را در نظر بگیرید. در اینجا نمادی برای خلاصه نویسی چندین قاعده معرفی می نمایم بدین صورت که قواعد که طرف چپ یکسانی دارن، طرف چپ یکسان آنها در یک سطر و طرف راست متناظر توسط علامت (|) از هم جدا می شوند. قاعده $S \to aAb \mid \lambda$ بجای دو قاعده $S \to aAb$ و $S \to \lambda$ خلاصه نویسی شده اند. این گرامر هم ارز گرامر G در مثال ۱۱-۱ می باشد. این هم ارزی را می توان به سادگی با نشان دادن این که

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

اثبات کرد که به عنوان تمرین آورده شده است.

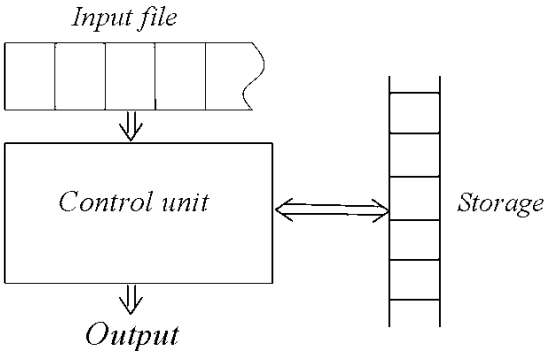
آتاماتا

یک آتاماتا یک مدل انتزاعی از کامپیوتر است. طوری که هرآتاماتا شامل بعضی ویژگی خاص می باشد. آتاماتا دارای مکانیزمی برای خواندن ورودی است. فرض می شود که این ورودی رشته الفبای داده شده است که روی یک **فایل ورودی** نوشته شده است. آتاماتا می تواند فایل ورودی را بخواند اما نمی تواند آن را تغییر دهد. فایل ورودی به سلول های تقسیم شده و هر سلول یک نشانه از الفبا را می تواند در خود جای دهد. فایل ورودی از چپ به راست خوانده می شود (یک نشانه هر بار). مکانیزم خواندن می تواند انتهای ورودی را تشخیص دهد با خواندن وضعیت انتهای فایل آتاماتا می تواند خروجی تولید کند. آتاماتا همچنین می تواند دارای یک **حافظه** موقت باشد که دارای تعداد نامتناهی سلول است که هر کدام می تواند یک نشانه از الفبا(لزوماً با الفبایی ورودی یکسان نمی باشد) را در خود جای دهد و می تواند محتوای سلول های حافظه موقت را تغییر دهد. در نهایت آتاماتا دارای یک **واحد کنترل** است که هر زمان می تواند در یکی از **وضعیت های متناهی داخلی** خود باشد و می تواند با دنباله مشخص تغییر وضعیت دهد. شکل ۴-۱ مکانیسم کل یک آتاماتای عمومی را نشان می دهد.

آتاماتا در هر زمان در یک وضعیت خاص است و حرف خاصی از ورودی را می خواند. وضعیت بعدی توسط **تابع تغییر وضعیت** تعیین می گردد. این تابع براساس علامت ورودی، وضعیت فعلی و محتوای حافظه وضعیت بعدی را معین می کند که می تواند باعث خروجی و یا تغییر در حافظه شود. وضعیت کنترل، ورودی و حافظه را یک **پیکربندی** می گوئیم. تغییر از یک وضعیت به وضعیت دیگر **حرکت** گفته می شود.

مدل عمومی، همه آتاماتا های که در این کتاب بحث می شود را در بر دارد. برای ادامه بحث نیاز است که تفاوت بین **آتاماتای معین** و **آتاماتای نامعین** را بیان نماییم. در آتاماتای معین اگر وضعیت فعلی، ورودی و محتوای حافظه را بدانیم می توانیم رفتار بعدی آتاماتا را دقیقاً تعیین کنیم اما در آتاماتا نامعین اینگونه نیست در هر زمان آتاماتای نامعین می تواند چند حرکت مختلف انجام دهد.

