)$ نیاز است.

٦-۲ برنامه های احتمالاتی که ممکن است خطا کنند

به دلایل تجربی زیادی برنامه ها اجازه خطا کردن با احتمال خیلی کم روی بعضی از خروجی ها را دارند.

مثال ٦-۲-۱. یک الگوریتم غیرهوشمند برای حل کردن مسئله غیراول بودن زمان نمایی صرف میکند(مثال ۳-۱-۵ را ببینید) برنامه شکل ٦-۲-۱ مثالی است از یک برنامه

احتمالی که غیراول بودن اعداد را در زمان چند جمله ای تعیین می کند.

```
read x
y := random({2, . . . , √x })
if x is divisible by y then
   answer := yes      /* not a prime number?*/
else answer := no
?
```

شکل ٦-۲-۱. یک برنامه احتمالی نامطلوب برای مسئله اول نبودن

برنامه ،احتمال خطای صفر روی ورودی هایی که اعداد اول هستند دارد. به هر حال برای تعداد بی نهایت اعداد غیراول احتمال جواب نادرست دادن برنامه بالاست. بویژه احتمال خطا روی عدد غیراول m به میزان $1- s/(\sqrt{m}-1)$ است که s تعداد مقسوم علیه های مجزای m در $\{2$ و ... و $\sqrt{m}\}$ فرض می شود. همچنین احتمال خطا میتواند به مقدار $1- 1/(\sqrt{m}-1)$ برای m هایی که توان دوم یک عدد اول باشد برسد.

احتمال دادن پاسخ غلط برای عدد ورودی m با k بار اجرای برنامه می تواند کاهش پیدا کند . در چنین مواردی اگر با هر k اجرا پاسخ مثبت به دست آیدm به طور حتم غیراول اعلام می شود . در غیر این صورت , با احتمال خطای حداکثر $(1- 1/(\sqrt{m}-1))^k$ معلوم می شود که m اول است. با افزایش m و با توجه به اینکه $k = c(\sqrt{m}-1)$ (جایی که c ثابت ۷۱۸۲۸. ۲ باشد این احتمال به مقدار $(1/\lambda)^c < 0.37^{\,c}$ نزدیک می شود. به هر حال این مقدار برای k به طور خیلی زیادی بزرگ است زیرا آن نسبت به طول نمایش m یعنی $\log m$ ، نمایی است .

با استفاده از نتیجه زیر برنامه احتمالی بهبود می یابد .

نتیجه: $W_m(b)$ گزاره ای است که درست است اگر و تنها اگر هر دو شرط با هم برقرار باشد.

a. $(b^{m-1} - 1) \bmod m \neq 0$

b. $1 < \gcd(b^t - 1, m) < m$ برای برخی i و t به طوری که $m-1 = t.2^i$.

آنگاه برای هر عدد صحیح $m \geq 2$, دو شرط زیر برقرار است .

الف) m عدد اول است اگر و فقط اگر $W_m(b)$ برای همه ی b هایی که $2 \leq b < m$, نادرست باشد.

ب) اگر m اول نباشد آنگاه مجموعه $\{ b \mid 2 \leq b < m$ و $W_m(b)$ برقرار است $\}$ حداقل از کاردینالیتی $(m - 1)(۳/٤)$ است .

این نتیجه به برنامه احتمالی در شکل ٦-٢-٢ اشاره می کند .

```
read x
y := random{2, . . . , x - 1}
if W_x(y) then answer := yes
else answer := no
```

شکل ٦-٢-١. یک برنامه احتمالی خوب برای مسئله غیراول بودن.

برای اعداد اول m برنامه همیشه پاسخ درست می دهد . از طرفی برای اعداد غیر اول m برنامه با احتمال حداکثر $(1- (3/4)(m - 1)/(m - 2) \leq ۱/٤$ برای یک خطا , پاسخ درست می دهد.

احتمال پاسخ اشتباه می تواند به ثابت ε مورد نظر با $k \geq \log_{1/4} \varepsilon$ بار اجرای برنامه کاهش یابد . آن عدد k که برنامه به تعدادش اجرا می شود به ورودی m بستگی دارد.

بررسی شرط $(b^{m-1} - 1) \bmod m \neq ۰$ در زمان چند جمله ای با استفاده از رابطه

$$(a + b) \bmod m = ((a \bmod m) + (b \bmod m)) \bmod m$$

و رابطه $(ab) \bmod m = ((a \bmod m)(b \bmod m)) \bmod m$ می تواند انجام شود .

بررسی شرط $\gcd(b^t - 1, m)$ در زمان چند جمله ای با استفاده از الگوریتم اقلیدس (Euclid) می تواند انجام شود . در نتیجه برنامه شکل ٦-٢-٢ پیچیدگی زمانی چند جمله ای دارد .

مثال ٦-٢-٢ درستی رابطه $AB \neq C$ را برای هر ماتریس A وB وC بررسی کنید .

یک الگوریتم غیرهوشمندانه برای تصمیم گیری مسئله می تواند محاسبه $D = AB$ و $E = D - C$ را انجام دهد بررسی کند که آیا

$$E \neq \begin{pmatrix} 0 & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & 0 \end{pmatrix}$$

آیا یک ضرب غیرهوشمندانه برای A و B زمان $O(N^3)$ نیاز دارد (تحت معیار ارزش یکنواخت) اگر A و B از بعد $N \times N$ باشند الگوریتم غیرهوشمندانه برای تصمیم گیری اینکه آیا $AB \neq C$ نیز به زمان $O(N^3)$ نیاز دارد.

به هر حال نامساوی $AB \neq C$ برقرار است اگر و تنها اگر نامساوی

$$\begin{pmatrix} \hat{x}_1 \\ \vdots \\ \hat{x}_N \end{pmatrix}_{(AB - C)} \neq \begin{pmatrix} 0 \\ \vdots \\ 0 \end{pmatrix}$$

برای برخی بردارها برقرار باشد .

$$\hat{x} = \begin{pmatrix} \hat{x}_1 \\ \vdots \\ \hat{x}_N \end{pmatrix}$$

در نتیجه، نامساوی $AB \neq C$ می تواند با برنامه احتمالی که موارد زیر را تعیین می کند ،مشخص گردد:

الف) یک بردار ستونی

$\hat{x} = \begin{pmatrix} \hat{x}_1 \\ \vdots \\ \hat{x}_N \end{pmatrix}$ برای اعداد تصادفی از {-1,1} .

ب)

ج) مقدار بردار $\hat{y} = B\hat{x}$

د) مقدار بردار $\hat{z} = A\hat{y}$

ه) مقدار بردار $\hat{x} = C\hat{x}$

و) مقدار بردار

$$\begin{aligned} \hat{v} &= \hat{z} - \\ &= A\hat{y} - C\hat{x} \\ &= AB\hat{x} - C\hat{x} \\ &= (AB - C)\hat{x} \end{aligned}$$

$$= \begin{pmatrix} d_{11} & \cdots & d_{1N} \\ \vdots & \ddots & \vdots \\ d_{N1} & \cdots & d_{NN} \end{pmatrix} \begin{pmatrix} \hat{x}_1 \\ \vdots \\ \hat{x}_N \end{pmatrix}$$

$$= \begin{pmatrix} d_{11}\hat{x}_1 + \cdots + d_{1N}\hat{x}_N \\ \vdots \\ d_{N1}\hat{x}_1 + \cdots + d_{NN}\hat{x}_N \end{pmatrix}$$

اگر بعضی از درایه ها در بردار

$$\hat{v} = \begin{pmatrix} d_{11}\hat{x}_1 + \cdots + d_{1N}\hat{x}_N \\ \vdots \\ d_{N1}\hat{x}_1 + \cdots + d_{NN}\hat{x}_N \end{pmatrix}$$

مخالف صفر هستند, پس برنامه احتمالی گزارش می دهد که نامساوی $AB \neq C$ باید برقرار باشد. وگرنه برنامه با احتمال خطای حداکثر ½ نشان می دهد که $AB = C$.

برنامه زمان $O(N^2)$ را مصرف می کند .

تحلیل برنامه برای احتمال خطایش در نتیجه زیر مشخص می شود .

نتیجه: $d_1x_1 + \ldots + d_Nx_N = 0$ یک معادله خطی است با ضرایبی که نامساوی $(d_1, \ldots, d_N) \neq (0, \ldots, 0)$ را ارضا می کند. پس این معادله خطی حداکثر 2^{N-1} ریشه $\{\hat{x}_1 \ldots \hat{x}_N\}$ روی بازه {۱و ۱-} دارد .

اثبات : هر معادله خطی $d_1x_1 + \ldots + d_Nx_N = 0$ را با در نظر گرفتن شرط $d_1 >$

0 در نظر بگیرید .

اگر { x_N ....x_2, . . . , x_N} ریشه های معادله خطی روی {۱و ۱-} باشد آنگاه (x_1 , - x_1, . . . , x_N) و (x_1, x_N) نمی تواند معادله را حل کند . به عبارت دیگر اگر (x_1, . . . , x_N) معادله را روی {۱و ۱-} حل کنند, آنگاه نامساوی (x_1, . . . , x_N) ≠(-x_1, . . . , x_N) به دست می آید جایی که (x_1, . . . , x_N) ≠(x_1, . . x_N) .

در نتیجه , هر ریشه در معادله یک انتساب مربوط دارد که معادله را حل نمی کند . حداکثر نیمی از انتساب های (x_1, . . . , x_N) های تعیین شده روی {۱و ۱-} می تواند به عنوان ریشه معادله به کار رود .

برنامه احتمالی می تواند روی هر سه گانه A و B و C داده شده k بار اجرا شود. در این مورد اگر هر اجرا بردار غیرصفر 0 را نتیجه دهد $AB \neq C$ باید به دست بیاید . در غیر این صورت $AB = C$ با احتمال خطای حداکثر $(1/2)^k$ به دست می آید . با تکرار اجرای برنامه احتمالی می توان احتمال خطا را به هر مقدار دلخواه کاهش داد . بنابراین , برای همه ی انتخاب های ممکن از A و B و C احتمال خطای $(1/2)^k$ نتیجه می شود .

احتمال خطا در تکرار اجرای برنامه احتمالی

برنامه های احتمالی در دو مثال قبل خاصیت احتمال خطای یکطرفه را نشان می دهد. بویژه پاسخ مثبت آنها همیشه درست است . از طرفی, پاسخ منفی آنها ممکن است درست یا نادرست باشد . عموما و به هر حال برنامه های احتمالی ممکن است روی بیش از یک نوع از جواب خطا کنند . در این مورد پاسخ با در نظر گرفتن اکثریت جوابهای بدست آمده از محاسبات تکراری از نمونه های گرفته شده می تواند به دست آید .

لم زیر احتمال خطا در محاسبات تکراری را به منظور ثابت کردن یک جواب با اکثریت مطلق را تحلیل می کند .

لم ٦-٢-١ هر برنامه احتمالی P را در نظر بگیرید.فرض می شود که P احتمال e را برای تهیه کردن یک خروجی متفاوت از y با ورودی x دارد. پس pاحتمال

$$\Phi(N,e)=\sum_{k=0}^{N}\binom{2N+1}{k}e^{2N+1-k}(1-e)^{k}$$

را دارد.

با داشتن نهایتا N+1 محاسبه با خروجی هایی که از y متفاوت هستند با هر دنباله از 2N+1 محاسبه از P با ورودی X احتمال بالا را دارد.

اثبات: P و X و Y و e را در صورت لم در نظر بگیرید . تابع φ(N, e, k) احتمالی را مشخص می کند که در یک ترتیب از 2N+1 محاسبه روی ورودی x و p دقیقا k بار محاسبه دارد با یک خروجی که با y مساوی است . تابع Φ(N, e) احتمال اینکه در یک ترتیب از 2N+1 محاسبه روی ورودی P، x نهایتا N+1 محاسبه با خروجی که متفاوت با Y است خواهد داشت رامشخص می کند. .

بر طبق تعریف

$$\begin{aligned}\Phi(N,e) &= \text{(Probability that in exactly } 2N+1 \text{ computations the output will not equal } y)\\ &\quad + \text{(Probability that in exactly } 2N \text{ computations the output will not equal } y)\\ &\quad \vdots\\ &\quad + \text{(Probability that in exactly } N+1 \text{ computations the output will not equal } y)\\ &= \text{(Probability that in exactly 0 computations the output will equal } y)\\ &\quad + \text{(Probability that in exactly 1 computation the output will equal } y)\\ &\quad \vdots\\ &\quad + \text{(Probability that in exactly } N \text{ computations the output will equal } y)\\ &= \sum_{k=0}^{N}\varphi(N,e,k).\end{aligned}$$

احتمال داشتن پاسخ Y در و فقط در i_1 ام تا...i_k ام محاسبه در یک دنباله از ۲N+1

محاسبه از P روی ورودی n مساوی است با

$$e^{i_1-1}(1-e)e^{i_2-i_1-1}(1-e)e^{i_3-i_2-1}(1-e)\cdots e^{i_k-i_{k-1}-1}(1-e)e^{2N+1-i_k} = (1-e)^k e^{2N+1-k}$$

$i_1,\ldots,i_k$ برای ارضا کردن $1 \le i_1 < i_2 < \ldots < i_k \le N+1$ ۲ فرض شده است.

هر مجموعه $\{C_1,\ldots,C_{2N+1}\}$ از ۲N+1 محاسبه P دارد

$$(2N+1)(2N)(2N-1)\cdots(2N+1-k+1) = \frac{(2N+1)!}{(2N+1-k)!}$$

ترتیب ممکن C_{i1}-C_{ik} از K محاسبه .در این ترتیب فقط

$$\frac{(2N+1)!}{k!(2N+1-k)!} = \binom{2N+1}{k}$$

شرط $i_1 < \ldots < i_k$ را ارضا می کند . در نتیجه در ترتیب از ۲N+1 محاسبه ی P $\binom{2N+1}{k}$ راه ممکن برای بدست آوردن خروجی Y با دقیقا K بار اجرا وجود دارد.

نتیجه: که (تعداد ترتیب ممکن از ۲N+1 محاسبه, دقیقا K بار خروجی Y را دارد)= $\Phi(N, e, k)$, به تعداد (احتمال داشتن یک ترتیب از N+1۲ محاسبه با دقیقا K خروجی از Y). دفعه بدست می آید.

عموما ما تنها به برنامه های احتمالی علاقه مندیم که در زمان چندجمله ای اجـرا شـوند و احتمال خطایی داشته باشند که بتواند با اجرای برنامه به تعداد اعداد چندجمله ای بـه سمت یک ثابت دلخواه کاهش یابد .

قضیه زیر کاربرد برنامه احتمالی را که ممکن است خطا کند بررسی می کند .

قضیه ٦-۲-۱: $\Phi(N, e)$ را به عنوان صورتی که در لم ٦-۲-۱ آمده است , قرار می دهیم . در نتیجه $\Phi(N, e)$ خاصیت های زیر را دارد .

الف) $\Phi(N,\frac{1}{2}) = \frac{1}{2}$ برای همه N ها

ب) $\Phi(N, e)$ برای هر ثابت e > ½ زمانی که N به ∞ میل می کند , به سمت صفر میل می کند.

ج) $\Phi(N, \frac{1}{2}-\frac{1}{N})$ زمانی که N به سمت ∞ میل می کند به سمت ثابتی که بزرگتر از $1/(2\lambda^6)$ است میل می کند.

اثبات :

الف: مساوی $\Phi(N,\frac{1}{2})=\frac{1}{2}$ برای همه مقادیر N از روابط زیر گرفته شده است :

$$2\Phi(N,\frac{1}{2}) = \sum_{k=0}^{N}\binom{2N+1}{k}\left(\frac{1}{2}\right)^{2N+1-k}\left(1-\frac{1}{2}\right)^{k}_{2}$$

$$= \sum_{k=0}^{N}\binom{2N+1}{k}\left(\frac{1}{2}\right)^{2N+1-k}\left(1-\frac{1}{2}\right)^{k}$$

$$\sum_{k=N+1}^{2N+1}\binom{2N+1}{k}\left(\frac{1}{2}\right)^{2N+1-k}\left(1-\frac{1}{2}\right)^{k} +$$

$$= {}^{2N+1}\left(\frac{1}{2}+\left(1-\frac{1}{2}\right)\right)$$

$$= 1$$

ب: مساوی

$$\sum_{k=0}^{2N+1}\binom{2N+1}{k} = 2^{2N+1}$$

برای مقادیر k به صورت $\binom{2N+1}{k} \le 2^{2N+1}$ نامساوی زیر را بیان می کند

$$\binom{2N+1}{k} \le 2^{2N+1}$$

در نتیجه با در نظر گرفتن $e=\frac{1}{2}-\delta$ نتیجه از روابط زیر به دست می آید .

$$\Phi(N, e)= \sum_{k=0}^{N}\binom{2N+1}{k}e^{2N+1-k}(1-e)^{k}$$

$$\leq e^{N+1}(1-e)^N \binom{2N+1}{N}(N+1)$$

$$\leq (N+1)2^{2N+1}e^{N+1}(1-e)^N$$

$$= {}^N\left(\tfrac{1}{2}+\delta\right)^{N+1}\left(\tfrac{1}{2}-\delta\right)(N+1)2^{2N+1}$$

$$= {}^N\delta)^N(1+2\delta(N+1)(1-2\left(\tfrac{1}{2}-\delta\right)2$$

$$= {}^2)^N\delta 2e(N+1)(1-4$$

ج: نتیجه ای که از روابط زیر به دست می آید به دلیل میل کردن $(1-2/N)^{N/2}$ با N به سمت ∞ 1است.

د:.بر طبق قضیه ی ٦-٢-١ الف , به منظور کاهش احتمال بدست آوردن پاسخ نادرست با تکرار اجرای برنامه , یک برنامه احتمالی باید یک خطای احتمالی $e(x)$ کوچکتر از ½ داشته باشد . بر طبق قضیه ی ٦-٢-١ ب , احتمال خطای $e(x)$ که از برخی ثوابت $\varepsilon > 0$ ½ کوچکتر است باعث کاهش به یک اندازه دلخواه در سرعت که به x ورودی وابسته است می شود.

از طرفی با قضیه ٦-٢-١ ج, سرعت مورد نظر از کاهش با $f(x) = 1/((1/2) - e(x))$ محدود است , چون احتمال $\Phi(f(x), e(x))$ از بدست آوردن یک پاسخ نادرست با تکرار اجرای برنامه برای $f(x)$ بار بزرگتر است از $1/(2\lambda^6)$. همچنین بویژه زمانی که $f(x)$ از یک چند جمله ای در $|x|$ بیشتر است , سرعت مورد نظر کاهش یافتن بزرگتر است .

خروجی برنامه های احتمالی

قضیه ای که گذشت , تعاریف زیر را به دنبال دارد .

برنامه احتمالی p بیان می کند برای داشتن یک خروجی y روی ورودی x اگر احتمال p بزرگتر از ۱/۲ باشد یک محاسبه با خروجی y روی ورودی x داریم . اگر چنین y ای وجود نداشته باشد آنگاه خروجی p روی ورودی x تعریف نشده است .

برنامه احتمالی p بیان می کند برای محاسبه تابع f(x) اگر p روی هر ورودی x احتمال 1- e(x) برای یک محاسبه پذیرش با خروجی f(x) , داشته باشد , آنگاه :

الف) e(x)<1/2 هرگاه f(x) تعریف شده باشد .