آتاماتا های که خروجی آن "آری " یا "نه " است یک **پذیرنده** است یک پذیرنده ورودی خود را یا قبول، یا رد می کند. اگرچه ما بیشتر با پذیرنده سروکار داریم اما آتاماتای که خروجی به صورت رشته دارد **تراگذر** نامیده می شود.



شکل ۴-۱

تمرینات

۱- به ازای تمام رشته های u و تمام n ها، بااستفاده از استقراء روی n ثابت کنید: $|u^n| = n|u|$

۲- معکوس یک رشته که قبلاً مطرح شد، می تواند بوسیله قوانین بازگشتی زیر بطور دقیق تر تعریف شود: برای همه $a \in \Sigma, w \in \Sigma^*$

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

با استفاده از این تعریف به ازای هر $u, v \in \Sigma^+$ ثابت کنید: $(uv)^R = v^R u^R$

۳- به ازای همه $w \in \Sigma^*$ ثابت کنید: $(w^R)^R = w$

۴- فرض کنید که $L = \{ab, aa, baa\}$ کدام یک از رشته های زیر در L^* هستند: $abaabaaabaa$، $aaaabaaaa$، $baaaaabaaaab$، $baaaaabaa$؟ کدامیک در L^* هستند.

۵- فرض $\Sigma = \{a,b\}$ و $L = \{aa, bb\}$ باشد برای توصیف $\overline{L}$ از عبارت مجموعه ها استفاده کنید.

۶- فرض L هر زبانی روی الفبای غیر تهی باشد. نشان دهید که L و $\overline{L}$ هر دو با هم نمی توانند متناهی باشند.

۷- آیا زبانهای وجود دارد که در آنها رابطه $\overline{L^*} = \overline{L}^*$ برقرار باشد؟

۸- برای تمام زبانهای L_1 و L_2 ثابت کنید: $(L_1 L_2)^R = L_2^R L_1^R$

۹- نشان دهید برای تمام زبانها $(L^*)^* = L^*$ می باشد.

۱۰- درستی یا نادرستی روابط زیر را ثابت نمایید.

(الف)برای تمام زبانهای L_2 و L_1 رابطه $(L_1 \cup L_2)^R = L_1^R \cup L_2^R$ همیشه برقرار است.

(ب) برای تمام زبانها L، $(L^R)^* = (L^*)^R$ است.

۱۱- گرامرهای روی $\Sigma = \{a,b\}$ که مجموعه های زیر را تولید می کند پیدا کنید:

(الف) تمام رشته های که فقط یک a دارند.

(ب) تمام رشته های که حداقل یک a دارند.

(ج) تمام رشته های که حداکثر سه a دارند.

(د) تمام رشته های که حداقل سه a دارند.

در هر مورد دلایل متقاعدکننده ای بیاورید که گرامرهای را که ارائه داده اید زبان مورد نظر را تولید می کند.

۱۲- توصیف ساده ای از زبان تولید شده توسط گرامر زیر ارائه دهید:

$$S \to aA$$
$$S \to bS$$
$$S \to \lambda$$

۱۳- زبانی که گرامر زیر را تولید می کند بنویسید:

$$S \to Aa$$
$$A \to B$$
$$B \to Aa$$

۱۴- برای هر یک از زبانهای زیر، گرامری که آن را تولید می کند بیابید:

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

(ب) $L_2 = \{a^n b^{2n} : n \geq 0\}$

(ج) $L_3 = \{a^{n+2} b^n : n \geq 1\}$

(د) $L_4 = \{a^n b^{n-3} : n \geq 3\}$

(ه) $L_1 L_2$

(و) $L_1 \cup L_2$

(ز) L_1^3

(ر) L_1^*

(ی) $L_1 - \overline{L_4}$

۱۵- گرامرهای زبانهای زیر را روی $\Sigma = \{a\}$ بنویسید.

الف) $L = \{w : |w| \bmod 3 = 0\}$

ب) $L = \{w : |w| \bmod 3 > 0\}$

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

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

۱۶-گرامری که زبان زیر را تولید می کند بنویسید.

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

توجیه کاملی برای جوابتان ارائه دهید.