ب) e(x) تعریف شده نیست , هرگاه f(x) تعریف نشده باشد .

e(x) احتمال خطای p گفته می شود . احتمال خطای e(x) احتمال خطای محدود شده است اگر یک ثابت کوچکتر از ۱/۲ وجود داشته باشد که e(x)≤ برای تمام x هایی که f(x) روی آنها تعریف شده است .

P که یک برنامه احتمالاتی با خطای محدود است اگر یک احتمال خطای محدود شده داشته باشد.

در طول بحث قبل نشان داده شد که برنامه های احتمالی , که پیچیدگی زمانی چندجمله ای و احتمال خطای محدود شده دارند , برنامه های خوبی هستند .

۳-۶ مبدل تورینگ احتمالاتی

در مطالعه محاسبات غیراحتمالی در مدل های نظری قطعی و غیرقطعی

مبدل های تورینگ به کار می رود . برای مطالعه ی محاسبات احتمالی ما از مدل های نظری مشابه که مبدل های تورینگ احتمالی نامیده می شوند استفاده خواهیم کرد .

به بیان ساده تر , یک مبدل تورینگ احتمالی , یک مبدل تورینگ است که غیر قطعیت را به صورت تصادفی بودن نشان می دهد . معمولا یک مبدل تورینگ احتمالی یک مبدل تورینگ

$$M=<Q,\Sigma,\Gamma,\Delta,\delta,q_0,B,F>$$

است که محاسبات آن به روش زیر تعریف می شود . به دنباله C از حرکت های M یک محاسبه می گوید, اگر دو شرط زیر برقرار باشد :

الف) C از یک پیکربندی اولیه شروع می شود .

ب) زمانی که C متناهی است , در یک پیکربندی پذیرش یا یک پیکربندی غیر پذیرش که حرکت در آن ممکن نیست , خاتمه می یابد.

اگر محاسبه M در یک پیکربندی پذیرش خاتمه یابد محاسبه پذیرفته شده است در غیر اینصورت بیان می شود که محاسبه پذیرفته نشده است.

با این تعریف، یک مبدل تورینگ احتمالی ممکن است روی یک ورودی مفروض , هم محاسبه ی پذیرفته شده و هم محاسبه ی پذیرفته نشده داشته باشد .

هر محاسبه مبدل تورینگ احتمالی شبیه مبدل تورینگ غیرقطعی است . تنها استثناء به محض رسیدن به پیکربندی که در آن بیشتر از یک حرکت ممکن است , به وجود می آید. در این مورد انتخاب بین حرکات ممکن با احتمال مساوی برای هر حرکت , تصادفا انجام می شود.

تابعی که یک مبدل تورینگ احتمالی و احتمال خطای آن را محاسبه می کند مشابه برنامه های احتمالی تعریف می شود . ماشینهای تورینگ احتمالی مشابه مبدلهای تورینگ احتمالی تعریف می شوند .

یک ماشین تورینگ احتمالی M بیان می کند که یک زبان L را می پذیرد اگر :

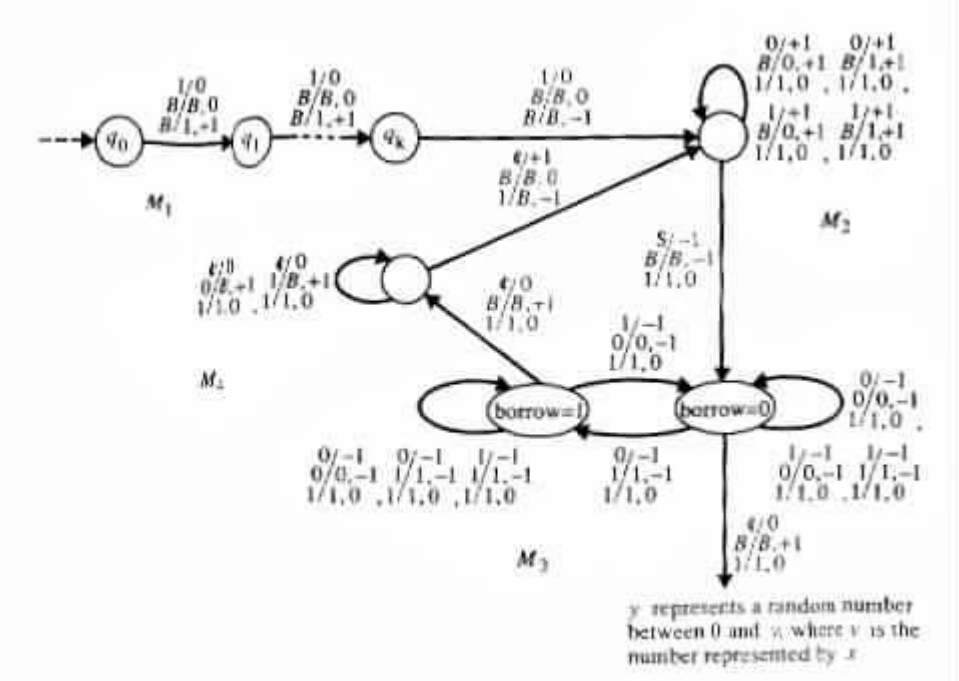
الف) روی ورودی X از M , L احتمال $1 - e(x) > 1/2$ را برای یک محاسبه ی پذیرش , داشته باشد.

ب) روی ورودی X خارج از زبان M , L احتمال $1 - e(x) > 1/2$ را برای یک محاسبه ی غیر پذیرش داشته باشد.

$e(x)$ احتمال خطای M را بیان می کند , احتمال خطا محدود شده است , اگر $\leq$

$e(x)$ برای همه X ها یک ثابت $\varepsilon > 1/2$ داشته باشد . به M یک ماشین تورینگ احتمالی با خطای محدود گفته می شود اگر احتمال خطای محدود شده داشته باشد.

مثال ۶-۳-۱. شکل ۶-۳-۱ را در نظر بگیرید تا به حال در تحلیل زمان مورد نیاز برنامه ها بدترین حالت را در نظر می گرفتیم. برنامه هایی که بدترین حالتشان خوب است، بدیهی است که برای حل کردن مسئله ها مطلوبترین انتخاب است. به هرحال در خیلی از شرایط ممکن است زمانی که برنامه های بهتری وجود ندارد ما با برنامه هایی که برای هر ورودی عموماً رفتار خوبی دارند نیز راضی شویم. در حقیقت گاهی ممکن است برنامه هایی برای ما منع کننده باشد که حتی احتمال ایجاد پاسخ نادرست را نیز شامل شود.



شکل ۶,۳,۱ بخشی از یک ماشین تورینگ احتمالی که یک عدد تصادفی را تولید می

کند.

نمودار انتقال یک بخش M از یک ماشین تورینگ احتمالی داده شده است . روی ورودی M, X با احتمال $1 - (1/2)^k$ یک عدد تصادفی بین ۰ تا v پیدا می کند . فرض می شود X رشته ای در $\{0, 1\}^*$ که با ۱ شروع می شود , باشد و فرض می شود v عدد طبیعی است که با X نشان داده می شود .نمایش دودویی عدد تصادفی , در اولین نوار کار کمکی ذخیره می شود.

در M هر محاسبه به وسیله به کار بردن M_1 برای ثبت مقدار K شروع می شود . سپس M مکررا M_3 , M_2 و M_4 را برای تولید یک رشته تصادفی Yبه طول |x| به کار می برد و کنترل می کند که Y یک عدد صحیح بزرگتر از V را نشان ندهد. M زیر محاسبات خود را با موفقیت آمیز به پایان می رساند اگر و فقط اگر رشته Y را با K بار تلاش پیدا کند .

M_1 مقدار K را به صورت یکانی در دومین نوار کمکی M ثبت می کند . در اولین نوار کمکی M_2 , M یک رشته تصادفی Y با طول |x| را روی {۰،۱} تولید می کند . M_3 کنترل می کند که X یک عدد بزرگتر از Y نشان داده شده را نشان بدهد . M_3 عمل چک کردن را با شبیه سازی عمل تفریق انجام می دهد . M_4 رشته ی Y را که در اولین نوار کار کمکی ذخیره شده است , پاک می کند .

تعداد اجرای چرخه های M_3 , M_2 و M_4 با طول K از رشته ۱ ...1 که در دومین نوار کار کمکی ذخیره شده است , کنترل می شود . در پایان هر چرخه , رشته بوسیله یک نماد نشان داده می شود .

احتمال اینکه , M_2 یک رشته ای که یک عدد بین ۰ تا V را نشان دهد , تولید خواهد کرد , $(v+1)/2^{|x|} \geq 1/2$ می باشد . احتمال اینکه یک رشته در K چرخه تولید شود , برابر است با

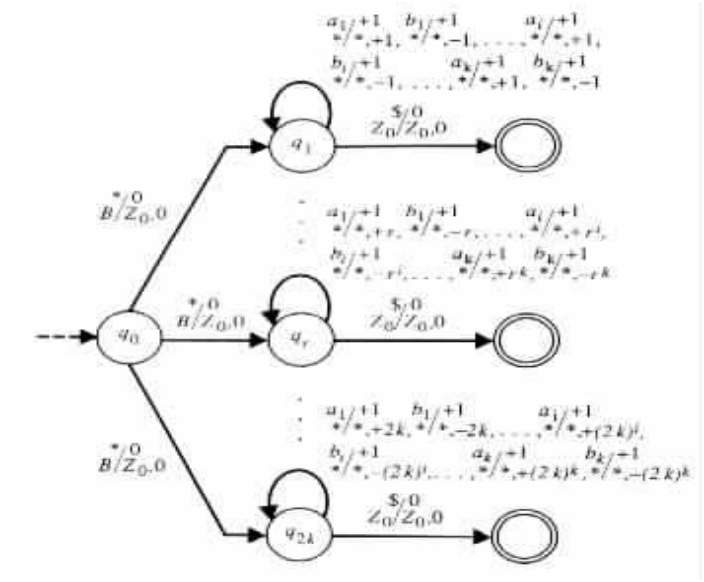
(Probability of generating the string in the first cycle)
+ (Probability of generating the string in the second cycle, but not in the first cycle)
⋮
+ (Probability of generating the string in the k th cycle, but not in the first $k-1$ cycles).

جمع این احتمالات برابر است با :

$$\begin{aligned}
&\tfrac{v+1}{2^{-|\alpha|}}\left(1+\left(1-\tfrac{v+1}{2^{-|\alpha|}}\right)+\left(1-\tfrac{v+1}{2^{-|\alpha|}}\right)^2+\cdots+\left(1-\tfrac{v+1}{2^{-|\alpha|}}\right)^{k-1}\right)\\
&= \tfrac{v+1}{2^{-|\alpha|}}\left(\left(1-\tfrac{v+1}{2^{-|\alpha|}}\right)^k-1\right)\left(\left(1-\tfrac{v+1}{2^{-|\alpha|}}\right)-1\right)^{-1}\\
&= 1-\left(1-\tfrac{v+1}{2^{-|\alpha|}}\right)^k\\
&\geq 1-(1/2)^k.
\end{aligned}$$

ماشین تورینگ احتمالی در مثال زیر , یک ماشین پشته ای احتمالی است که یک زبان وابسته به متن را می پذیرد . این ماشین می تواند به نحوی تغییر کند که که دقیقا n+2 حرکت روی هر ورودی با طول n ایجاد کند , در حالی که هر یک نوار کار کمکی , ماشین تورینگ غیراحتمالی به نظر می رسد که بیش از n+2 بار برای برای پذیرفتن زبان نیاز دارد .

مثال ۲-۳-٦. یک ماشین تورینگ احتمالی M با یک نوار کار کمکی در شکل ۲-۳-٦



شکل ۲,۳,٦ یک ماشین تورینگ احتمالی M با یک نوار کار کمکی ا که زبان $L=\{ w$ | w is in $\{a_1, b_1, \ldots, a_k, b_k\}^*$ را می پذیرد و W تعداد یکسان a_i's as b_i's دارد . $\delta_{i,j}^{+1}$ نشان دهنده این است که دنباله ای از قوانین انتقال j برای هر $1 \leq i \leq k$ که هد نوار کار کمکی را حرکت می دهد , مکان j را در راست قرار می دهد . این دنباله محتویات نوار را تغییر نمی دهد .

زبان $L=\{ w \mid w \text{ is in } \{a_1, b_1, \ldots, a_k, b_k\}^*$ را می پذیرد و W , تعداد a ها و b ها با هم برابرند برای هر $1 \leq i \leq k$ دارد .

M روی هر ورودی مفروض W , محاسبات را بوسیله انتخاب تصادفی از برخی اعداد بین ۱ تا ۲K شروع می کند . بنابراین از حالت اولیه q_0 به حالت معادل q_r حرکت می کند . بعلاوه M ، Z_0 را روی نوار کار کمکی می نویسد . سپس M , هد نوار کار کمکی را از مکان r^i به راست برای هر نماد a_i خوانده شده حرکت می دهد و r^i را برای هر نماد b_i خوانده شده در چپ قرار می دهد .

در پایان هر محاسبه هد نوار کار کمکی + $n(a_k) - n(b_k))r^k$ $n(a_1) - n(b_1))r +$ + $(n(a_2) - n(b_2))r^2$ در راست Z_0 قرار داده می شود , جایی که $n(c)$ نشان دهنده تعداد دفعات ظاهر شدن C در W است . اگر هد آن روی Z_0 قرار داده شود , آنگاه ورودی پذیرفته شده است , وگرنه ورودی رد می شود .

با این دستورالعمل , M هر ورودی W از زبان L را می پذیرد . متناوبا M ممکن است همچنین بعضی رشته هایی که در L نیستند را با احتمال $e(x) = \varepsilon < ½$ را در جایی که $\varepsilon = (k-1)/(2k)$ می پذیرد .

این تساوی $\varepsilon = (k-1)/(2k)$ برقرار است زیرا اگر W در L نباشد آنگاه $\neq 0$ $n(a_k)$) + ۰= تساوی مواردی چنین در.است برقرار یک i حداقل برای $n(a_i) - n(b_i)$ $(n(a_1) - n(b_1))r$ - $n(b_k))r^k$ + می تواند با حداکثر K-1 مقدار غیرصفر r ارضا شود . گرچه امکان تعیین ۲k برای r وجود دارد . در نتیجه احتمال اینکه r یک مقداری که تساوی را ارضا کند بگیرد , بزرگتر از $(k - 1)/(2k)$=ε نیست .

محدودیت احتمال خطای $e(x)$ می تواند بوسیله تعیین تصادفی r با یک مقدار از { $k/ε$1} , . . . , تا هر مقدار دلخواه کاهش پیدا کند.

M بیشتر از $T(n) = (2k)^k n + 2$ حرکت را روی ورودی به طول n نمی گیرد . M می تواند طوری تغییر کند که دقیقا $T(n) = n+ 2$ حرکت با ذخیره کردن ارزش واحد $(2k)^2$ در نوار کار کمکی را ایجاد کند . در این مورد مقدار میانی کوچکتر در کنترل حالت نهایی M ذخیره شده است .

٦-٤ زمان چند جمله ای احتمالی

در مورد مبدل های تورینگ قطعی و غیرقطعی ، هر حرکت مبدل تورینگ احتمالی

فرض می شود که یک واحد زمان را می گیرد. زمان یک محاسبه برابر است با تعداد حرکات طی آن محاسبه. مکان یک محاسبه برابر است با تعداد مکان های ملاقات شده در نوار کار کمکی که بیشترین تعداد را دارد .

پیچیدگی زمانی احتمالاتی

یک مبدل تورینگ احتمالی M بیان می کند که T(n) محدودیت زمانی دارد یا دارای T(n) پیچیدگی زمانی است اگر M در زمان T(n) در هر محاسبه روی هر ورودی به طول n متوقف شود . اگر T(n) یک چند جمله ای باشد آنگاه به M چندجمله ای با زمان محدود یا از پیچیدگی زمانی چندجمله ای نیز گفته می شود .

M بیان می کند که T(n) محدودیت زمانی مورد انتظار را دارد یا از T(n) پیچیدگی زمانی مورد انتظار است اگر برای هر ورودی x از M تابع T(n) رابطه زیر را ارضا کند .

$$T(|x|) \geq \sum_{j=0}^{\infty} \left(\begin{array}{l} \text{Probability that } M \text{ on input } x \text{ will have} \\ \text{a computation that takes exactly j moves} \end{array} \right) \cdot j$$

اگر T(n) یک چندجمله ای باشد آنگاه M با زمان محدود شده ی مورد انتظار چندجمله ای نامیده می شود یا از پیچیدگی زمانی مورد انتظار چندجمله ای است .

آرگومان ها شبیه آرگومانهای فرض شده برای نظریه ی چرچ در قسمت ٤,١ می باشد و برای محاسبات متوالی مانند قسمت ٥,١ فرض می شود . همچنین برای نظریه ی زیر در نظر گرفته می شود

نظریه محاسبات احتمالی: یک تابع که به طور ماشینی با کمک انتخابهای احتمالی محاسبه پذیر است , همچنین می تواند به وسیله ی یک مبدل تورینگ احتمالی چندجمله ای مرتبط با پیچیدگی زمانی مورد انتظار محاسبه شود .

رده های پیچیدگی احتمالی

رام شدنی بودن مسائل نسبت به زمان احتمالی بوسیله ی "وجود مبدل های تورینگ احتمالی با خطای محدود و دارای پیچیدگی زمانی چندجمله ای برای حل این مسائل" تعیین می شود.برای روشن شدن موضوع ، رده های زیر از مسائل تشخیص زبان , از فواید آن هستند .

BPP -- رده ی مسائل عضویت زبان ها در { L | L یک زبان پذیرفته شده بوسیله ی یک ماشین تورینگ احتمالی با خطای محدود دارای پیچیدگی زمانی چندجمله ای باشد }.

RP – رده ی مسائل عضویت زبان ها در { L | L یک زبان پذیرفته شده بوسیله ی چندجمله ای با زبان محدود و ماشین تورینگ احتمالی M که دو شرط زیر را برای برخی ثابت های ε >۱ برقرار می کند .

الف) روی ورودی x از L،M یک محاسبه ی پذیرش با احتمال $1 - e(x) \geq 1 - \varepsilon$ دارد .

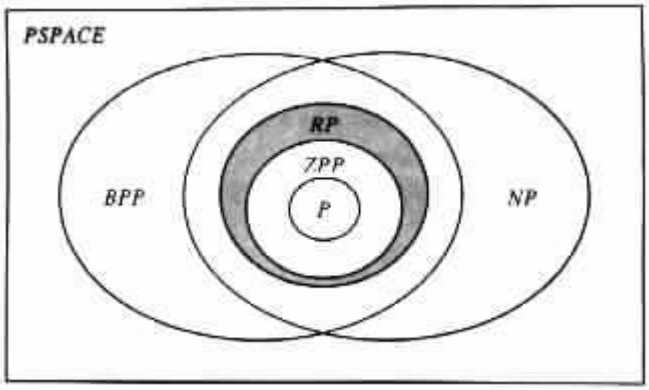
ب) روی ورودی x خارج از زبان L ، M فقط محاسبات غیرپذیرش دارد . }

ZPP -- رده ی مسائل عضویت زبان ها در

{ L | L یک زبان پذیرفته شده بوسیله ی ماشین تورینگ احتمالی است که خطای احتمالی صفر و پیچیدگی زمانی مورد انتظار چندجمله ای را دارد .}

روابط بین رده های پیچیدگی احتمالاتی و غیراحتمالاتی

ارتباط بین رده های مختلف مسائل به خوبی روابط آنها در رده های مطالعه شده در فصل ۵ در شکل ٦-٤-۱ شرح داده شده است .



شکل ٦,٤,۱. سلسله مراتب برخی رده های مسائل

کامل بودن هیچکدام از شمول ها مشخص نیست . ارتباط به صورت زیر ثابت می شود

.

قضیه ٦-٤-١ : BPP در PSPASE قرار گرفته است .

اثبات : هر مسئله K در BPP را در نظر بگیرید . L زبانی را نشان می دهد که K استنتاج می کند . با این تعریف از BPP , یک ماشین تورینگ احتمالی M_1با خطای محدودو زمان محدود چندجمله ای وجود دارد که زبان L را می پذیرد. $\varepsilon < ½$ را یک ثابت از M_1 که احتمال خطایش محدود شده است قرار دهید و $P(n)$ را پیچیدگی زمانی M_1 قرار دهید .

بدون از دست دادن عمومیت , فرض می شود M_1 یک ثابت K دارد چنان که هر دو حرکت احتمالی M_1 دقیقا K گزینه دارد . (هر ماشین تورینگ احتمالی می تواند با داشتن خاصیت K ای که کوچکترین ضرب از تعداد گزینه ها در حرکت های مختلف ماشین تورینگ است , تعریف شود . بعلاوه فرض می شود M_1 چندجمله ای $q(n)$ دارد به صورتی که در هر محاسبه در هر ورودی x دقیقا$q(|x|)$ حرکت احتمالی انجام می دهد . در نتیجه M_1 روی هر ورودی x دقیقا$k^{q(|x|)}$ محاسبه ی ممکن دارد. با هر محاسبه یک احتمال مساوی برای هر رویداد دارد .

از M_1 یک ماشین تورینگ قطعی M_2 می توان ساخت که زبان L را بپذیرد . M_2 بر دو خاصیت زیر از M_1 متکی است .

الف) اگر x در L باشد آنگاه M_1 روی ورودی x حداقل احتمال $1-\varepsilon > 1/2$ را برای داشتن یک محاسبه ی پذیرش دارد .

ب) اگر x در L نباشد آنگاه M_1 روی ورودی x حداقل احتمال $1-\varepsilon > 1/2$ را برای داشتن یک محاسبه ی غیرپذیرش دارد .

روی یک ورودی مفروض x، M_2تعیین می کند کدامیک از خاصیت های بالا برقرار هستند و بنابراین تصمیم می گیرد که آیا ورودی پذیرفته می شود یا رد می شود .

یک ورودی x داده شده است . ماشین تورینگ M_2 محاسبات را بوسیله ی محاسبه ی $p(|x|)$ شروع می کند . سپس در یک زمان M_2 تمام ردیف های قوانین انتقال M_1 را که حداکثر طول $p(|x|)$ دارند لیست می کند .برای هر دنباله M_2 کنترل می

کند که این ردیف متناظربا محاسبه ی M_1 باشد.M_2 تعیین می کند که محاسبه ی M_1 رد شده است یا پذیرفته شده است . بعلاوه M_2 تعداد m_a ی محاسبات پذیرفته شده و تعداد m_r محاسبات پذیرفته نشده را می شمارد .

M_2 ورودی x را می پذیرداگر تعیین کند که احتمال $(m_a/(m_a + m_r)$ برای پذیرش x توسط M_1 بیشتر از ۱/۲ باشد. یعنی ،اگر $m_a > m_r$ باشد . M_2 , اگر تعیین کند که احتمال $(m_r/(m_a + m_r)$ برای رد کردن x توسط M_1 بیشتر از ۱/۲ باشد , x را رد می کند یعنی، $m_r > m_a$ برقرار باشد .

مسئله غیر اول بودن مثالی از مسائل در رده ی RP است (مثال ٦-۲-۱ را ببینید) .

برای RP نتیجه ی زیر برقرار است:

قضیه ی RP.۲,٤,٦ در NP∩BPP قرار دارد .

اثبات : هر مساله K در RP را در نظر بگیرید . L زبانی است که K را استنتاج می کند . با این تعریف RP به صورت زیر یک ثابت ε< ۱ > و یک ماشین تورینگ با زمان محدود چندجمله ای M_1 که شرایط زیر را ارضا کند موجود است.

الف) اگر x در L باشد , آنگاه M_1 روی x یک احتمال $0 < ε - 1$ برای داشتن یک محاسبه ی پذیرش دارد .

ب) اگر x در L نباشد , آنگاه M_1 روی x فقط محاسبات غیر پذیرش دارد .

L بوسیله ی یک ماشین تورینگ غیرقطعی M_2 شبیه M_1 و با پیچیدگی زمانی یکسان پذیرفته می شود . تنها تفاوت این است که M_2 هر حرکت احتمالی M_1 را غیرقطعی در نظر می گیرد . در نتیجه RP در NP است .

M_1 می تواند بوسیله ی ماشین تورینگ احتمالی با خطای محدود M_3 با پیچیدگی زمانی مشابه نیز شبیه سازی شود . K را هر ثابت $ε^k < 1/2$ قرار می دهیم . آنگاه M_3 ، Kمحاسبه از M_1 را روی ورودی مفروض x شبیه سازی می کند . M_3، x را می پذیرد اگر M_1، x را در هر محاسبه ی شبیه سازی شده بپذیرد . در غیراین صورت M_3 x را رد می کند . پس RP در BPP نیز هست .

سرانجام برای ZPP نتیجه ی زیر نشان داده می شود .

قضیه ی ۳,٤,٦. ZPP در RP قرار دارد .

اثبات : هر ماشین تورینگ احتمالی M_1 که احتمال اشتباه صفر دارد , را در نظر بگیرید . $T(n)$ نشان دهنده ی پیچیدگی زمانی موردانتظار M_1 است . فرض می شود که $T(n)$ یک چندجمله ای درجه ی n است. از M_1 یک ماشین تورینگ احتمالی M_2 به صورت زیر می تواند ساخته شود . یک ورودی x داده شده است, ماشین تورینگ احتمالی M_2 محاسبات را بوسیله ی ارزیابی $T(x)$ شروع می کند . سپس M_2 روی ورودی x برای ثابت $c > 1$ حرکت های M_1 را شبیه سازی می کند . M_2 در حالت پذیرش متوقف می شود اگر طی شبیه سازی به یک حالت پذیرش M_1 برسد . در غیراین صورت M_2 در حالت غیر پذیرش متوقف می شود .

با این ساختار , اگر x در $L(M_1)$ نباشد M_2 روی ورودی x محاسبه ی پذیرش ندارد . از طرف دیگر اگر x در $L(M_1)$ باشد آنگاه M_2 با احتمال مساوی برای اینکه M_1 یک محاسبه ی پذیرشی که بیشتر از $cT(|x|)$ حرکت نیاز دارد در حالت غیرپذیرش متوقف می شود. احتمال خطای $e(x)$ برابر است با

$$\sum_{i=cT(|x|)+1}^{\infty} p_i$$

که p_i احتمال آن را روی x نشان می دهد M_1 یک محاسبه ای که دقیقا i مرحله می گیرد , دارد .

حالا

$$\begin{aligned}
T(|x|) &\geq \bar{t}(x) \\
&= p_0.0 + p_1.1 + \Lambda + p_{cT(|x|)}.cT(|x|) + \Lambda \\
&\geq p_0.0 + p_1.1 + \Lambda + p_{cT(|x|)}.cT(|x|) + (cT(|x|)+1)\sum_{i=cT(|x|)+1}^{\infty} p_i \\
&= p_0.0 + p_1.1 + \Lambda + p_{cT(|x|)}.cT(|x|) + (cT(|x|)+1)e(x) \\
&= (cT(|x|)+1)\,e(x) + K
\end{aligned}$$

در نتیجه x،M_2 را با احتمال زیر می پذیرد .

$$1-e(x) \geq 1-\frac{T(|x|)}{cT(|x|)+1}$$
$$\geq 1-1/c$$

فصل هفتم

محاسبات موازی

اهداف

در پایان فصل، دانشجو با مفاهیم زیر آشنا می‌شود:

برنامه های موازی

ماشینهای دستیابی تصادفی موازی

مدارها

خانواده مدارهای یکنواخت و محاسبات ترتیبی

خانواده مدارهای ترتیبی و PRAM ها

مقدمه

در فصل هفتم توانایی و کاربرد پذیری تصادفی بودن برای سرعت محاسبات ترتیبی را بررسی کردیم . در این فصل این مطلب را که چگونه موازی سازی به هدف مشابهی می رسد بحث می کند. در اولین بخش مفهوم موازی سازی در برنامه ها معرفی می شود . تعمیمی از RAM ، ماشین دستیابی تصادفی موازی یا PRAM نامیده میشود که اجازه می دهد از یک انتزاع سطح بالا برای محاسبات موازی در بخش دو استفاده

کنیم . در سومین بخش خانواده ی از مدارهای بولی را با هدف آماده کردن یک سطح سخت افزاری انتزاعی برای محاسبات موازی معرفی می کنیم. در بخش چهار مسایل با سازگاری طبقه عمومی از خانواده مدارهای بولی سر و کار دارند که بعنوان انتزاع سطح پایین مورد بررسی قرار می گیرد و دسته ی محدود برای چنین هدفی پیشنهاد می شود . خانواده ی از دسته ی محدود، خانواده یکنواخت از مدار ها نامیده می شود . در بخش پنجم خانواده یکنواخت از مدارها را در برابر محاسبات ترتیبی نشان می دهد.خصوصا ، نشان می دهد که موازی سازی باعث افزایش مسائل حل شدنی نمی شود. علاوه بر آن در کاربردی از موازی سازی که در روند رو به رشد قابل توجهی به محاسبات امکان پذیر سرعت می بخشد بحث می کند .در بخش ٦ از این فصل بحث را با رابطه PRAM وخانواده یکنواخت از مدارها به پایان میبریم.

۷-۱ برنامه های موازی

برنامه موازی $\hat{p}$ یک سیستم $<p,x,y>$ است از تعداد زیادی برنامه های ترتیبی قطعی $p_1,p_2...$ ،تعداد زیادی متغیر ورودی... $x(1),x(2)$ و تعداد زیادی متغیر خروجی $y(1),y(2)...$. برنامه های ترتیبی $p_1,p_2...$ فرض می شود که به صورت مشابه باشد به جز هر P_i که اندیس i به خودش اشاره می کند. برای هر جفت از اندیس های j,i برنامه ترتیبی P_j. می تواند از برنامه ترتیبی Pi با جایگذاری هر اشاره گر i در P_i با اشارگر j بدست اورده شود

در شروع محاسبات : ورودی $\hat{p}$ در متغیر های ورودی ذخیره می شود. ورودی شامل N مقدارذخیره شده از $x(1),x(2),...x(N)$ است که هر یک از متغییرها یکی از مقادیر ورودی را نگه میدارند. در طول محاسبات ، $\hat{p}$ ، . $P_1,$ . . , P_m . برای بعضی m که به ورودی وابسته است بکار می گیرد.فرض می شود که هر P_i مقدار N و m را می شناسد.به محض متوقف شدن،خروجی $\hat{p}$ در متغیر های خروجی قرار می گیرد. خروجی شامل K مقدار

$y(1),...,y(k)$ است که هر یک از متغیرها یکی از مقادیر خروجی رانگه می دارد.

هر مرحله از محاسبه شامل چهار فاز است به صورت زیر است:

a. هر P_i یک مقدار از متغیر های ورودی $X(1), . . . , X(N)$ را می خواند

b. هر P_i برخی محاسبات داخلی را انجام می شود

c. هر ممکن است بر روی یکی از متغیرهای خروجی $y(1),y(2),...$ بنویسد

d. $P_1, . . . , P_m$ اطلاعات مطلوب را بین خودشان مبادله می کنند .

هر یک از مراحل ، همگام می شوند تابه وسیله همه برنامه های ترتیبی $P_1, . . . , P_m$ به طور موازی انجام شوند .

اگرچه دو یا تعداد بیشتری برنامه ترتیبی قابلیت خواندن هم زمان از متغیرهای ورودی یکسان را دارد ولی در هیچ مرحله ای قادر نیست آنها رادر خروجی یکسان بنویسد

عمق محاسبه یک برنامه موازی $\hat{p} = <P, X, Y>$ تعداد مراحل اجرا شده درطول محاسبه است. یک برنامه موازی از پیچیدگی عمق $D(N)$ گفته می شود اگر برای هر N همه محاسباتش ،روی ورودی که شامل N مقدار باشدکه حداکثر عمق $D(N)$ داشته باشد. برنامه موازی $\hat{p}$ از پیچیدگی اندازه $Z(N)$گفته می شود اگر هیچ برنامه های ترتیبی به غیر از$P_{Z(N)}$, . . . ,P_1 در هر ورودی که شامل N مقدار میشود بکار نگیرد.

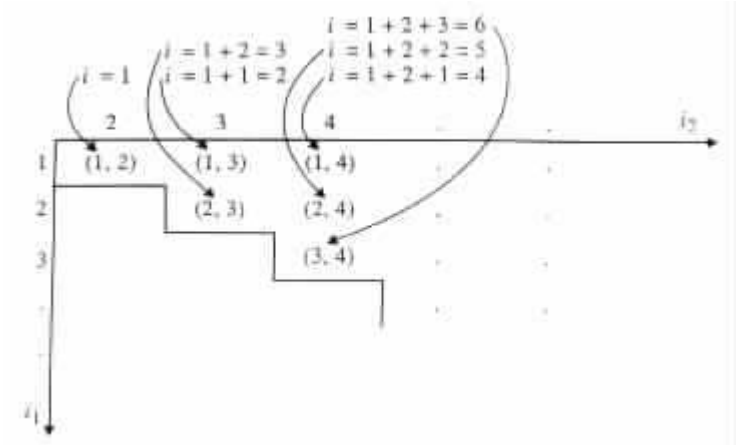
زمان مورد نیاز برای محاسبه در برنامه ی موازی و برنامه با پیچیدگی زمانی می تواند از راه مشابه تعریف شود. هر چند در اینجا این قبیل مفاهیم قابل

اندازه گیری نیستند زیرا ما هنوز مشخص نکرده ایم که چه طور برنامه های ترتیبی با هم ارتباط برقرار می کنند .

مثال ۷-۱-۱ مسئله Q که کوچکترین مقدار در مجموعه مفروض S انتخاب می کند ، در نظر بگیرید.توجه شما را به برنامه های موازی که در هر مرحله برنامه های ترتیبی را محدودمی کند تا اطلاعات را از یک برنامه ترتیبی نه بیشتر دریافت کند، محدود می کنیم.

برنامه بوسیله یک برنامه موازی $\hat{p}_1 = <P, X, Y>$ با پیچیدگی اندازه $Z(N)=N(N-1)/2$و پیچیدگی عمق ثابت ، قابل حل است. که N تعداد اعضای مجموعه مفروض S را مشخص می کند. برنامه موازی می تواند از روش brute-force برای چنین هدفی استفاده کند .

به ویژه هرجفت (i_1, i_2) ،, به طوری که $1 \le i_1 \le i_2 \le N$ ، متناظر با i های مختلف انچنان که $1 \le i_1 \le i_2 \le N(N-1)/2$. برای مثال تناظر می توانند به شکل $i = 1 + 2 + \ldots + (i_2 - 2) + i_1 = (i_2 - 2)(i_2 - 1)/2 +$ i_1 باشد (شکل۷-۱-۱)



$P_{(i1,i2)}$ برنامه ترتیبی P_i را مشخص میکند ، که (i_1, i_2) جفت متناظر با i هستند .

هـر محاسـبه $\hat{p}$ بـا مرحلـه ای کـه در آن جفـت (i_1, i_2) متنـاظر بـا i کـه $1 \le i \le N(N-1)/2$ از هر P_i ناشی می شود، شروع می گـردد. محاسـبه بـا مرحلـه ۲ ادامـه پیـدا مـی کنـد کـه در آن هـر $P_{(i1,i2)}$ عناصـری از S کـه در $X(i_1)$ و $X(i_2)$ ذخیره می شود را می خواند.علاوه بر این، در مرحله سـوم هـر $P_{(i1,i2)}$ مقادیر خوانده شده از $X(i_1)$ و $X(i_2)$ را مقایسه می کند و یـک نتیجـه "منفی "به P_{i1} یا P_{i2} مبادله می کند . اگر $X(i_1) \ge X(i_2)$ باشد نتیجه به P_{i1} در غیر این صورت نتیجه به P_{i2} مبادله می شود. در طـول مرحلـه چهـارم ، تنهـا برنامه ی ترتیبی فعال P_j است که $1 \le j \le N$ که نتیجه منفـی دریافـت نکـرده است. در طول این مرحله، P_j مقـدار $X(j)$ را مـی خوانـد و آن را بـر روی $y(1)$ می نویسد.محاسبه بعد از مرحله چهارم به پایان می رسد.

برنامه Q همچنین بـه وسـیله یـک برنامـه مـوازی $\hat{p}_2 = <P, X, Y>$ بـا پیچیدگی اندازه $Z(N)=N/2$ وپیچیدگی عمـق $D(N) = O(\log N)$. قابـل حل است. در این مورد برنامه به سادگی مکررا تقریبا نیمـی از عناصـر را از S حذف می کند تا این که S با یک عنصر باقی بماند .

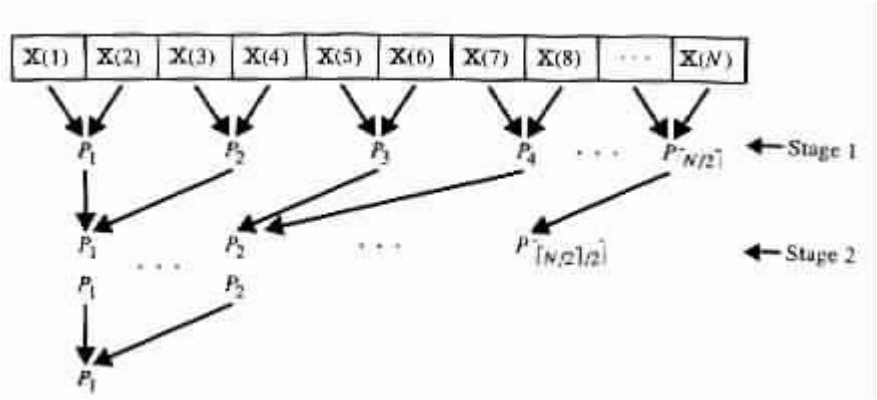
در اولین مرحله از هر محاسـبه هـر P_i، $1 \le i \le N/2$ ،مقـادیر دخیـره شـده در $X(2i-1)$ و $X(2i)$ را میخواند. علاوه بر ان هر P_i مقـایر خوانـده شـده را مقایسه می کنداگر $X(2i) < X(2i-1)$ بود پس P_i بـا $P_{i/2}$ مقـدار $X(2i-1)$ را مبادله می کند . وگرنه P_i با $P_{i/2}$ مقدار $X(2i)$ را مبادله مـی کنـد . در پایـان مرحله اول $P_1, \ldots, P_{\lceil n/2 \rceil}$ عناصری از S که هنوز حذف نشده است نگـه می دارند.

در شروع هر مرحله متوالی از محاسبه ، یک برنامـه ترتیبـی P_i خـود را فعـال تعیین میکند اگـر وفقـط اگـر مقـداری از S را در مرحلـه قبلـی مبادلـه کـرده باشد.در طی مرحله مفروض هر P_i فعال مقادیر a_1 و a_2 را مقایسه مـی کنـد

که در مرحله قبلی مبادله شده بودند. i، اگر $a_1 < a_2$ باشد ،سپس P_i ،a_1 را به $P_{\lceil i/2 \rceil}$ مبادله میکند و در غیر این صورت P_i را a_2 به $P_{\lceil i/2 \rceil}$ مبادله میکند.