۱۷- یک توصیف شفاهی برای زبان تولید شده توسط گرامر زیر ارائه نماید.

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

۱۸- با توجه به مثال ۱۳-۱،گرامرهایی که زبانهای زیر را تولید می کند پیدا کنید. فرض کنید: $\Sigma = \{a,b\}$

(الف) $L = \{w : n_a(w) = n_b(w) + ۱\}$

(ب) $L = \{w : n_a(w) > n_b(w)\}$

(ج) $L = \{w : n_a(w) = ۲n_b(w)\}$

(د) $L = \{w \in \{a,b\}^* : |n_a(w) - n_b(w)| = ۱\}$

۱۹- تمرین قبلی را با $\Sigma = \{a,b,c\}$ تکرار کنید.

۲۰- بحث مربوط به مثال۱۴-۱ را کامل کنید و نشان دهید که $L(G_۱)$ حقیقتاً زبان ارائه شده را تولید می کند.

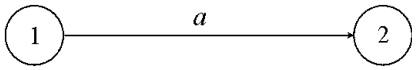
۲۱-آیا دو گرامر $S \to aSb|ab|\lambda$ و $\begin{matrix} S \to aAb \mid ab \\ A \to aAb \mid \lambda \end{matrix}$ معادلند؟ فرض کنید S در هر دو گرامر علامت شروع باشد.

۲۲- نشان دهیدگرامر $G = (\{S\},\{a,b\},S,P)$ با قوانین $S \to SS|SSS|aSb|bSa|\lambda$ معادل گرامر مثال ۱۳-۱ است.

۲۳- نشان دهید که گرامرهای $S \to aSb \mid bSa \mid SS \mid a$ و $S \to aSb \mid bSa \mid a$ معادل نیستند.

۱-۳ کاربردها

اگرچه هدف ما مسائل نظری و ریاضی زبانهای رسمی و آتاماتا است، اما این مفاهیم در حقیقت کاربردهای وسیعی در علم کامپیوتر و غیره دارند. در این بخش، ما مثالهای ساده آورده ایم تا به خواننده این دید را بدهیم که چیزهای که مطالعه می کند صرفاً انتزاع نیست بلکه بعضی اوقات در فهمیدن بسیاری از مسائل واقعی و مهم کمک می کند. زبانهای رسمی و گرامر ها بطور وسیعی در زبانهای برنامه نویسی استفاده می شود. در بسیاری از برنامه ها با فهمیدن بصری کم یا زیاد از زبانها برای نوشتن برنامه استفاده کرده ایم. اگر ما یک کامپایلر بنویسیم یا اگر بخواهیم دلیلی برای درست بودن برنامه بیاوریم در هر مرحله به تعریف دقیق زبانها نیاز داریم. در میان راه های که زبانهای برنامه نویسی را بتوان بطور دقیق تعریف کرد. گرامر ها شاید بیشترین کاربرد را داشته باشند. گرامر های که زبانها معمولی مانند C و پاسکال را توصیف می کنند خیلی گسترده می باشند به همین دلیل ما قسمتی از زبانها را آورده ایم.



شکل ۵-۱

مثال ۱۵-۱

مجموعه تمام شناسه متغیرها در پاسکال یک زبان است. بطور غیر رسمی آن مجموعه تمام رشته های است که با یک حرف شروع و بقیه می تواند تعدادی از اعداد یا حروف باشد. گرامر زیر این تعریف غیر رسمی از این زبان را نشان می دهد.