بعد از $O(\log N)$ مرحله فقط P_1 فعال است و تنها یک عنصر از S را نگه میدارد. سپس P_1 مقدار $Y(1)$ را مینویسد و محاسبه پایان می پذیرد.

جریان اطلاعات در $\hat{p}_2$ در طول محاسبه برنامه موازی توضیح داده شده است .



شکل ۷-۱-۲ جریان اطلاعات

به طور مشابه ، مسئله Q با برنامه موازی $\hat{p}_3 = <P, X, Y>$ با پیچیدگی اندازه $N/2 > Z(N)$ و پیچیدگی عمق $O(N/Z(N) + \log Z(N))$ قابل حل است.در اغاز هر محاسبه، هر P_i $m=Z(n)$ را محاسبه می کندو بطور مستقل در $O(N/m)$ مرحله کوچکترین مقادیر $X(\frac{N}{m}(i-1)+1),\ldots,X(\frac{N}{m}i)$ پیدا می کند .سپس مانند مورد قبلی p_2 به طور $P_1,\ldots,P_m$ موازی پیش می روند تادر $O(\log m)$ مرحله کوچکترین مقادیر بین مقادیر mکه نگه داری می شوند را تعیین کند.

۷-۲ ماشینهای دستیابی تصادفی موازی

مطالعه محاسبات ترتیبی به سمت مدلهای انتزاعی مانند RAM و مبدل

های تورینگ هدایت شده است . .RAM برای طراحی چنین محاسباتی مفید است و و مبدل های تورینگ در تحلیل آنها مفیدند.مشاهده محاسبات موازی به عنوان تعمیمی از محاسبات ترتیبی ، تعمیم های مشابهی برای مدلهای مربوط فراخوانی می کند.تعمیم RAM در طول چنین خطی در زیر توصیف شده است.

یک ماشین دستیابی تصادفی موازی یاPRAM ،یک سیستم <M, X, Y, A>,از تعداد نامتناهی RAM،, . . . M_1,M_2,. ، تا تعدادنامتناهی سلول ورودی . . . X(1),X(2),. ، تعدادنامتناهی سلول خروجی . . Y(1),Y(2),. , .وتعداد نامتناهی سلول های حافظه مشترک . . A(1),A(2),.تشکیل شده است . هر M_i پردازنده ای از M̂ نامیده میشود.فرض ی شود همه پردازنده ها . . . M_1,M_2,. با هم برابرند بجز توانایی اینکه هرM_i اندیس iی خود را تشخیص می دهد.

در شروع محا سبات،تعدادN مقادیر ورودی معرفی می شود که به ترتیب در X(N), . . . ,X(1), دخیره می شود. در پایان محاسبات مقادیر خروجی در K=>0, Y(K), . . . ,Y(1), ذخیره میشود. در طول محاسبات M̂، m پردازنده M_m, . . . M_1, استفاده میشود که m فقط به ورودی بستگی دارد. فرض میشود که هر یک از پردازنده هااز عددN–تعداد مقادیر ورودی– وعدد m – تعداد پردازنده ها – آگاه است .

هر مرحله شامل ۵ فاز زیر میباشد که به وسیله همه پردازنده ها به طور موازی انجام می شود :

a.هر پردازنده یکی از سلول های ورودی.X(N), . . . ,X(1), را می خواند.

b. هر پردازنده یکی از سلول های حافظه مشترک .. A(1),A(2),. را می خواند.

c. هر پردازنده تعدادی محاسبات داخلی انجام می دهد

d.هر پردازندهممکن است بر روی یکی از سلوهای خروجی . .,Y(2),Y(1) . بنویسد.

e. هر پردازنده ممکن است بر روی یکی از سلول های حافظه مشترک .,A(2),A(1) بنویسد.

دو چند پردازنده قابلیت خواندن همزمان از سلول یکسان دارد.اگر چه تداخل نوشتاری وقتی که دو یا چند پردازنده تلاش میکنند بر سلول یکسان بنویسند اتفاق می افتد .تداخل نوشتاری بر طبق گونه ای از PRAM'S در حال استفاده مورد برخورد قرار می گیرد. سه مورد از گونه های مختلف ممکن در زیر آمده است :

a.CREW(خواندن همزمان – نوشتن انحصاری) در این روش امکان تداخل نوشتاری وجود ندارد

b.COMMOMدر این روش همه پردازنده هایی که همزمان روی یک سلول حافظه یکسان ،می نویسند باید مقدار یکسان بنویسند.

c.PRIORITY.در اینگونه از تداخل نوشتاری بر طبق خواسته پردازندهM_i حل میشود که دارای کوچکترین اندیس i در میان پردازندههای درگیر با تداخل است.

طول Nاز ورودی(v_N , . . . ,v_1) در یکPRAM با طول نمایش نمونه برابر فرض میشود.

عمق محاسباتPRAM،و پیچیدگی عمق واندازه آن به نسبت طول n از ورودی در روشی مشابه با برنامه های موازی تعریف میشوند.زمان مورد نیاز محاسبه برای PRAMو پیچیدگی زمانی آن،تحت معیار ارزش لگاریتمی و یکنواخت ،به روش واضحی تعریف می شود.

وقتی به علت رابطه مشخص بین عددNاز مقادیر ورودی و طـول nاز ورودی ،مشکلی پیش نیاید ، M و n را می توان به جای هم به کار برد .

مثـــال۷-۲-۱ PRAM هـــای COMMONو PRIORITY , $\hat{M}$ =<M,X,Y,A> مشابه با برنامه های موازی $\hat{p}_i$ <P, X, Y> = از مثال ۷-۱-۱ ، می توانند برای حل مـسئله Q بـرای انتخـاب کـوچکترین عنـصر از مجموعه مفروض S با تعداد اعضای N ، استفاده شوند .مبادلـه بـین پردازنـده هـا میتوانـد بـه طـور غیـر مـستقیم از راه سـلول هـای حافظـه مـشترک A(1),A(2),. . . انجام شود .

خصوصا هر پیغام به M_i در A(i) ذخیره می شود . آنجایی کـه PRAMهـا شبیه $\hat{P}_1$ هستند هیچ مشکلی بوجود نمی آید زیرا همه پیغـام هـا دقیقـا مـشابه هستند. به طور متناوب ، آنجایی که PRAM ها مشابه P_2 یا P_3 هستند ،هـیچ مسُله ی رخ نمی دهد زیرا تداخل نوشتاری اتفاق نمی افتد.

طبق تعریف هر CREW PRAMیک COMMON PRAM نیز هست و هر COMMON PRAM نیز یک PRIORITY PRAM است. نتیجه زیر نشان می دهد که یک هر PRIORITY PRAM می تواند با یک CREW PRAM شبیه سازی شود.

قـضیه ۷-۲-۱هـر PRIORITY PRAM بـا پیچیـدگی انـدازه Z(n)و پیچیدگی عمق D(n) دارای یک CREW PRAM $\hat{M}$ معادل ،بـا پیچیـدگی اندازه $Z^2(n)$ و پیچیدگی عمق.D(n)log Z(n) می باشد .

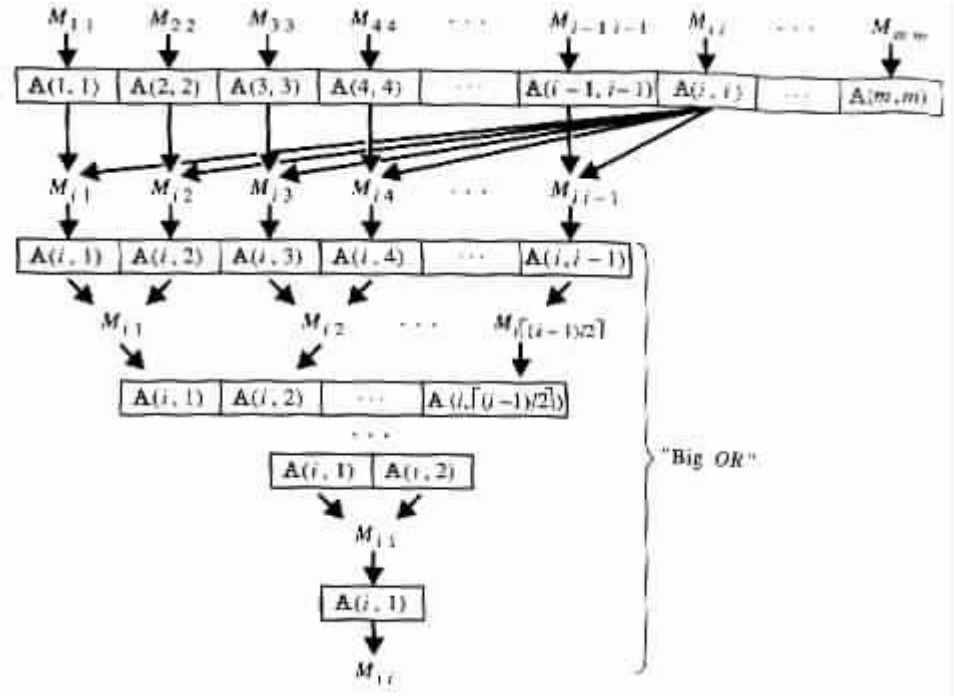
اثبات: هر PRIORITY PRAM $\hat{M}$= <M, X, Y, A> را در نظر بگیرید. $\hat{M}'$ = <M', X, Y, A> میتوان یک CREW PRAM به شکل زیر باشد،که پردازنده های آن با $M_{1\,1},\ldots,M_{1\,m}, M_{2\,1},\ldots,M_{2\,m},\ldots,M_{m\,1},\ldots$

$M_{m\,m}$, مشخص شده اند.

در ورودی مفروض، $\hat{M}'$محاسبات $\hat{M}$ را شبیه سازی می کند. $\hat{M}'$ پردازنده های$M_{m\,m}$, . . . ,$M_{2\,2}$,$M_{1\,1}$ برای پردازنده های ترتیبی مشابه . . . ,M_m,M_2,M_1 از $\hat{M}$ استفاده میکند.مشابها،$\hat{M}'$ مقادیر$A(m^2 + i)$را در سلول حافظه مشترک $A(i)$ که $i \geq 1$ ثبت می کند. تفاوت اصلی زمانیست که یک نوشتن شبیه سازی شود .

در چنین موردی،هر$M_{j\,j}$ با هر$M_{i\,i}$ نشانی سلولی را که M_j قصد نوشتن دارد، مبادله می کند . سپس هر$M_{i\,i}$ از آدرس آنها تعیین می کند، که آیا این دارای بالاترین تقدم نوشتن در سلول انتخاب شده، می باشد، وبر اساس آن تصمیم می گیرد که ایا عمل نوشتن انجام شودیا نه . برای چنین هدفی M_i پردازنده های$M_{i\,i-1}$, . . . ,$M_{i\,1}$, و سلول حافظه مشترک . . . ,$A(i, 1)$, $A(i, i - 1)$, را به کار می گیرد که $A(i_1, i_2)$ نماینده برای $A((i_1 - 1)m + i_2)$ است.

برای حل شدن تداخل نوشتاری، هر $M_{j\,j}$ در $A(j, j)$، نشانی که M_j قصد دارد بر روی آن بنویسدرا ذخیره می کند. (شکل ۷-۲-۱ را ببینید)



شکل ۷-۲-۱ : جریان اطلاعات برای تشخیص اولویت M در نوشتن

سپس M_{ii} به روش زیر تعیین میکند که ایا سلول حافظـه مـشترک A(i, i) آدرسی را نگه میداردکه متفاوت است با انهای کـه در A(i - 1), . . . ,A(1, 1). ذخیره شده اند .

هر پردازندهM_{ir}، $1 \le r<i$ ، با خواندن از نشانی های ذخیره شـده در A(r, r)و در A(i, i) شروع میشود و۱ در A(i, r) ذخیره می کند اگر وتنهـا اگـر A(r, r) = A(i, i). پردازنده$M_{i1}, . . . ,M_{i\lceil (i-1)/2 \rceil}$ به طور موازی برای تعیین اینکه آیا مقدار ۱ در هر سلول حافظه مشترک A(i, i - 1), . . . ,A(i, 1) ظاهر میشود ، در O(log i) مرحله ،بکار گرفته می شود. در هـر مرحلـه تعداد پردازنده های "فعال" و تعداد سلول هـای حافظـه مـشترک بـه نـصف کاهش پیدا میکند.در نتیجه هر مرحله مفروض هر پردازنده فعال M_{ir} مقـدار ۱ را در A(i, r) دخیره می کند

اگر وفقط اگر یا A(i, 2r - 1)A(i, 2r) یا A(i, 2r) مقدار را نگه دارند .

$M_{i\,i}$ تعیین میکند که A(i, i) ادرسی را نگه دارد که با آنچه در A(1, 1),. .. A(i - 1, i - 1), ذخیره شده ،متفاوت است .با مشخص کردن اینکه A(i, 1) مقداری که از ۱ متفاوت است را نگه میدارد. در چنین موردی، $M_{i\ i}$ مشخص میکند که M_i بالاترین حق تقدم در نوشتن روی سلولانتخاب شده را دارد وگرنه $M_{i\,i}$ مشخص میکند که M_i بالاترین حق تقدم را ندارد .

۷-۳ مدارها

در نوشته ها تعداد زیادی ماشین های موازی درکنار PRAM ها پیشنهاد شده است.اگر چه،برخلاف مدلهای انتزاعی با ماشینهای ترتیبی،هیچ راه اشکاری برای مرتبط کردن مدلهای انتزاعی مختلف با ماشین های موازی وجود ندارد.بنابرین کوچکترین تقسیم کننده مشترک این چنین مدلهای نمایش سخت افزاریشان است،به نظر میرسد که انتخاب طبیعی برای تحلیل مدل ها برای منابعی که نیاز دارند می باشند.

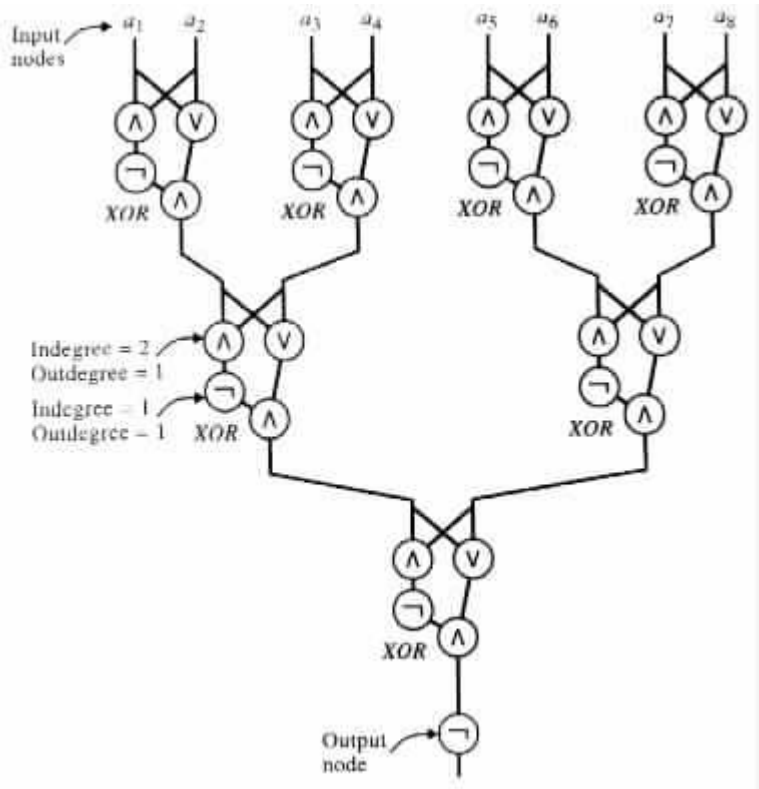
در اینجا نمایشهای در نظر گرفته شده ،به صورت گراف های غیر جهت دار بدون دور هستند که مدارهای بولی ترکیبی یا برای سادگی مدارها نامیده می شوند.هر گره در مدار دارای یک درجه ورودی (یکindegree)(تعداد لبهای جهت داری که به سمت یک گره اشاره می کند)کمتر یا مساوی ۲ و یک درجه خروجی (یکoutdegree)(تعداد ر رئوس جهت داری که از گره خارج میشود)با مقدار نامحدود است.هر گره با درجه ورودی 0 ، با هر یک از دوتا نام متغیرها که با ثابت 0 یا ثابت 1 است نامگذاری میشود.هر گره از **درجه ورودی** ۱ ،با توابع بولی نامگذاری میشود .هر گره از **درجه ورودی** ۲ ، با هر یک از دوتا تابع بولی ∨ یا ∧ نامگذاری می شود.

هر گره از درجه ورودی بزرگتر از ۰ یک گیت نامیده میشود.یک گیت ،گیت *NOT*

گفته مشود اگر با ¬ نامگذاری شود،ویک گیت *AND* است اگر با ∧ نامگذاری شود،ویک گیت OR است اگر با ∨ نامگذاری شود.گره های نامگذاری شده با نام متغیرها را گره های ورودی می نامند.گره های با درجه خروجی ۰ ، گرههای خروجی نامیده میشود.یک گره نامگذاری شده با۰ یک گره ثابت ۰ نامگذاری میشودویک گره با۱ یک گره ثابت ۱نامگذاری میشود.

یک مدار C که nگره ورودی mگره خروجی دارد تابع $f: \{0, 1\}^n \rightarrow \{01\}^m$ را از راه اشکاری محاسبه می کند.

مثال ۷-۳-۱ مداردر شکل۷-۳-۱ را در نظر بگیرید



شکل ۷-۳-۱ شبکه زوجیت(بیت توازن)

7×4+8+1=37گره دارد که ۸ گره ورودی و یک گره خروجی است.

هر مدار تابع توازن را برای مقادیر n=8 ورودی محاسبه میکند. مدار، خروجی ۰ را برای موردی که a_1 ـa_8 تعداد فردی از۱ ها را دارد فراهم میکند.ومدار ،خروجی ۱ را برای موردی که a_1 ـa_8 تعداد زوجی از ۱ها را دارد فراهم میکند..

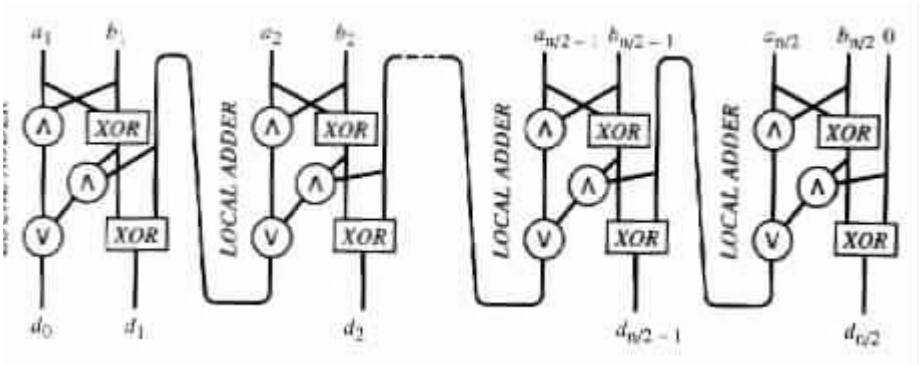
راهکار مدارها به دو روش مشاهده شده در زیر تکیه میکند.

a.تابع توازن مقدارش را تغییر نمیدهد زمانی که یک ۰ از رشته ورودی خذف شود.

b. .تابع توازن مقدارش را تغییر نمیدهد اگر که یک جفت ۱ با یک ۰ از رشته ورودی تعویض شود.

هر گروه XOR از گیت ها ۱ را به خروجی می دهند اگر مقادیر ورودی برابر نباشد.و ۰ را به خروجی می دهند اگر مقادیر ورودی برابر باشد. هر سطح XOR ها از مدار سایز ورودی مفروض را به نصف کاهش می دهد.

مثال ۷-۳-۲ شکل۷-۳-۲را در نظر بگیرید



شکل۷-۳-۲ جمع کننده

یک جمع کننده است که جمع a, a_1 ـ$a_{n/2}$, d_0 ـ$d_{n/2}$. d = a + b و b_1 ـ$b_{n/2}$را با نمایشهای باینری از d,a,b محاسبه می کند. , هر LOCAL ADDER(جمع کننده عمومی) در مدارها یک ورودی دارد که شامل ۲ بیت متناظر a,b است،بعلاوه یک

رقم نقلی از جمع کننده عمومی قبلی. خروجی هر جمع کننده عمومی یک بیت متناظر در d است، بعلاوه یک رقم نقلی جدید که به جمع کننده بعدی منتقل میشود .

سایز مدار تعداد گیت های درآن است.عمق یک مدار تعداد گیت های در طولانی ترین مسیر از گره ورودی به گره خروجی است.

مثال ۷-۳-۳ در مدار شکل۷-۳-۱ هر XORسایز ٤ و عمق ۳ دارد. همه مدار سایز ۲۹ و عمق ۱۰ دارد.

در مدار شکل ۷-۳-۲ هر جمع کننده عمومی سایز ۱۱ وعمق٦ دارد.همه مدار سایز$11n/2$و عمق $5+2(n/2 -2)+3 = n+4$ دارد.

خانواده ی مدار ها

$C = (c_0, c_1, c_2,\ldots)$, خانواده ی از مدارها گفته می شود اگر c_n یک مدار با n گره ورودی برای هر $n \geq 0$ باشد.یک خانواده از مدارها گفته میشود پیچیدگی سایز $Z(n)$ دارد اگر$Z(n) \geq$ (size of c_n) برای همه $n >= 0$ باشد. خانواده از مدارها گفته میشود پیچیدگی عمق $D(n)$دارد اگر(عمق c_n ها)$\geq D(n)$ (برای همه $n >= 0$ باشد.

یک خانواده $C = (c_0, c_1, c_2,\ldots)$ از مدارها تابع مفروض $f: \{0, 1\}^* \rightarrow \{0, 1\}$ را محاسبه می کنند اگر به ازای $n\geq 0$ مدار c_nتابع $f_n: \{0, 1\}^n \rightarrow \{0, 1\}^k$ را برای برخی $k\geq 0$ که به n بستگی دارد ، را محاسبه کند.

f_n تابعی فرض شده است که واجد شرایط $f_n(x) = f(x)$ برای هرxدر $\{۰،۱\}^n$ است.

یک تابع f از پیچیدگی اندازه $Z(n)$ گفته می شود اگر با خانواده ای از مدار ها با پیچیدگی اندازه $Z(n)$ قابل محاسبه باشد. تابع f از پیچیدگی عمق $D(n)$ گفته می شود اگر با خانواده ی از مدارها با پیچیدگی عمق $D(n)$ قابل محاسبه باشد.

یک خانواده $C = (c_0, c_1, c_2, \ldots)$ از مدار ها زبان L در $\{0,1\}^*$ راتصمیم می گیرد اگرتابع مشخصه ی L با C قابل محاسبه باشد(Fیک تابع مشخصه ی از L است اگـر ۱ =$f(x)$ برای هر x در L ،و $f(x) = 0$ برای هر x در L نباشد) پیچیدگی های عمق و اندازه یک زبان پیچیدگی های عمق واندازه تابع مشخصه آن است.

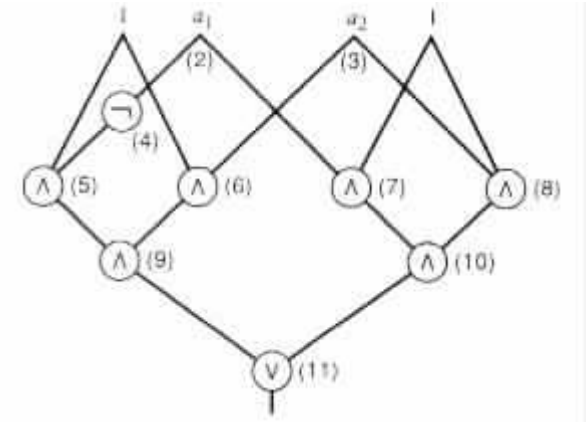
مثال ۷-۳-٤ زبان {تعداد ۱ های زوج دارد a_1 ـa_n| $a_1,\ldots,a_n$ are in $\{0, 1\}$, a_1 ـa_n } با خانواده ی از مدار های مشابه شکل ۷-۳-۱ قابل تصمیم گیری است است.زبان پیچیدگی عمق$O(\log n)$ دارد وپیچیدگی اندازه $O(n/2 + n/4 + \text{ـ} + 1)$ $= O(n)$.

نمایش مدارها

در ادامه، ما فرض خواهیم کردهر مدار c نمایشی به شکل زیر دارد. عدد ۰ را بـا هـر گره ثابت ۰در c مربوط کنید، عدد۱ با هر گره ثابت۱در c مربوط کنیـد ، اعـداد ۲,. . . $n + 1$, با n گره های ورودی با c مربوط کنید. اعداد متوالی با شروع در $n + 2$،را با هر یک از گیت های c ربـط دهیـد.سـپس نمایـشی از c بـا شـروع از شـکل $E(u_1)$ $\ldots E(u_m)F(V_1)\ldots F(v_k)$ است.

$u_1,\ldots,u_m$ گیتهای از cهستندو $v_1,\ldots,v_k$ گرههـی خروجـی هـستند. $E(u)$برابـر با(g, t, g_L, g_R),است که g عددی است که به گیـت u نـسبت داده شـده اسـت،t از نوع uدر $\{\wedge, \vee, \neg\}$است و g_Lو g_R اعدادی هستند که به پردازنده های بی واسـطه u نسبت داده شده اند . خصوصا، $g_L = g_R$زمانی که $t = \neg$ باشد. $F(v)$ برابـر بـا(g) است زمانی که g عدد منسوب شده به گیت v باشد.

مثال۷-۵-۳ شکل ۷-۳-۳ را در نظر بگیرید



شکل ۷-۳-۳ یک مدار با گره های شمارش شده

مداری که گره های آن شمارش شده اند ایجاد کرده است . مدار دارای نمایش زیر است :

$$(4,\neg,2,2)(5,\wedge,1,4)(6,\wedge,1,3)(7,\wedge,2,1)(8,\wedge 3,1)$$
$$(9,\wedge,5,6)(10,\wedge 7,8)(11,\vee,9,10)(11)$$

۷-٤ خانواده مدارهای یکنواخت

مدارهای "حستجوی جدولی "

خانواده مدار ها معرفی شدند تابه توصیف منابعی که مسائل برای ماشین های موازی نیاز دارند ، کمک کنند.

قضیه زیر دلالت میکند که خانواده ها، در فرم کلی خود ، نمی توانند در راه چنین هدفی مفید باشند ،زیرا

انها توانایی تشخیص زبان هایی را دارند که بازگشتی شمارش پذیر نیستند.

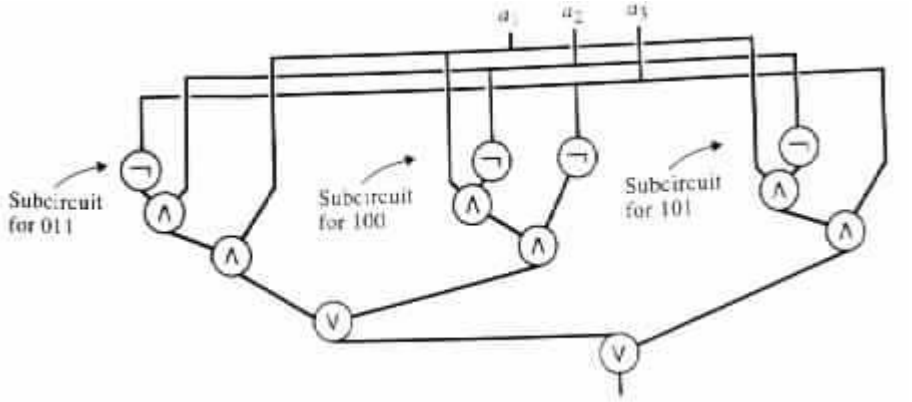
قضیه ۷-٤-۱ هر زبان L در $*\{0, 1\}$ با یک خانواده ی از مدار ها قابل تصمیم گیری است

اثبات : هر زبان L در $\{0,1\}^*$ ،و هر n عدد طبیعی را در نظر بگیرید. و L_n را نماینده مجموعه $L \cap \{۰,۱\}^n$ بگیرید که ، L_n مجموعه ی از همه رشته های باینری با طول nدر L است .

برای هر رشته مفروض w در L_n، a زیر مدار c_w با n گره ورودی می تواند ساخته شود که یک ورودی مفروض را می پذیرد اگر وتنها اگر ورودی برابر با w باشد . زبان L_n متناهی است . در نتیجه ، زیر مدار های c_w که متناظر با رشته های w در L_n است می توانند ادغام شوند ،به این صورت که گره های ورودی را به اشتراک بگذارند و خروجی یشان را OR کنند . مدار به دست آمده c_n ،عضویت در L_n را با تکنیک جستجوی جدولی تعیین می کند.

در نتیجه ، Lبا خانواده $(c_0, c_1, c_2, \ldots)$ از مدار ها قابل تصمیم گیری است

مثال ۷-٤-۱ مدار c_3 در شکل ۷-٤-۱ را در نظر بگیرید



شکل ۷-٤-۱یک "جستجوی جدولی " مدار برای زبان $\{L_3 = \{011, 100, 101\}$ زبان $L_3 = \{011, 100, 101\}$ با شیوه جستجوی جدولی تصمیم پذیر است .برای هر رشته w در L_3، مدار دارای یک زیر مدارمتناظر Wاست که فقط عضویت در رشته را تصمیم می گیرد.

.

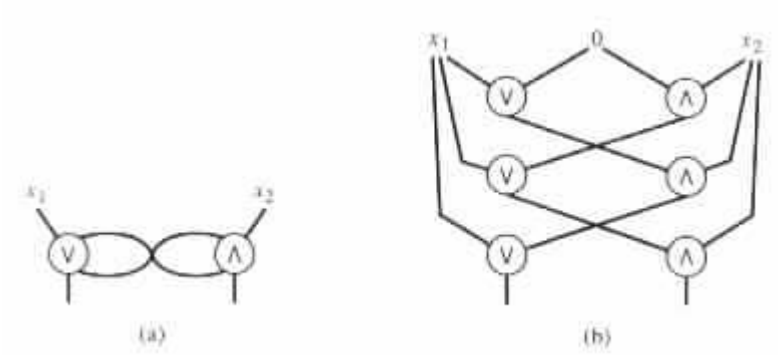
خانواده ی مدارها در جستجوی جدولی در اثبات قضیه۷-٤-۱ پیچیدگی اندازه $2^{O(n)}$

وپیچیدگی عمق $O(n)$ دارد. این خانوادها پیچیدگی تصمیم گیری زبان را منعکس نمی کنند، زیرا دانش این که کدام رشته در زبان داده شده است و کدام نیست ، برای آنها مفروض است . از رشته پیچیدگی لازم برای تصمیم گیری زبان ها به پیچیدگی ساخت خانواده ی مدار های متناظر انتقال می یابد.

مدارهای" قطعات نپیچیده شده "

یک مدار می تواند توصیفی از محاسبات توقف با ماشین موازی را بـه وسـیله کنـار گذاشتن بخشی از قطعات در که در محاسبات در گیر می شود بـه دسـت آورد. در طول کنار گذاشتن قطعات، سیکل ها می توانند به وسیله ی قطعات نپیچیده شده جلو گیری شوند . عمق این چنین مـداری سنجـشی بـرای زمـان را فـراهم میکنـد کـه در محاسبات نیاز دارند،و اندازه مدار یک حد بالا از فضای مورد نیاز محاسبات را فـراهم می کند.

مثال ۷-٤-۲ مدار شکل۷-٤-۲ b) را در نظر بگیرید



شکل ۷-٤-۲ (a)"قطعه"(b) "قطعات پیچیده نشده" متناظر

تابعی را محاسبه می کند که قطعه در شکل ۷-٤-۲ (a آن را در سه واحد زمانی محاسبه می کند . فرض شده که در ابتدا هر ورودی به یک گیت یکی از دو مقدار

ورودی یا ثابت ۰ است

در راهی مشابه، می توان یک مدار c_n که متناظر با همه محاسبا ت توقف از M روی نمونه های از طول n و 0^n است را بدست آورد. (خروجی برای ورودی با طول مفروض n ، در انتهای آن رشته به شکل $0\ldots 10$ اضافه می شود تا همه آنها دارای طول یکسانی باشند .) در نتیجه،این روش ،خانواده ی $C = (c_0, c_1, c_2,\ldots)$از مدار ها برای هر ماشین موازی M نشان می دهد که روی همه ورودی ها متوقف می شود. علاوه بر این، خانواده ی مدار ها پیچیدگی محاسبات موازی را بدرستی بر می گرداند و می تواند بطور موثر از یک چنین ماشین موازی M بدست آید

خانواده مدارهای یکنواخت

طبق بحث قبلی ، از هر ماشین موازی M که روی همه ورودی ها توقف می کند ، یک سا زنده مدار می تواند برای محاسبه $\{ (0^n, c_n) \mid n \geq 0 \}$ بدست آید کـه $C = (c_0, c_1, c_2,\ldots)$خانواده ی از مدار ها است که تابعی یکسان ماننـد M محاسـبه مـی کنـد. سازنده مدار ها می تواند خانواده ای از جستجوی جدولی ، قطعات نا پیچیده یا دیگـر نوع از مدار ها را فراهم آورد.

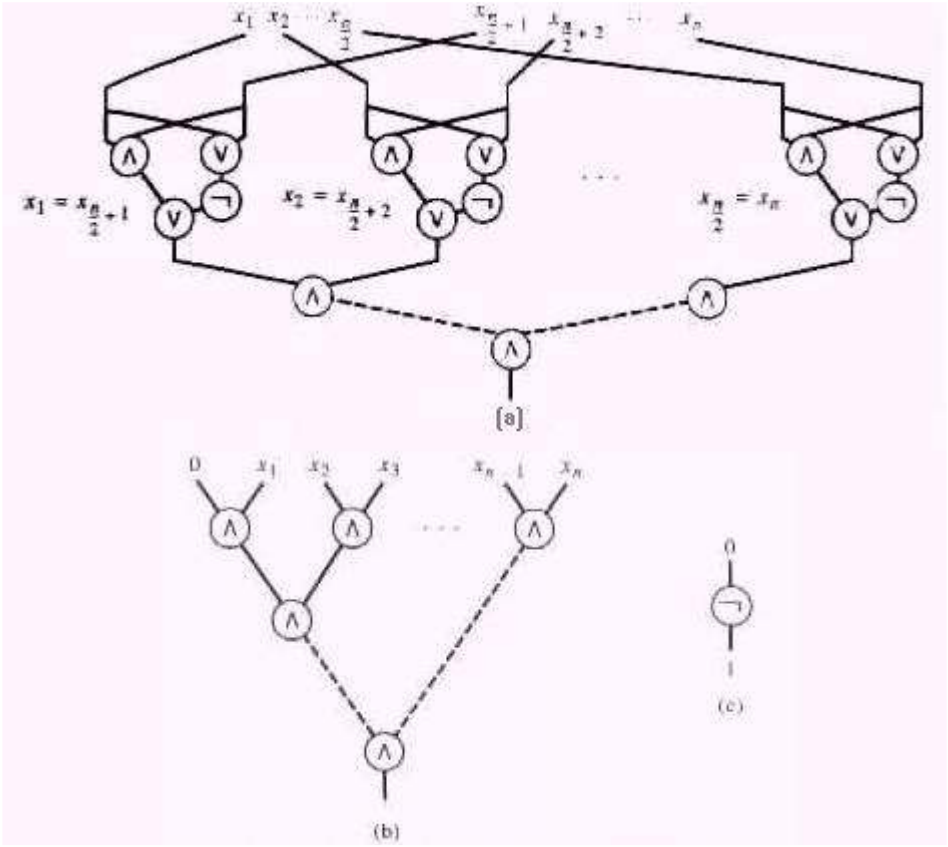
توجه این قسمت روی سازنده های مداری است کـه پیچیـدگی ماشـین مـوازی داده شده را در خانواده ی مدار هایی را که می سازند، حفظ مـی کننـد. کـه چنـین سـازنده هایی اجازه انتقال پیچیدگی از خانواده های مدار های ساخته شده بـه سـازنده هـا را نمی دهد. علاوه بر آن، آنها همچنین اجازه افزایش غیر واقعی در پیچیدگی خانواده ی مدار های ساخته شده را نمی دهد. سازنده های مدار ها بـا چنـین توصـیفاتی سـازنده های مدار یکنواخت گفته می شوند . یک خـانواده $C = (c_0, c_1, c_2,\ldots)$ یکنواخـت است اگر سازنده مدار های یکنواخت بتواند تابع $\{ (1^n, c_n) \mid n \geq 0 \}$ را محاسـبه کند.

خصوصیات زیادی برای سازندهای مدار های یکنواخت ارایه شده است. توصیفاتی در اینجا استفاده شده است که در بسیاری از موارد پذیرفته شده اند ، و شرایط آنها بر اساس کلاس مبدل های تورینگ قطعی تعریف می شود .

تعریف:یک مبدل تورینگ یک سازنده مدارهای یکنواخت گفته میشود اگر آن مبدل تورینگ قطعی با محدودیت مکانی $O(\log Z(n))$ باشد ، که $\{ (1^n,c_n) \mid n \geq 0 \}$ را محاسبه می کند، که$C = (c_0, c_1, c_2,\ldots)$ یک خانواده از مدار ها با پیچیدگی اندازه $Z(n)$ است .یک خا نواده $C = (c_0, c_1, c_2,\ldots)$ از مدار ها با پیچیدگی سایز $Z(n)$ خانواده یکنواخت از مدار ها گفته می شود اگر یک مبدل تورینگ قطعی با محدودیت مکانی $O(\log Z(n))$، بتواند $\{ (1^n,c_n) \mid n \geq 0 \}$را محاسبه کند .

خصوصیات خانواده مدار های یکنواخت با روش قطعات نپیچیده شده تحریک شده است.با چنین روشی سازنده مدارها به $O(\log H(n) + \log T(n)) = O(\log (H(n)T(n)))$ فضا نیاز دارد، اگرماشین موازی پیچیدگی سایز $H(n)$و پیچیدگی زمان $T(n)$ داشته باشد.$O(\log H(n))$ فضا برای ردیابی سراسر قطعات استفاده می شود،و $O(\log T(n))$فضا برای ردیابی مابین زمان استفاده می شود. $H(n)T(n)$ به صورت ترتیب مشابهی از مقادیر با اندازه $Z(n)$در مدارهاست.