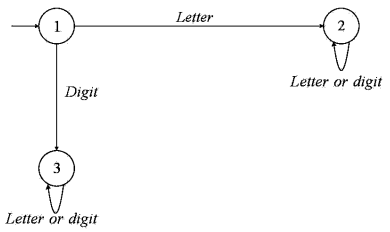
$$\langle id \rangle \to \langle letter \rangle \langle rest \rangle$$
$$\langle rest \rangle \to \langle letter \rangle \langle rest \rangle | \langle digit \rangle \langle rest \rangle | \lambda$$
$$\langle letter \rangle \to a|b|...|z$$
$$\langle digit \rangle \to ۰|۱|...|۹$$

در این گرامر $\langle id \rangle, \langle rest \rangle, \langle letter \rangle, \langle digit \rangle$ متغیر ها هستند و a،b،c،...،z و ۱،۲،...، ۹ کاراکتر پایانی هستند. اشتقاق شناسه a۰ بدین صورت می باشد.

$$\langle id \rangle \Rightarrow \langle letter \rangle \langle rest \rangle$$
$$\Rightarrow a \langle rest \rangle$$
$$\Rightarrow a \langle digit \rangle \langle rest \rangle$$
$$\Rightarrow a۰ \langle rest \rangle$$
$$\Rightarrow a۰$$

استفاده از گرامر برای تعریف یک زبان برنامه سازی بسیار معمول و مفید است اماجایگزین های راحتی وجود دارد برای مثال از پذیرنده هم می توان استفاده کرد. برای اینکه این مورد را بصورت مختصر بیان نماییم لازم است که آتاماتا را بصورت رسمی تعریف کنیم.

یک آتاماتا را می توان بوسیله یک گراف نشان داد که در آن رئوس، حالت های داخلی و یال ها حالت تغییر وضعیت هستند. برچسب یالها نشان می دهد که موقع گذر چه اتفاق افتاده است. برای مثال شکل ۵-۱ یک انتقال از حالت ۱ به حالت ۲ را نشان می دهد که این موقعی اتفاق می افتد که a بعنوان ورودی بیاید. حال به مثال قبلی بر می گردیم.



شکل ۶-۱

مثال ۱۶-۱

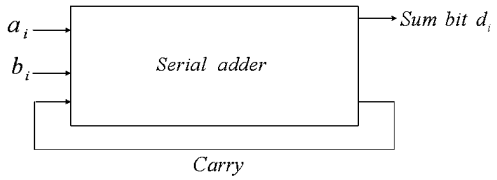
شکل ۶-۱ آتاماتای را نشان می دهد که همه شناسه های پاسکال را می پذیرد. ما فرض می کنیم حالت شروع آتاماتا حالت ۱ می باشد، این را با یک پیکان ورودی به این حالت نشان می دهیم. همیشه رشته ای که باید مورد بررسی قرار گیرد از چپ به راست در هر مرحله یک کاراکتر را می خوانیم. اگر علامت اول ورودی یک حرف باشد به وضعیت ۲ می رود و در آنجا هر چیز دیگری (حرف یا رقم) که بیابید در آن وضعیت می مانیم، وضعیت ۲ وضعیت قبول (yes) است و اگر علامت اول رقم باشد به وضعیت ۳ می رویم و در آنجا هر چیز دیگر بیاید در آن وضعیت می مانیم وضعیت ۳ وضعیت رد (NO) می باشد. در راه حل مان فرض کردیم که بجز حرف و رقم هیچ نوع ورودی دیگری نداریم.

کامپایلرها و دیگر مترجمها که یک زبان را به زبان دیگر تبدیل می کنند از این ایده که در مثال قبل آمده استفاده های زیادی می کنند. زبانهای برنامه نویسی را می توان بواسطه گرامر همانند مثال ۱۵-۱ به طور دقیق تعریف نمود. هم گرامرها و هم آتاماتا نقش اساسی در فرآیند تصمیم گیری در مورد پذیرش یا عدم پذیرش یک قطعه کد توسط یک زبان برنامه سازی ایفا می کنند. مثال فوق یک راهنمایی مقدماتی در مورد چگونگی انجام اینکار است. مثال بعدی این مورد را بیشتر توضیح می دهد.