مثال ۷-٤-۳ زبان $L=\{ uu \mid u \text{ is in } \{0, 1\}^*\}$ را در نظر بگیرید.بگذارید $L_n = L \cap \{0,1\}^*$,برای$n\geq0$ که، L_n مجموعه ی از همه رشته های باینری با طول n در L را مشخص میکند. زبان L_n بوسیله c_n در شکل۷-٤-۳ (a) تصمیم گرفته می شود.



شکل ۷-٤-۳ یک مدار c_n است که، بر حسب مورد چک میکند ایا ورودی با طول n شکلی از uu است (a) $n \neq 0$ و . (b) n زوج است فرد است. (c) $n = 0$.

خانواده $(c_0, c_1, c_2, \ldots)$ از مدار ها با پیچیدگی عمق $D(n) = O(\log n)$ و پیچیدگی اندازه $Z(n) = O(n/2 + n/4 + \ldots + 1) = O(n)$ می باشد . خانواده یکنواخت است زیرا تابع $\{ (1^n, c_n) \mid n \geq 0 \}$ با یک مبدل تورینگ قطعی مکان محدود $Z(n) = O(\log n)$ ، محاسبه پذیر است .

نظریه زیر برای محاسبات موازی بر اساس خانواده ی مدارهای یکنواخت بیان شده ا ست. طبق آنچه در نظریه قبل برای محاسبات ترتیبی و احتمالی آمد ، فقط شواهد محکم می توانند درستی نظریه را نشان دهند .

نظریه محاسبات موازی: یک تابع میتواند با ماشین موازی با پیچیدگی اندازه $H(n)$ و پیچیدگی زمان $T(n)$ (بطور خودکار) محاسبه شود فقط اگر یک خانواده از مدارهای یکنواخت از پیچیدگی اندازه $p(H(n)T(n))$ و پیچیدگی عمق $p(T(n))$، برای بعضی چند جمله ای $p(.)$ داشته باشد.

۷-۵ خانواده مدار های یکنواخت و محاسبات ترتیبی

اندازه مدار ها بزرگترین منبع برای محاسبات موازی است و همان طور که زمان بزرگترین منبع برای محاسبات ترتیبی است. قضیه زیر نشان می دهد که هر دو نوع از منابع به صورت چند جمله ا ی به هم مرتبط اند

تعریف : در ادامه $DTIME\_F(T(n))$ کلاسی از توابع محاسبه پذیر به وسیله مبدل های تورینگ قطعی با محدودیت زمانی $O(T(n))$، را مشخص می کند . کلاسی از توابع با پیچیدگی اندازه $SIZE\_F(Z(n))$ با $O(Z(n))$ مشخص می شوند. کلاسی از زبان هاکه توابع مشخصه آنهادر $SIZE\_F(Z(n))$ قرار دارند ،با $SIZE(Z(n))$مشخص می شوند . $U\_SIZE\_F(Z(n))$ کلاس توابع محاسبه پذیر با خانواده مدارهای یکنواخت *با اندازه پیچیدگی* $O(Z(n))$ را مشخص می کند. کلاسی از زبان هایی که توابع مشخصه آنها در $U\_SIZE\_F(Z(n))$ است با $U\_SIZE(Z(n))$ مشخص خواهند شد. $U\_DEPTH\_F(D(n))$ کلاسی از توابع محاسبه پذیر با خانواده مدار های یکنواخت از پیچیدگی عمق $O(D(n))$ را مشخص خواهد کرد ، کلاسی از زبان هایی که توابع مشخصه آنها در $U\_DEPTH\_F(D(n))$ هستند با $U\_DEPTH(D(n))$ مشخص خواهد شد. $U\_SIZE\_DEPTH(F(Z(n), D(n$ کلاسی از توابعی را مشخص خواهد کرد که با خانواده مدارهای یکنواخت با پیچیدگی اندازه $Z(n)$ و پیچیدگی عمق $D(n)$ به طور همزمان محاسبه پذیرند .

قضیه ۷-۵-۱ اگر $\log T(n)$ کاملاً قابل ساخت مکانی باشد، آنگاه

$$\bigcup_{d\geq 0} DTIME\_F(T^d(n)) = \bigcup_{d\geq 0} U\_SIZE\_F(T^d(n))$$

برهان قضیه دلالت بر دو لم زیر می کند

e

از زمان ترتیبی با اندازه مدار

اثبات لم اول شامل قطعات نپیچیدگی مبدل های تورینگ ماشین است

لم ۷-۵-۱ اگر $\log T(n)$ کاملا قابل ساخت مکانی باشد، آنگاه

$$DTIME\_F(T(n)) \subseteq U\_SIZE\_F(T^2(n)).$$

اثبات : هر مبدل تورینگ قطعی با محدودیت زمانی $T(n)$ به صورت $M=<Q,\Sigma,\Gamma,\Delta,\delta,q_0,B,F>$ را در نظر بگیرید . که $\log T(n)$ کاملاً قابل ساخت مکانی است. بدون از بین رفتن کلیت فزض می شود $\Sigma=\{۰, ۱\}$ و m تعداد نوار های کاری کمکی M را مشخص کند .

یک نسخه اصلاح شده از M

فرض شده که Δ شامل سمبل های a و bنیست. M از راه زیر اصلاح می شود

a. هر قانون انتقال را که خروجی را با قانون a فراهم نمی کند طوری تغییر دهید که خروجی b را فراهم کند.

b . قوانین انتقالی را که از مراحل پذیرش ،سرچشمه می گیرند حذف کنیدو حالت های پذیرش را به صورت حالت های غیر پذیرش درآورید ،یک حالت غیر پذیرش جدید اضافه کنید و کردن قوانین انتقال جدیدی اضافه کنید که M با به وسیله آنها با نوشتن سمبل a از حالت های پذیرش قدیمی به حالت های جدید برود. حالت جدید را حالت a می نامند .

c.برای هر حالتq , سمبل ورودی c, و سمبل های نوار کاری کمکی b_m, . . . b_1, که هر کدام از ($q, c, b_1, . . . , b_m$) تعریف نشده است،قانون انتقال($q, c, b_1, . . .$, b_m, $q, 0, b_1, 0, . . . , b_m, 0, \alpha$) را به δ اضافه کنید. فرض می شود α با a برابر باشد اگر q یک حالت a باشد و α با b برابر باشد اگر q یک حالت a نباشد.

M تغییر یافته یک مبدل تورینگ قطعی است ، که روی هر ورودی محاسبه ای با تعداد حرکات نا محدود دارد. بر روی یک ورودی که Mاصلی i حرکت دارد، M تغییر یافته در

i + 1مین حرکت وارد حلقه بی نهایت می شود. .درهر حرکت M تغییر یافته یک سمبل را بر نوار خروجی می نویسد. خروجی M تغییر یافته در i+1مین، i + 2, مین... حرکت a می باشد اگر وفقط اگر ورودی در M اصلی پذیرفته شده باشد، علاوه بر آن، خروجی M اصلی می تواند از رشته ای که M تغییر یافته بر نوار خروجی می نویسد ، بوسیله حذف همه سمبل های a و همه سمبل های b بدست آید.

یک مدار c_n برای شبیه سازی M

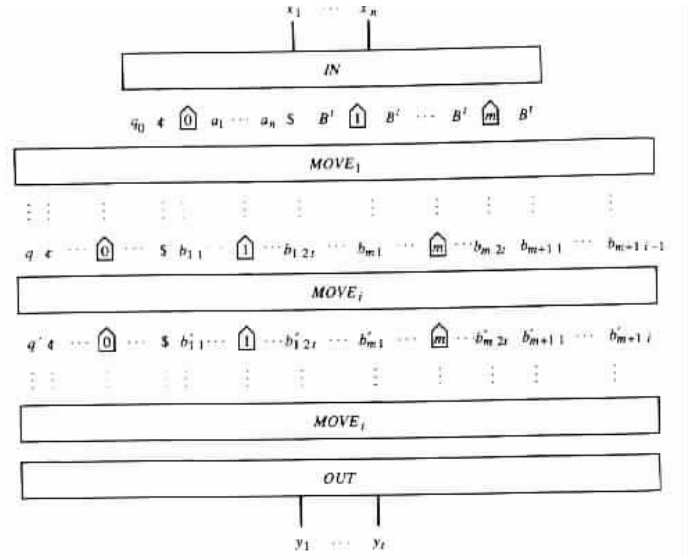
مدار c_nاز شکل زیر با Mاصلی روی ورودی با طول n و بوسبله شبیه سازی اولین t $= 2^{\lceil \log (T(n)+1) \rceil}$ حرکت هااز M تغییر یافته بر روی ورودی مفروض شبیه سازی می شود .

شبیه سازی دقیقاً $t = 2^{\lceil \log (T(n)+1) \rceil}$ حرکت از M به c_n اجازه می دهد تا خروجی با همان طول t برای همه ورودی های با طول n تولید کند . چنین یکنواختی در در طول خروجی ها لازم است زیرا ثبات و استحکام مدار ها در طول خروجی هایشان است .

انتخاب $t = 2^{\lceil \log (T(n)+1) \rceil}$به جای T(n) + 1برای تعداد حرکات M، انجام شده است تا اجازه دهد که مقدار فقط به وسیله نشانه گذاری فضای با اندازه O(log T(n)) محاسبه شود.

c_n تعدادی نمایش باینری قطعی برای مجموعه {a, b, ?, $, - 1, 0, + 1, Δ, ∪} Q فرض می کند. عناصر مجموعه میتوانند با رشته های بـاینری بـا طول مساوی k نمایش داده شود. $q_0, \ldots, q_m$ با سمبل های جدید متناظر بـا هـدهای M فرض شده اند.

c_n شامل t + 2 زیـر مـدار اسـت کـه بـه آنهـا IN و $MOVE_1$ و ... و $MOVE_t$ و OUT می گویند .(شکل۷-۵-۱ را ببینید)



شکل ۷-۵-۱ یک مدارc_n که توابع محاسبه پذیر بوسیله مبدل تورینگ قطعی M در نمونه های با طول n رامحاسبه میکند

IN یک زیرمدار است که پیکربندی اولیه M را روی ورودی $a_n, \ldots, a_1$ نتیجه می

دهد.

$$(\$q_0a_1\cdots a_n\$, B^tq_0B^t, \ldots, B^tq_0B^t, \epsilon)$$

lNامقادیر $a_n, \ldots a_1$را از گره هـای ورودی $x_n, \ldots x_1$ ، مقـادیر ثابـت گـره هـای صفر و مقادیر ثابت گره های یک را برای نمایشی از پیکربندی مطلوب استفاده می کند .

$$(\$a_1\cdots q'\cdots a_n\$, b'_{11}\cdots q'\cdots b'_{12t}, \ldots, b'_{m1}\cdots q'\cdots b'_{m2t}, b'_{m+11}\cdots b'_{m+1t})$$

زیرمدار $MOVE_i$ ، $1 \le i \le t$، i امین پیکربندی را *نتیجه می دهد.*

$$(\$a_1\cdots q'\cdots a_n\$, b'_{11}\cdots q'\cdots b'_{12t}, \ldots, b'_{m1}\cdots q'\cdots b'_{m2t}, b'_{m+11}\cdots b'_{m+1t})$$

از M توسط $i-1$ مین پیکربندی

$$(\$a_1\cdots q\cdots a_n\$, b_{11}\cdots q\cdots b_{12t}, \ldots, b_{m1}\cdots q\cdots b_{m2t}, b_{m+11}\cdots b_{m+1t-1})$$

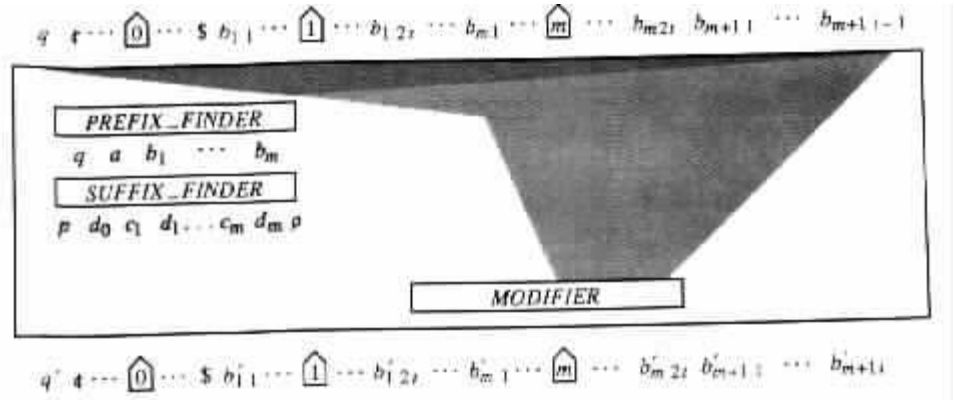
در M.

OUT زیرمداری اسـت کـه (رمـز گـذاری) خروجـی $b_t\ldots b_1$ را کـه M در t امـین پیکربندی اش دارد ، استخراج میکند. OUT این کار را با حذف سـمبل هـایی کـه در $\{a, b\}\cup\Delta$ نیست انجام می دهد برای مثال با استفاده از گیت های AND.

مدار فرعی MOVE i

$MOVE_i$ مولفه های PREFIX _FINDERو SUFFIX _FINDE برای تعیین قانون انتقال $(q,a, b_1, \ldots, b_m, p, d_0, c_1, d_1, \ldots, c_m, d_m, \epsilon)$ استفاده می

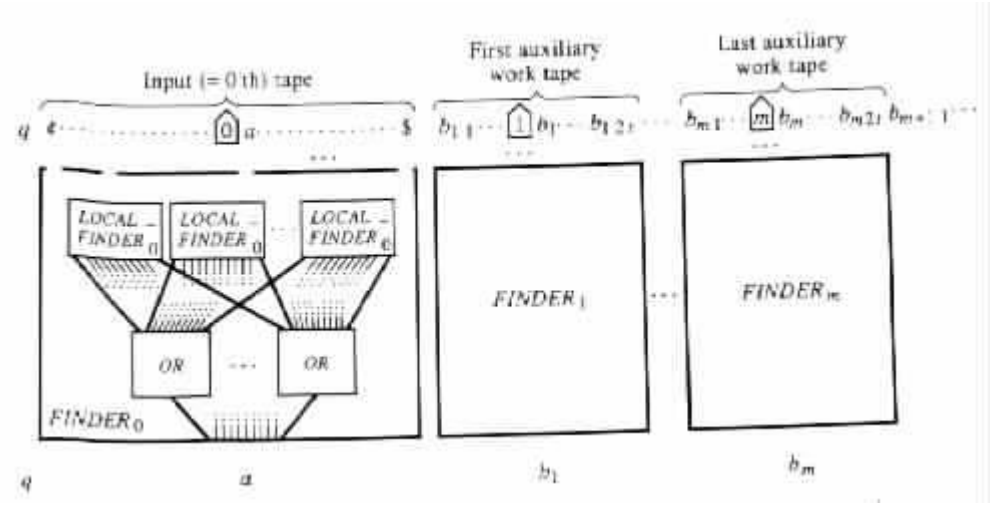
کند که M در حرکت iم استفاده می کند.(شکل۷-۵-۲ را ببینید)



شکل ۷-۵-۲ زیرمدار $MOVE_i$ برای شبیه سازی یک انتقال مبدل تورینک قطعی بین دو پیکربندی.

PREFIX _FINDER پیشوند$(q, a, b_1, \ldots, b_m)$ از قانون انتقال از $i-1$ مین پیکربندی M تعیین می کند. SUFFIX _FINDER پسوند$(p, d_0, c_1, d_1, \ldots, c_m, d_m, \beta)$ از قانون انتقال برای$(q, a, b_1, \ldots, b_m)$ تعیین می کند. MOVE $_i$ مولفه MODIFIER را برای انجام تغییرات لازم روی پیکربندی $i-1$ ام M استفاده می کند .

های PREFIX _FINDER دارای مولفه $FINDER_i$ متناظر با هر نوار غیرخروجی M می باشد .(شکل ۷-۵-۳ را ببینید)



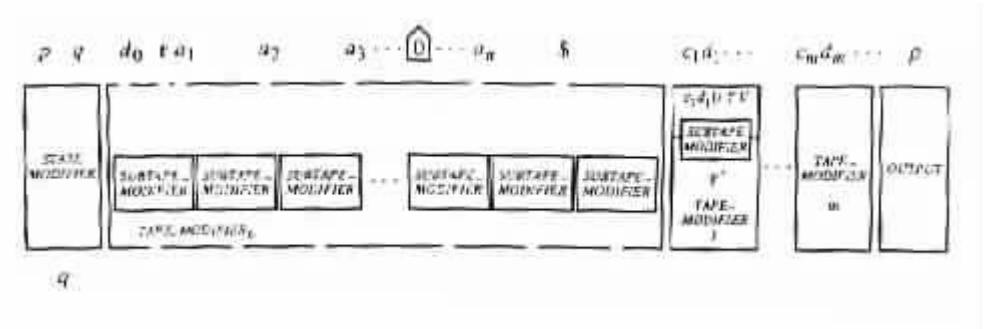
شکل ۷-۵-۳ یک زیرمدار PREFIX _FINDERبرای تعیین یک قانون انتقال مبدل تورینگ است

$FINDER_i$ سمبلی را تعیین میکند که تحت هد i امین نوار M است. $FINDER_i$ زیر مدار LOCAL _FINDER را برای هر جفت از سمبل های متوالی در بخشی از پیکربندی که متناظر با i مین نوار M است ، بکار می گیرد .

. $LOCAL\ \_FINDER_i$ (نمایش) سمبل a را به خروجی می دهد اگر ورودی آن متناظر با جفتی به صورت aباشد وگرنه، زیرمدار $LOCAL\ \_FINDER_i$ فقط ۰ را به خروجی می دهد . خروجی هر $LOCAL\ _FINDER_i$با مدارجستجوی جدولی تعیین می شود. خروجی های همه $LOCAL\ \_FINDER_i$ ها OR شده تا خروجی مطلوب $FINDER_i$ بدست آید.

SUFFIX _FINDER روی ورودی$(q, a, b_1, \ldots, b_m)$ یک روش جستجوی جدولی برای پیدا کردن$(p,d_0,c_1,d_1,\ldots,c_m,d_m,\rho)$ به کار می گیرد .

MODIFIER شامل یک مولفه $TAPE\_MODIFIER_i$ برای هر نوار غیر خروجی i از مبدل تورینگ M که $1 \leq i \leq m$ است . (ببینید شکل۷-۵-٤)



شکل ۷-۵-٤ یک زیرمدار MODIFIER برای اصلاح کردن یک شکل مبدل تورینگ.

$TAPE\_MODIFIER_i$ شامل یک زیر مدار SUBTAPE _MODIFIER برای هرمکان در پیکربندی ساخته شده از مبل تورینگ M است .

یک SUBTAPE _MODIFIER که متناظر با مکان j است سه سمبل U, Y و V را به عنوان ورودی در مکانهای $j-1$ و j و $j+1$ در پیکربندی M دریافت می کند که تغییر کرده اند .(استثنا وقتی اتفاق می افتدکه مکان jام مکان مرزی است .در چنین موردی SUBTAPE _MODIFIER فقط دو مقدار ورودی دریافت میکند) علاوه بر آن SUBTAPE _MODIFIER به عنوان ورودی تغییرات (c_i و d_i)را می گیرد که در نوار jام M ساخته می شود. SUBTAPE _MODIFIER سمبل Y' برای مکان jام در پیکربندی ساخته شده M به خروجی می دهد .

یک سازنده مدار های یکنواخت

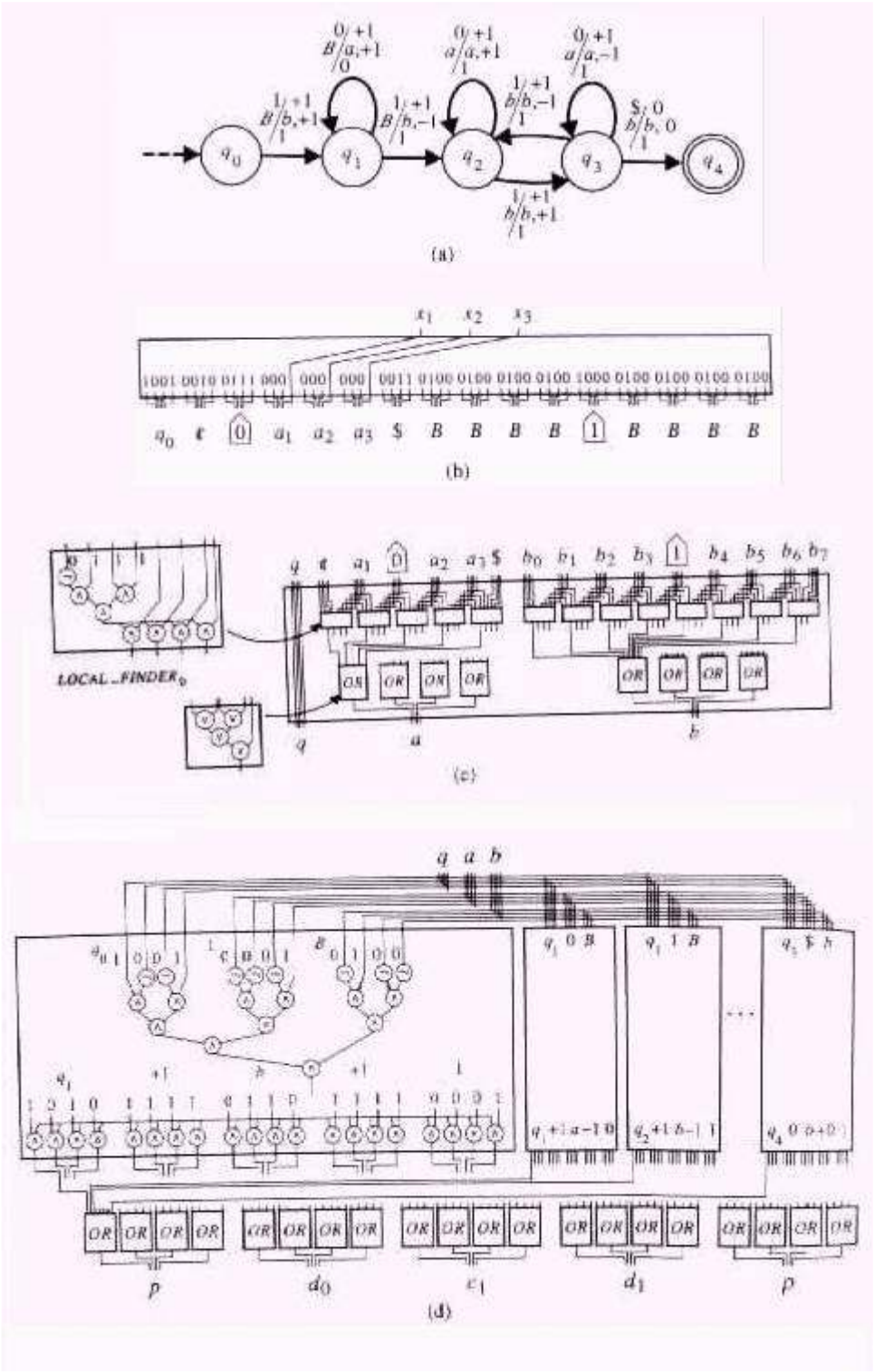
IN سـایز ۰ دارد. هـر $FINDER_i$ شـامل $O(T(n))$ زیرمـدارهای LOCAL $FINDER_i$_ و تعداد ثابتی . است

مدارهای OR می با شد .هر $LOCAL\_FINDER_i$ سایز ثابـت دارد.هـر زیرمـدار ORسایز $O(T(n))$ دارد. بنابرین، PREFIX _FINDER سـایز $O(T(n))$دارد. SUFFIX _ FINDER سـایز ثابـت دارد،و TAPE _MODIFIER سـایز $O(T(n))$ دارد درنتیجه، c_n سایز $O(T^2(n))$ دارد.

یک مبدل تورینگ X با مکان محدود $O(\log T(n))$ ، می تواند برای محاسبه $\{ ^n, \} \mid n \geq 0$ $(c_n۱)$ } به روش brute-force ساخته شود .

مثال ۷-۵-۱ مبدل تورینگ قطعی M با یک نوار کاری کمکـی د ر شـکل۷-۵-۵(ا)را در نظر بگیرید. M پیچیدگی زمان $T(n) = n + 1$ دارد. برای هدف مثال تفـسیر M مانند خودش بدون اصلاح در نظر بگیرید. بااستفاده کردن ازاصطلاحات در اثبـات لم۱,۵,۷_ ,m = 1 ,$\Gamma = \{0, 1, B\}$,$\Sigma = \Delta = \{0, 1\}$,$Q = \{q_0, q_1, \ldots, q_4\}$و, $k = 4$.نمایش باینری زیر را انتخاب کنید

E: E(0) = 0000, E(1) = 0001, E(?) = 0010, E($) = 0011, E(B) = 0100, E(a) = 0101, E(b) = 0110, E(□) = 0111, E(□) = 1000, $E(q_0) = 1001$, $E(q_1) = 1010$, $E(q_2) = 1011$, $E(q_3) = 1100$, $E(q_4) = 1101$, E(-1) = 1110, E(+1) = 1111. Choose n = 3.



شکل ۷-۵-۵ PREFIX _FINDERمدارهای فرعی مشابه SUFFIX (a)مبدل تورینگ . (b) مدار های فر عی مشابه (c).IN _FINDER مدارهای فرعی مشابه(d).

در چنین موردی، $t = 4$. زیرمدارهای INدر شکل مفروض۷-۵-۵ (b)،مدارهای فرعی PREFIX _ FINDER در شکل مفروض۷-۵-۵ (c)،ومدارهای فرعی SUFFIX _FINDERدر شکل مفروض۷-۵-۵ (d)است.

از اندازه مدار ها تا زمان ترتیبی

درلم قبلی با کاربردی کردن موازات برای شبیه سازی محاسبات ترتیبی سرو کار داشتیم.در لم زیر با شبیه سازی کردن محاسبات موازی بوسیله محاسبات ترتیبی سرو کار داریم.

لم ۷-۵-۲ $U\,\mathrm{U\_SIZE\_F}(Z(n)) \subseteq \bigcup_{d\geq 0} \mathrm{DTIME\_F}(Z^d(n))$

اثبات: هر تابع $Z(n)$، وهر خانواده یکنواخت $C = (c_0, c_1, c_2,\ldots)$از مدار ها با پیچیدگی اندازه $Z(n)$ را در نظر بگیرید . بگذارید X مبدل تورینگ قطعی با محدودیت مکانی $O(\log Z(n))$ باشد که تابع $\{ (1^n, c_n) \mid n \geq 0 \}$ را محاسبه می کند .یک مبدل تورینگ قطعی M می تواند همین تابع را مانند C به روش زیر محاسبه کند:

ورودی مفروض $a_n, \ldots a_1$، M، X را به کار می گیرد تا نمایشی از مدار های c_n را تعیین می کند. نمایش می تواند در زمان $2^{O(\log Z(n))} = Z^{O(1)}(n)$ پیدا شود زیرا X از محدودیت مکانی $O(\log Z(n))$ است (ببینید قضیه ۵-۵-۱). علاوه بر این

نمایش طـول $O(Z(n)\log Z(n))$ دارد. زیـرا c_n حـداکثر $Z(n)$ گیـت دارد ،و هـر گیت(g, t, g_L, g_R) یک نمایش از طول $O(\log Z(n))$ دارد .

با داشتن نمایش c_n، مبدل تورینگ M خروجی هر گره c_n را ارزیابی می کند. M این کار را به این روش انجام می دهد که مکررا نمایش c_nرا برای چهـار تـایی(g, t, g_L, g_R)که متناظر با گره های g_L وg_Rاست کهمقادیر خروجی شان قبل از این معلوم است ، پیمایش میکند در صورت پیدا شدن چنین چهار تـایی(g, t, g_L, g_R) ، مبـدل تورینگ M ارزیابی می کند ومقدار خروجی g را نگه می دارد. بعد از حد اکثر $Z(n)$ تکرار ، M مقادیر خروجی همه گره های c_nرا تعیین می کند.

سرانجام ، M تعیین می کند که کدام یک ازگره های c_n گره هـای خروجـی هـستند و مقادیر شان را می نویسد .

توسط قضیه۷-۵-۱ ،زمـان محاسـبات ترتیبـی و انـدازه خـانواده یکنواخـت از مـدارها بصورت چند جمله ای با هم رابطه دارند .

نتیجه ۷-۵-۱ یک مسئله با زمان چند جمله ی قابل حل است اگروفقط اگر توسط خانواده یکنواخت از مدارها با پیچیدگی سایز چند جمله ی قابل حل باشد.

NC وU_FNC , U_NC

محاسبات ترتیبی امکان پذیر در نظر گرفته می شوند فقط اگر حـد زمـانی بـه صـورت چند جمله ا ی باشد. به طور مشابه ،خانواده ای از مدارها امکان پذیر در نظر گرفته می شوند فقط اگر از محدودیت اندازه چند جمله ا ی باشد. . در نتیجه،به نظر می رسد کـه موازی سازی تاثیر زیادی روی مسائلی که در زمان چند جمله ا ی قابل حـل نیـستند ، ندارد. از سویی دیگر، موازی سازی برای مسائلی که در زمان چندجملـه ای قابـل حـل هستند دارای اهمیت مرکزی است زیرا به طور شاخصی سرعت محاسبه را بالا می بـرد

. چنین کلاسی از مسائل هستند که میتوانند توسط خانواده یکنواخت از مدار ها حل شوند که به طور همزمان پیچیدگی اندازه چند جمله ای وپیچیدگی عمق *polylog* (یعنی$O(\log^i n)$ $i \geq 0$)داشته باشند.این کلاس از مسائل با U_FNC مشخص شده است.

زیر کلاس U_FNC،که با محدود کردن پیچیدگی عمق خانواده ای از مدار هابه $O(\log^i n)$ بدست می آید ، U_FNC^i مشخص شده است. زیر کلاس از مسائل تصمیم گیری U_FNC،با U_NCمشخص شده است.زیر کلاس از مسائل تصمیم گیری, U_FNC^iبا U_NC^i مشخص شده است .

FNC نماینده کلاسی از مسائل قابل حل توسط خانواده ای از مدار ها (لزوما یکنواخت نیست) است که به طور همزمان پیچیدگی اندازه چند جمله ای و پیچیدگی عمق polylog دارند .

وزیرکلاس از مسائل تصمیم گیری FNC،با NC مشخص شده است. زیرکلاسی از FNC که با محدود کردن خانواده ای از مدار ها با پیچیدگی عمق $O(\log^i n)$ بدست آمده،با FNC^i مشخص شده است. NC^i کلاس مسائل تصمیم گیری در FNC^i مشخص شده است .

برای خانواده ای از مدار های غیر یکنواخت قیاس قضیه زیر مطرح میشود .

| قضیه ۷-۵-۲ NC^1 *شامل مسائل غیر قابل تصمیم گیری است .*

. اثبات :هر زبان یکانی L روی الفبای $\{1\}$ توسط یک خانواده $(c_0, c_2, \ldots)$ $C = c_1$ از مدار ها به طورهم زمان با پیچیدگی اندازه چند جمله ای و پیچیدگی عمق لگاریتمی قابل تصمیم گیری است . مخصوصا هر c_n در C یک مدار جستجوی

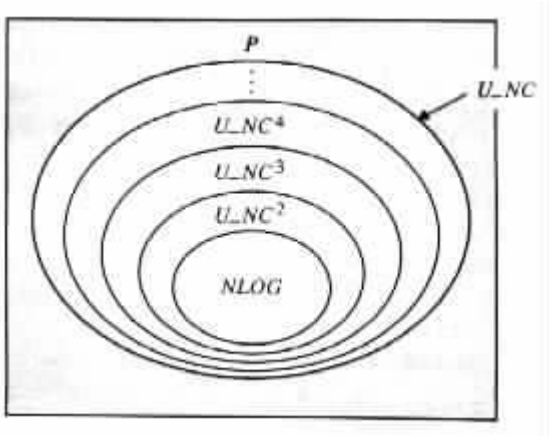
جدولی است که روی ورودی داده شده a_1 ـa_n خروجی ۱ می دهد اگـر وفقـط اگـر a_1 ـ$a_n = 1^n$ و 1^n در L باشد.

به هر حال،یک اثبات توسط قطری سازی (یعنی روی قطـر اصـلی مـاتریس مخـالف صفرو بقیه عناصر مساوی صفر) دلالت می کند که مسئله عضویت برای زبان یکانی

{ماشین تورینگ M_i رشته 1^i | . 1^i } تصمیم پذیر نیست .

فضای ترتیبی و زمان موازی

توسط نتیجه, ۷-۵-۱ و تعریف بالا و لم زیر، سلسله مراتب نشان داده شده در شکل ۷-۵-۶ طرح میشود.



شکل ۷-۵-۶ یک سلسله مراتب از مسائل تصمیم گیری بین *NLOG* و *P*.

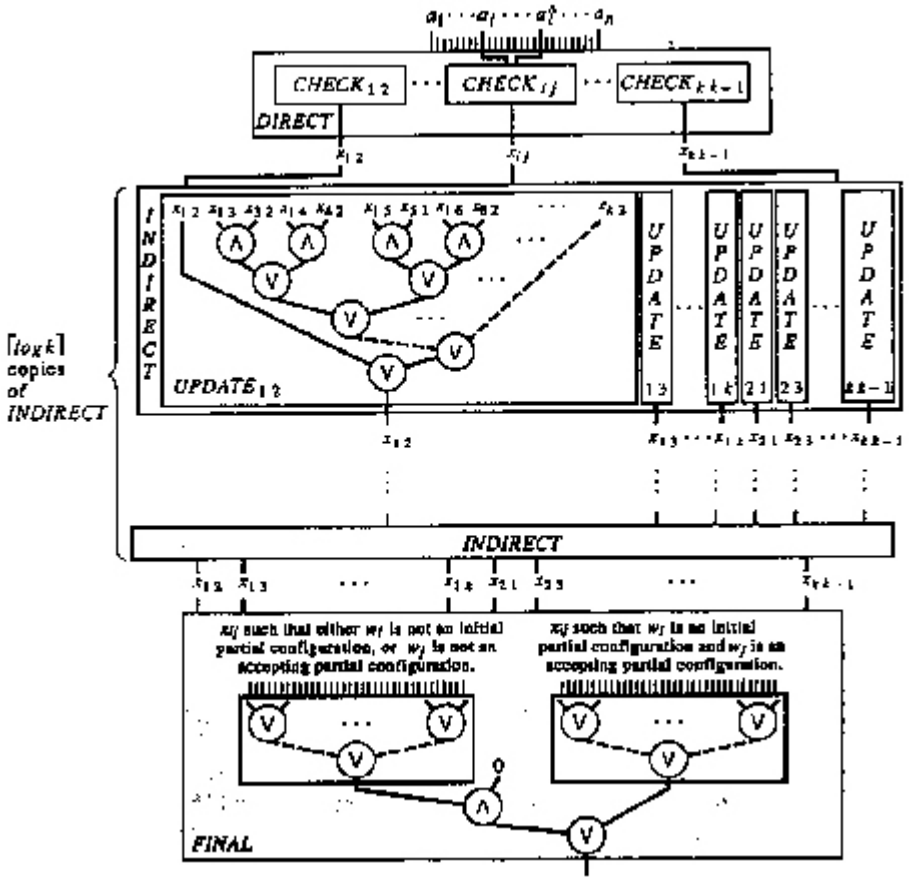
لم ۷-۵-۳ $NLOG \subseteq U\_NC^2$ اسـت.اثبـات : هـر ماشـین تورینـگ غیـر قطعـی بـا محدودیت مکانی $S(n) = O(\log n)$ بـه صـورت $< Q, \Sigma, \Gamma, \delta, q_0, B, F >$ = M با m نوار کاری کمکی را در نظـر بگیریـد. بـدون از بـین رفـتن کلیـت، $\Sigma=\{0, 1\}$ فرض می شود.بگذارید یک چند تـایی $w = (q, i, a, u_1, v_1, \ldots, u_m, v_m)$ یـک

پیکربندی پاره ای از M بر روی ورودی $a_1 \_a_n$ نامیده میشود اگر M یـک پیکربنـدی $(qa, u_1qv_1, \ldots, u_mqv_m)$ بـا $a_1 \_a_n\$ = \alpha a$ و $|\alpha| = i$ باشـد. بگذاریـد یـک پیکربندی پاره ای که پیکربندی پاره ای اولیه نامیده می شود اگـر بـا پیکربنـدی اولیـه متناظر باشد. بگذارید یک پیکربندی پاره ای یک پیکربندی پاره ای پذیرش نامیـده مـی شود اگر با پیکربندی پذیرش متناظرباشد.

هر پیکربندی پاره ای M به $O(\log n)$ فضا نیاز دارد.عدد K از پیکربندی هـای پـاره ای $w_1, \ldots, w_k$ کهM روی مجموعـه ورودی هـای بـا طـول n دارد شـرایط $k = 2^{O(\log n)}$ = $n^{O(1)}$ را ارضا می کند.

مثلا M مستقیما می تواند به پیکربندی پاره ای 'wاز پیکربنـدی پـاره ای w دسترسـی پیدا کند اگر w و 'w با برخی پیکربندی های Ω_w و $\Omega_{w'}$ از M ، متنـاظر باشـند. بـه طـوری کـه $\Omega_w \vdash \Omega_{w'}$. مـثلا M مـستقیما مـی توانـد بـه پیکربنـدی پـاره ای 'wاز پیکربندی پاره ای w دسترسی پیدا کند اگر w و 'w بـا برخـی از پیکربنـدی هـای Ω_w و $\Omega_{w'}$ از Mمتناظر باشند به طوری که $\Omega_w \vdash^* \Omega_{w'}$.

برای هر n،زبان $L(M) \subseteq \{0, 1\}^n$توسط مدار c_n تـصمیم پـذیر اسـت کـه شـامل $\log k + 2$ زیرمدار بـا نـام DIRECT, FINAL و $\log k$نمونـه از INDIRECT است.(شکل۷-۵-۷)



شکل ۷-۵-۷ یک مدار c_n که متناظربا ماشین تورینگ غیر قطعی با محدودیت مکان $O(\log n)$.

ساختار c_n تکیه بر این دارد که ماشین تورینگ M یک ورودی مفروض a_1 ـa_n را می پذیرد اگرو فقط اگر M پیکربندی پاره ای $w_0,. . . ,w_t$ را روی ورودی a_1 ـa_n داشته باشد، این چنین w_0ی یک پیکربندی پاره ای اولیه است، w_t یک پیکربندی پاره ای پذیرش است، و m میتواند مستقیماز w_{i-1} به w_i $1 \leq i \leq t$برسد.