یک زمینه مهم دیگر کاربرد طراحی دیجتال است که در آنجا مفاهیم مربوط به تراگذرها مطرح می شود. اگرچه این موضوع در اینجا مورد بحث گسترده واقع نخواهد شد اما یک مثال ساده داده خواهد شد. اصولاً به کامپیوتر می توان به عنوان یک آتاماتا نگریست گرچه چنین نگرشی ممکن است که خیلی مناسب نباشد. فرض کنید که ثبات های داخلی و حافظه اصلی کامپیوتر واحد کنترل آتاماتا باشند. آنوقت آتاماتا دارای ۲^n وضعیت داخلی است که n در اینجا تعداد بیت های ثبات ها و حافظه است. حتی با یک n کوچک این عدد آنقدر بزرگ می شود که با این تعداد وضعیت امکان کار کردن وجود ندارد. اما اگر به یک واحد خیلی کوچکتر از این نگاه کنیم آنوقت نظریه آتاماتا یک ابزار طراحی سودمند است.

$a_i \backslash b_i$	0	1
0	0 *NO carry*	1 *NO carry*
1	1 *NO carry*	0 *Carry*

شکل ۷-۱



شکل۸-۱

مثال۱۷-۱

یک جمع کننده باینری جزء لاینفک هر کامپیوتر همه منظوره است. چنین جمع کننده ای دو رشته از بیت ها را که نمایانگر دو عدد هستند به عنوان ورودی دریافت می کند و جمع آنها را بعنوان خروجی تولید می نماید. برای سادگی کار فرض کنید که فقط اعداد مثبت را جمع می زنیم و نمایشی که برای اعداد استفاده می کنیم به صورت $x = a_0 a_1 .... a_n$ که نمایانگر عدد صحیح $v(x) = \sum_{i=0}^{n} a_i 2^i$ است. این نحوه نمایش عکس نمایش اعداد باینری به صورت معکوس است.

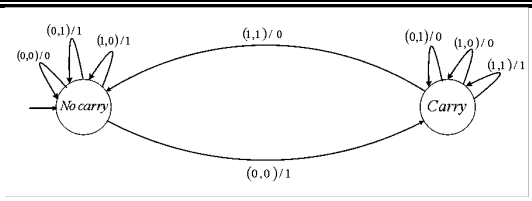
یک جمع کننده سری دو عدد مانند $x = a_0 a_1 .... a_n$ و $y = b_0 b_1 .... b_n$ را به صورت بیت به بیت از چپ به راست جمع می زند. جمع هر بیت یک رقم جمع و یک رقم نقلی تولید می کند. جدول باینری در شکل ۷-۱ این فرایند را ساده کرده است.

این فرایند در شکل ۸-۱ به صورت تصویری نشان داده شده است. این تصویر نشان می دهد که یک جمع کننده جعبه ای است که دو بیت را بعنوان ورودی می پذیرد و جمع و بیت نقلی آنها را تولید می کند. این تصویر عملکرد جمع کننده را توضیح می دهد اما چیزی در مورد ساختار درونی آن نمی گوید. یک آتاماتا (یک تراگذر در این مورد) می تواند این مسئله را بسیار واضح تر بیان نماید.

ورودی تراگذر زوج بیت های (a_i, b_i) و خروجی آن بیت جمع d_i است. بازهم می توانیم آتاماتون را بصورت یک گراف برچسب دار نمایش دهیم البته یال های آن را با $(a_i, b_i)/d_i$ علامت گذاری می کنیم. رقم نقلی از یک بیت به بیت بعدی توسط آتاماتا از طریق دو وضعیت داخلی " نقلی " و " بدون نقلی " بخاطر سپرده

می شود. در ابتدا تراگذر در وضعیت " بدون نقلی " خواهد بود تا دریافت زوج (۱و) در این وضعیت باقی می ماند. زوج (۱و۱) آتاماتون را به وضعیت" نقلی " می برد. این نقلی در زمانی که زوج بعدی خوانده می شود اعمال می گردد. تصویر کامل یک جمع کننده سری در شکل ۸-۱ نشان داده شده است. این گراف را باچند مثال دنبال کنید تا مطمئن شوید که درست کار می کند.