DIRECT دارای مولفه $CHECK_{ij}$ برای هر جفت ممکن(w_i, w_j)از پیکربندی های پاره ای مجزا از M روی ورودی به طول n می باشد . $CHECK_{ij}$ روی ورودی a_1ـــa_n خروجی ۱ می دهد اگر w_i مانند w_j پیکربندی پاره ای از M روی ورودیa_1 ـa_n باشد و M بتواند مستقیماً از w_i بهw_j برسد. در غیر این صورت $CHECK_{ij}$ خروجی ۰ دارد.

مولفه $CHECK_{ij}$ یک مدار جستجوی جدولی است. به خصوص فرض شده است که $CHECK_{ij}$ با پیکربندی پاره ای $w_i = (q, l, a, u_1, v_1, \ldots, u_m, v_m)$ و$\hat{u}_1$. , $w_j = (\dot{q}, \dot{l}, \square>1$.,با هم متناظر هستند. در چنین مواردی $\hat{u}_m$). >m, , . . $CHECK_{ij}$ گره ثابت ۰ است وقتی M نمی تواند مستقیما از w_iبه w_jبرسد.از طرفی دیگر ،وقتی M می تواند مستقیما از w_iبه w_jبرسد،آنگاه $CHECK_{ij}$ یک مدار است که روی ورودی a_1 ـa_n خروجی یک دارد اگر و فقط اگر سمبل $l + 1$ مین در a_1 ـ$a_n$$، a باشد .

هر نمونه از زیرمدار INDIRECT مقادیر " متغییر های $x_{n\,n-1}$,.,$x_{1\,3}$,$x_{1\,2}$ به طور موازی تغییر می دهد که مقدار x_{ij} توسط مولفه $UPDATE_{ij}$ تغییر می یابد. به محض رسیدن به r امین INDIRECT ،متغییر x_{ij} مقدار یک را می گیرد اگر و فقط اگر M بتواند از w_iبه w_jبرسد در حد اکثر حرکت 2^r (به وسیله پیکربندی های پاره ای Mروی ورودی مفروض)، $r \leq \log k \leq 1$. خصوصا به محض خارج شدن از INDIRECT،ام r متغییر x_{ij} مقدار یک را می گیرداگر وفقط اگر M بتواند از w_iبه w_j برسد در حد اکثر حرکت 2^{r+1}.خصوصا به محض رسیدن به اولین INDIRECT، x_{ij} مقدار خروجی $CHECK_{ij}$را نگه میدارد .هر چند به محض خارج شدن از آخرین INDIRECT، x_{ij} مقدار یک را نگه می دارد اگر و فقط اگر بتواند از w_iبه w_j برسد.

FINAL تعیین می کند که آیا M می تواند از یک پیکربندی پاره ای اولیه به پیکربندی پاره ای پذیرش بر ورودی مفروض a_1 -a_n برسد،یعنی،آیا x_{ij} برابر ۱ برای تعدادی پیکربندی پاره ای اولیه w_i و تعدادی پیکربندی پاره ای پذیرش w_j داشته باشد.

زیرمدار های DIRECT اندازه $O(k^2) = n^{O(1)}$ و عمق ثابت دارد.هر یک از زیرمدارهای FINALو INDIRECT اندازه کمتر یا مساوی از $O(k^2) = n^{O(1)}$ داردوعمق بیشتراز $O(\log k)$ =($O(\log n)$ ندارد.در نتیجه ،مدار c_n حداکثر اندازه $O(k^2(\log k + 2)) = n^{O(1)}$دارد،وحداکثر عمق $O((\log k + 2)\log k) =$ $O(\log^2 n)$ دارد.

شمول *DLOG* در *U_NC*و حدس اینکه *U_NC* کاملاً شامل *P* است، اشاره میکند که مسایل *P*-complete توسط برنامه های موازی (برای مثال مسایل تمرین ۵-۱-۸) بطور کارامد قابل حل نیستند.بعلاوه قضیه زیر روشی برای تشخیص مسائلی که به طور کارآمد با برنامه های موازی قابل حل هستند ، ارائه می دهد .به علاوه، اثبات قضیه روشی خودکار برای بدست آوردن برنامه های موازی از برنامه های ترتیبی غیرقطعی متناظر که مسائل را حل میکند،ارائه می دهد .

نمادگذاری: در ادامه $NSPACE\_F(S(n))$ مجموعه ای از توابع محاسبه پذیر توسط مبدل های تورینگ غیر قطعی با محدودیت مکان $O(S(n))$ را مشخص می کند.

قضیه ۷-۵-۳ $NSPACE\_F(\log n) \subseteq U\_FNC^2$

اثبات: هر مبدل تورینگ $M = \langle Q, \Sigma, \Gamma, \Delta, \delta, q_0, B, F\rangle$باپیچیدگی فضا. $S(n) = O(\log n)$ را در نظر بگیرید .

فرض می شود که M تعدادی توابع f را محاسبه کند،علاوه بر این ،بدون از بین رفتن کلیت فرض می شود که Σ= Δ= {۰, ۱}. از M ،برای هر سمبل aدرΣ، یک ماشین تورینگ M_a =<Q_a, Σ, Γ, δ_a, q_{0a}, B, F_a> می تواند ساخته شود تا زبان {سمبل ورودی iام M روی ورودی a،x باشد } 1^i0x } را پذیرش کند.

خصوصا بر روی یک ورودی مفروض 1^i0x، M_a مقدار باینری i را روی یک نوار کار کمکی نگه داری می کند. سپس M_a محاسبه M روی ورودی x، را دنبال می کند . در طول شبیه سازی محاسبات ، M_aمقدار ذخیره شده iرا برای پیدا کردن سمبل iام در خروجی M استفاده می کند، در صورت نادیده گرفتن خروجی خودشان.، M_a،1^i0x را می پذیرد اگر و فقط اگر M یک محاسبه پذیرش روی ورودی x با یک سمبل a به عنوان سمبل iام در خروجی داشته باشد .

تابع f با خانواده ی ازمدارهای $C = (c_0, c_1, c_2,\ldots)$ فرم زیر ،محاسبه پذیراست . هر c_n یک خروجی y_1 ـ$y_2^{S(n)+1}$ با طول $2,2^{S(n)}$ را بر ورودی x_1 ـx_n فراهم می کند. هر زیررشته $y_{2j-1}y_{2j}$ از خروجی برابر با ۰۰, ۱۱, یا۱۰ است بسته به این که ایاسمبل iام در خروجی M, به ترتیب صفر یا یک یا تعریف نشده است،. y_{2j-1} توسط منفی کردن خروجی مداری بدست می آید که M_a را برای a = 1 روی ورودی $^j0x_1$ روی ۱ـx_n شبیه سازی می کند.

نتیجه ازلم ۷-۵-۳ به دست می آید. زیرا M_a یک ماشین تورینگ با محدودیت مکانی لگاریتمی برای a=0,a=1 است.

اثباتی مشابه آنچه برای قضیه قبل آمد، می تواند برای نشان دادن $NSPACE\_F(S(n)) \subseteq \bigcup_{d>0} U\_SIZE\_DEPTH\_F(2^{dS(n)}, S^2(n))$ برای هر تابع کاملاً قابل ساخت مکانی از $S(n) \geq \log$ استفاده شود .از این شمول و اثبات مکان لازم برای محاسبات ترتیبی و زمان لازم برای محاسبات موازی ، چندجمله ای هستند .

٧-٦ خانواده ی از مدار های یکنواخت و PRAM ها

این بخش نشان میدهد که خانواده ی از مدارها یکنواخت و PRAM هابصورت چند جمله ای به منابعی که نیاز دارند مرتبط اند. به عنوان یک استنباط، U_FNCدرست کلاسی از مسائل است که قابل حل بوسیله PRAMهایی هستند که پیچیدگی فضای چند جمله ای دارد وپیچیدگی زمان $O(\log^i n)$دارد.

نمادگـذاری:درادامـه، $PROCESSORS_TIME_F\ (Z(n), T(n))$مجمـوعی از توابع را مشخص می کندکه قابلیت محاسبه با PRAM هایی با هر دو پیچیدگی انـدازه $O(Z(n))$ و پیچیدگی زمان $O(T(n))$ دارد(تحت معیار ارزشی لگاریتمی).

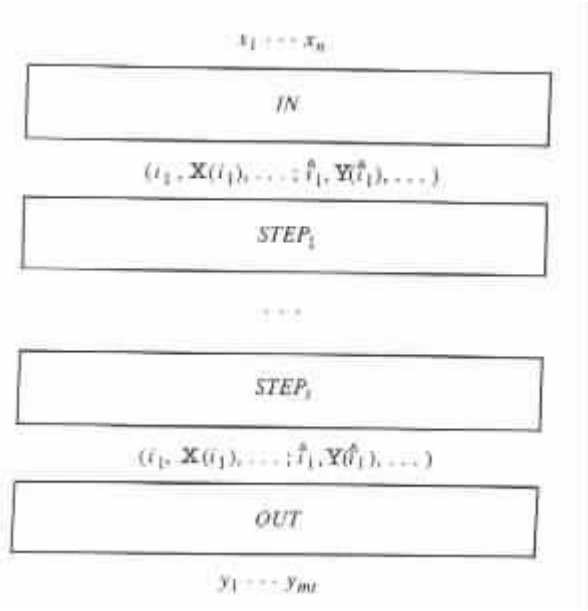
از PRAM هاتا خانواده ی از مدار های یکنواخت .

اثبات قضیه زیر نشان میدهد که چگونه قطعات هر PRAM مفروض تـوان بـاز شـدن برای ایجـاد یـک خـانواده مـدارهای یکنواخـت را دارنـد.مـورد رو بـه انحطـاطی کـه PRAMها محدود می شوند تا RAM ها در لم٧-٥-١ بررسی میشود.

قضیه ٧-٦-١ :اگر $\log T(n)$ و $\log Z(n)$ کاملا قابل ساخت مکانی باشند، $\log Z(n)$ و$O(T(n))$ آنگاه $n \leq [Z(n)T(n)]$

$$PROCESSORS\_TIME\_F(Z(n),T(n)) \subseteq \bigcup_{d\geq 0} U\_SIZE\_DEPTH\_F((Z(n)T(n))^d, T^d(n)).$$

اثبات: هرPRAM به صورت $\hat{M}$= <M, X, Y, A> با پیچیدگی اندازه $Z(n)$ و پیچیدگی زمان $T(n)$را در نظر بگیرید. با قضیه٧-٢-١ می توان فرض کرد که$\hat{M}$ یک CREW PRAM است. هرn را در نظر بگیرید و بگذارید $m = Z(n),\ t = T(n)$.محاسبات $\hat{M}$ با ورودی های با طول n قابلیت شبیه ساری با مدار c_n درشکل ٧,٦,١ را



دارد . شکل۷.-۶-۱

شکل ۷-۶-۱ مداری برای شبیه سازی محاسبات PRAM

سا ختار c_n

مدار c_n دارای ساختار زیربنایی مانند مدار c_n در اثبات لم ۷-۵-۱ است(شکل ۷-۵-۱ را ببینید). که شامل $t + 2$ مدار فرعی است،بطور مثال . . . IN, $STEP_1$ $STEP_t$,و OUT.زیر مدار IN یک مجموعه مفروض از ورودی با طول n به عنوان رمز گذازی برخی ورودی($v_1, . . . , v_N$) از$\hat{M}$ را در نظر بگیرید، که پیکربندی اولیه از $\hat{M}$ را تعیین می کند . $STEP_i$ پیکربندی که$\hat{M}$ بعد از گام i م به آن رسیده است را تعیین میکند. OUT خروجی $\hat{M}$ رااز خروجی $STEP_t$ استخراج میکند.

هر پیکربندی از$\hat{M}$ دارای فرم ($i_1, X(i_1), i_2, X(i_2), . . . ; \hat{i}_1, Y(\hat{i}_1), \hat{i}_2, Y(\hat{i}_2) . . . ;$ $i_1, A(i_1), i_2, A(i_2), . . . ; \hat{i}_1, V_1(\hat{i}_1), \hat{i}_2, V_1(\hat{i}_2), . . . ;. . . ; i_1, V_m(i_1), i_2,$ $V_m(i_2), . . .)$,, است،که$v_i(j)$ مقدار محلی متغیرjم از پردازنده M_i است.

$STEP_i$ شامل سه لایه است، یعنی WRITE READ, SIMULATE. لایه READ خواندن را از سلول های ورودی و سلول های حافظه مشترک شبیه سازی می کند. که در طول گام iام از محاسبات شبیه سازی شده ، اتفاق می افتد. لایه SIMULATE محاسبات داخلی را که در طول گام iام پردازنده های $M_m, \ldots M_1$ رخ می دهند ، را شبیه سازی می کند . لایه WRITE نوشتن را به سلول های خروجی و سلول های حافظه مشترک که در طول گام iام محاسبات رخ می دهد، شبیه سازی می کند .

بدون از بین رفتن کلیت، فرض شده است در هر مرحله پردازنده M_i از سلو ل ورودی $X(V_i(1))$ به توی $V_i(1)$ می خواند،و از سلو ل حافظه مشترک $X(V_i(2))$ بر روی $V_i(2)$. مشابها، فرض شده در هر مرحله M_i مقداری از $V_i(3)$ بر روی سلول خروجی $Y(V_i(4))$ می نویسد،و مقداری از $v_i(5)$ بر روی $A(V_i(6))$.

SIMULATE شامل یک زیرمدار SIM_RAM برای هر یک از پردازنده های $M_m, \ldots M_1$. است . محاسبات داخلی پردازنده ی M_j بوسیله ی یک SIM_RAM شبیه سازی میشود که ورودی اش $(i_1, V_j(i_1), i_2, V_j(i_2), \ldots)$. است.بدون از بین رفتن کلیت فرض میشود که اندیس j از M_j در $V_j(7)$ ذخیره شده است.

پیچیدگی c_n

مدار های IN, READ, WRITE و OUT هر یک می توانند یک مبدل تورینگ قطعی با محدودیت مکانی $O(\log (nZ(n)T(n)))$ را شبیه سازی کنند، که وظیفه خواسته شده را انجام میدهد.شبیه سازی ها میتوانند مانند اثبات لم ۷-۵-۳ باشد.بنابرین،هر یک از مدارهااندازه ی بیشتر از $(Z(n)T(n))^{O(1)}$ $\leq$ $(nZ(n)T(n))^{O(1)}$ و عمقی بیشتر از $T^{O(1)}(n) \leq (\log (nZ(n)T(n)))^{O(1)}$ ندارد. SIM _RAM می تواند یک پردازنده M_i را بطور غیر مستقیم مانند برهان لم ۷-۵-۱

شبیه سازی کند.این کار به وسیله یک مبدل تورینگ قطعی کاملا معادل با M_i انجام می شود. بنابرین هر SIM _RAM اندازه ی بیشتری از $T^{O(1)}(n)$ ندارد.

از خانواده ی از مدار های یکنواخت تا PRAM ها

در قضیه قبل شبیه سازی PRAMها با خانواده ی از مدارهای یکنواخت بررسی شد. شکل در قضیه بعد شبیه سازی در جهت دیگری را بررسی میکنیم.

قضیه۷-۶-۲

$$U\_SIZE\_DEPTH\_F(Z(n), D(n)) \subseteq \bigcup_{a>0} PROCESSORS\_TIME\_F(Z^a(n), D(n)log^a Z(n)).$$

اثبات: خانواده ای یکنواخت $C = (c_0, c_1, c_2,\ldots)$از مدارها با پیچیدگی اندازه $Z(N)$و پیچیدگی عمق $D(n)$را در نظر بگیرید. بگذارید $T = <Q, \Sigma, \Gamma, \delta, q_0, >$ B, F یک مبل تورینگ قطعی با محدودیت مکانی $S(n) = O(\log Z(n))$باشد، که $\{ (1^n, c_n) \mid n \geq 0 \}$ را محاسبه می کند. از T یک CREW PRAM به صورت $M = <M, X, Y, A>$ با پیچیدگی اندازه $Z^{O(1)}(n)$ و پیچیدگی زمان $D(n)\log$ $Z(n^{O(1)})$میتواند شبیه سازی محاسبات با C را با روشی آسان انجام دهد.

شبیه سازی گیت g_i بوسیله پردازنده M_i

برای هر گیت g_iدر c_n، PRAM M یک پردازنده متناظر M_i و یک سلول حافظه مشترک $A(i)$را به کار می گیرد. پردازنده M_i برای شبیه سازی عملیات g_i استفاده شده ، و $A(i)$برای ثبت نتیجه شبیه سازی استفاده شده است.)

در شروع هر شبیه سازی، M_i، $A(i)$ را با ۲ مقدار اولیه میدهد.بطوریکه مشخص کند که خروجی g_i هنوز در دسترس نیست.سپس M_i تا زمانی که عملوند در دسترس قرار گیرد منتظر می ماند.یعنی،تا زمانی که عملوند مقادیری متفاوت از ۲ برسد. M_i سلول

ورودی X(j) را به عنوان یک عملوند دارد اگر g_i یک ورودی از jمین گره ورودی x_j بگیرد. M_i سلول حافظه مشترک A(j)را به عنوان یک عملوند دارد اگر g_i یک ورودی از jمین گیت g_j بگیرد. زمانی که عملوندها در دسترس قرار می گیرند، M_i بر رویشان عملیات یکسانی با g_i انجام می دهد. M_i نتیجه را در Y(j) ذخیره میکند،اگر g_i jمین گره خروجی c_n باشد.وگرنه M_i نتیجه را در A(i) ذخیره میکند.

شناسایی گیت g_i بوسیله پردازنده M_i

قبل از شروع هر شبیه سازیc_n ، PRAM M برای هرگیتg_i در c_i تعیین میکند،که نوع t در {¬,∧,∨}از g_i چیست،و کدام پردازنده ها g_Lو g_R از g_iهستند. M این کار را به این صورت انجام می دهد که به طور موازی تعیین می کند که خروجی T روی ورودی با^n ۱ چیست و ارتباط هر زیررشته با فرم(g_i) و هر زیررشته با فرم(g_i, t, g_L, g_R) در خروجی متناظر پردازنده M_i است.

خروجی T بوسیله بکار بردن یک گروه $B_1,\ldots,B_{O(Z(n)\log Z(n))}$ پردازنده توسط M تعیین می شود . وظیفه پردازنده B_j مشخص کردن سمبل jم در خروجی T است.

B_j، پردازنده B_{ja} را برای هر سمبل ورودی a درالفبای خروجی Δ از T بکار می برد. وظیفه B_{ja} اخطار دادن به B_j است که آیا سمبل jم خروجی T سمبل a باشد. B_{ja} این کار را به این صورت انجام می دهد که ماشین تورینگ M_T محدودیت مکانی log Z(n)را شبیه سازی می کند که زبان { j a مین سمبل خروجی T است { ۱^n | را می پذیرد. شبیه سازی به طور موازی و بوسیله ی گروهی ازپردازنده ها انجام میشود که روش مشابه به آنچه در اثبات لم ۷-۵-۳ شرح داده شد، استفاده می گردد.

زمانی که خروجی T تعیین می شود، هر پردازنده B_j که سمبل "(" را دارد رشته"(...g_i)" که با $B_j,\ldots,B_{j+|(g_i...)|-1}$ نگهداری می شود را به پردازنده متناظر با M_i درM مبادله می کند.

سرانجام، هر پردازنده M_i که با رشته ی از شکل $(g_i, t, g_L$ $g_R)$ با آن ارتباط برقرار شده ، با قبلی خود ارتباط می گیرد تا گره های ورودی c_n را تعیین کند.