همانگونه که این مثال نشان می دهد آتاماتا مانند پلی میان توصیف عملی بسیار سطح بالای یک مدار و پیاده سازی منطقی آن با ترانزیستور، گیت و فلیپ فلاپ است. آتاماتا به روشنی منطق تصمیم را نشان می دهد و در عین حال آنقدر صوری است که پردازش ریاضی آنرا امکان پذیر می سازد. به همین دلیل روش های طراحی رقمی تکیه زیادی روی مفاهیم نظریه آتاماتا دارد. در صورت تمایل به مطالب مربوط به این موضوع می توان به مراجع مربوطه رجوع شود. برای مثال (*kavahi*,۱۹۷۸)



شکل ۹-۱

تمرینات

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

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

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

۴- فرض کنید یک زبان برنامه نویسی، شناسه هایی را که با یک حرف شروع می شوند و شامل حداقل یک و حداکثر سه رقم هستند و می توانند هر تعداد حرف را دارا باشند، تولید کند.یک گرامر و یک پذیرنده برای چنین مجموعه ای از شناسه ها ارائه کنید.

۵- یک گرامر برای اعلان var در پاسکال بنویسید.

۶- در سیستم اعداد رومی، اعداد بوسیله رشته هایی روی الفبای $\{M, D, C, L, X, V, I\}$مشخص می شوند. پذیرنده ای طراحی کنید که رشته هایی را که اعداد رومی را بدرستی نشان می دهند قبول کند. برای راحتی، قرار دادی به این صورت داریم که مثلاً عدد ۹ به جای اینکه بصورت VIIII نوشته شود، بصورت IX نشان داده شود.

۷- فرض کردیم که یک آتاماتا در یک چار چوب مراحل زمانی جداگانه کار می کند. اما این فرض تأثیر کمی روی بحث بعدی ما دارد. اگر چه در طراحی دیجیتال، عنصر زمان با اهمیت فرض می شود. به منظور هم زمان شدن سیگنال هایی که از قسمت های مختلف یک کامپیوتر می آیند، مدارهای تأخیر مورد نیاز هستند یک تراگذر تک-تأخیره وسیله ای است که ورودی را یک واحد زمان دیرتر مجدداً تولید می کند (بعنوان یک سیستم دائمی از سمبل ها در نظر گرفته می شود). بخصوص، اگر تراگذر بعنوان ورودی یک علامت a را در زمان t بخواند،

آنگاه همان علامت را در زمان $t+۱$بعنوان خروجی تولید خواهد کرد. در زمان $t=۰$ خروجی های تراگذر چیزی نیستند. این مطلب را به اینصورت می گویم که تراگذر، ورودی $a_۱a_۲....$ را به خروجی $\lambda a_۱a_۲....$ ترجمه می کند. گرافی رسم کنید که یک تراگذر تک-تأخیره طراحی شده روی $\Sigma=\{a,b\}$ را نشان دهد.

۸- یک تراگذره n تأخیره وسیله ای است که ورودی را چندین واحد زمانی (n) بعد، مجدداً تولید می کند. یعنی ورودی $a_۱a_۲....$ بصورت $\lambda^n a_۱a_۲....$ ترجمه می شود. بدین معنا که تراگذر هیچ خروجی را برای n زمان اول تولید نمی کند.

(الف) یک تراگذر دو تأخیره روی $\Sigma=\{a,b\}$بسازید.

(ب) نشان دهید که یک تراگذر n تأخیره باید حداقل $|\Sigma|^n$ وضعیت داشته باشد.

۹- متمم دو یک رشته بیتی که یک عدد صحیح مثبت را نشان داده است ابتدا با متمم کردن تک تک بیت ها و سپس با اضافه کردن ۱ به کم ارزش ترین بیت بدست می آید. یک تراگذر طراحی کنید که رشته های بیتی را به متمم دوی آن ترجمه کند، با فرض اینکه عدد باینری به همان صورتی که در مثال ۱۷-۱ نشان داده شد بازنمایی شده باشد و بیت های کم ارزش تر در قسمت چپ رشته باشند.

۱۰-یک تراگذر طراحی نمایید که اعداد باینری را به اعداد دهدهی تبدیل نماید برای مثال عدد دودویی ۰۰۱۱۰۱۱۱۰ را به عدد ۱۵۶ تبدیل نماید.

۱۱- فرض کنید $a_۱a_۲....$ یک رشته بیتی ورودی باشد. یک تراگذر طراحی کنید که زوجیت هر زیر رشته با سه بیت را محاسبه کند. بویژه، تراگذر باید خروجی زیر را تولید کند:

$$\pi_۱=\pi_۲=۰$$
$$\pi_۱=(a_{i-۲}+a_{i-۱}+a_i)\bmod ۲ \quad i=۳,۴$$

بعنوان مثال، ورودی ۱۱۰۱۱۱باید خروجی ۰۰۰۰۰۱ را تولید کند.

۱۲- یک تراگذر طراحی نماید که رشته $a_۱a_۲a_۳....$ را بعنوان ورودی دریافت و باقیمانده مقدار دهدهی هر۳ بیت را بر ۵ را به عنوان خروجی برگرداند. برای توضیح بیشتر تراگذر باید خروجیهای $m_۱, m_۲, m_۳, ...$ را تولید نماید.

$$m_۱=m_۲=۰$$
$$m_i=(۴a_i+۲a_{i-۱}+a_{i-۲})\bmod ۵ \quad i=۳,۴$$

۱۳- کامپیوتر های رقمی معمولاً با استفاده از رمز گذاری کل اطلاعات را بصورت رشته های بیتی نمایش می دهند. به عنوان مثال، اطلاعات حروفی می توانند با استفاده از سیستم شناخته شده ASCII رمز گذاری شوند. برای این تمرین، دو الفبای $\{a,b,c,d\}$ و $\{۰,۱\}$را در نظر بگیرید که یک رمزگذاری از الفبای اول به دوم بصورت زیر نشان داده می شود:

$$a\to ۰۰,\ b\to ۰۱,\ c\to ۱۰,\ d\to ۱۱$$

یک تراگذار برای رمز گشایی رشته ها روی $\{۰,۱\}$ به پیام اصلی بسازید. بعنوان مثال ورودی ۰۱۰۰۱۱ باید خروجی dfa را تولید کند.

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

فصل ۲

آتاماتای متناهی

معرفی مفاهیم محاسبات در فصل اول، بخصوص آتاماتا خیلی مختصر و غیر رسمی بود ما فقط آتاماتا را معرفی و گفتیم که یک آتاماتا را می توان به صورت یک گراف نشان داد. حال سعی می کنیم که در این خصوص کمی رسمی تر برخورد نماییم و نتایج دقیق تر بدست آوریم. ابتدا به مفهوم آتاماتای متناهی (FA) که مفهومی ساده ای دارد، می پردازیم. این آتاماتا دارای حافظه موقت نیست بنابراین فایل ورودی نمی تواند دوباره نوشته شود. لذا یک آتاماتای متناهی در به یادآوردن اطلاعات در طول محاسبه مشکل دارد و مقدار کمی اطلاعات می تواند در واحد کنترل نگهداری شود که آن هم به صورت قرارگرفتن در یک وضعیت. بدلیل اینکه تعداد وضعیت ها در آتاماتایی متناهی محدود می باشد مقدار اطلاعاتی که می تواند نگهداری کند محدود می باشد. مثال ۱۶-۱ نمونه از یک پذیرنده متناهی می باشد.

۲-۱ پذیرنده متناهی معین (قطعی)

اولین نوع آتاماتای مورد مطالعه آتاماتای متناهی است که دارای عملکرد معینی می باشد. حال با یک تعریف رسمی از پذیرنده های معین شروع می کنیم.

تعریف رسمی یک پذیرنده متناهی معین (قطعی)

تعریف ۱-۲

یک **پذیرنده متناهی معین** یا **dfa** بوسیله ۵ تایی

$$M = (Q, \Sigma, \delta, q_{.}, F)$$

تعریف می شود که در آن:

Q: مجموعه متناهی از **وضعیت** ها.

Σ: مجموعه متناهی از نشانه ها که **الفبای ورودی** نامیده می شود.

δ: $Q \times \Sigma \to G$ یک تابع کلی است به نام **تابع تغییر حالت**.