

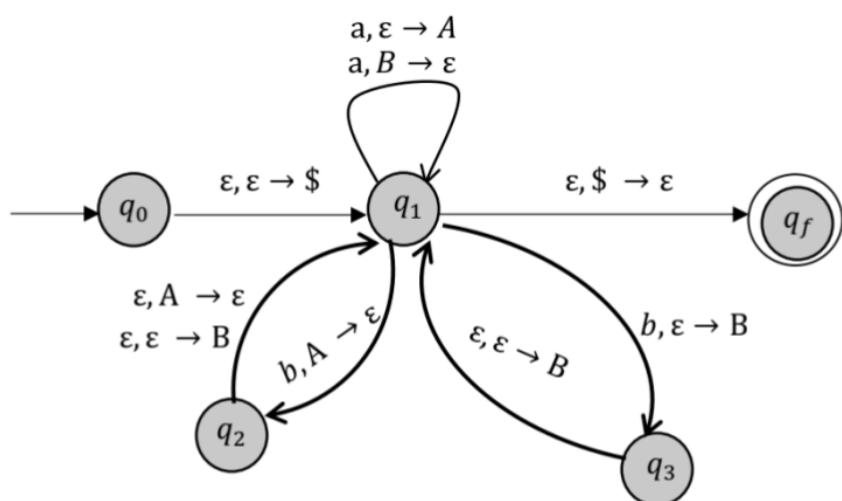


برای زبان‌های زیر PDA رسم کنید. (1)

a. $\{w | w \in \{1, 0\}^*\}$ تعداد ۱‌ها دو برابر تعداد ۰‌ها است.

برای رسم PDA این زبان اگر ۱ در ورودی دیده شود، اگر ۰ در استک باشد، آن را پاپ می‌کند و در غیر این صورت (۱ یا \\$ روی استک باشد) لازم است یک ۱ دیگر در استک پوش شود تا بتوان بعداً آن را با ۲ تا ۰ در رشته match کرد. زمانی که یک ۰ روی ورودی دیده شود، اگر ۱ در استک باشد، آن را پاپ می‌کند و از آنجایی که باید به ازای این ۰ یک ۱ دیگر هم در رشته باشد، اگر همچنان ۱ سر استک بود آن را پاپ می‌کند و اگر نه یک ۰ در استک پوش می‌شود. اما در صورتی که روی ورودی ۰ باشد و در استک چیزی به جز ۱ باشد، ۲ عدد ۰ در استک پوش می‌شود.

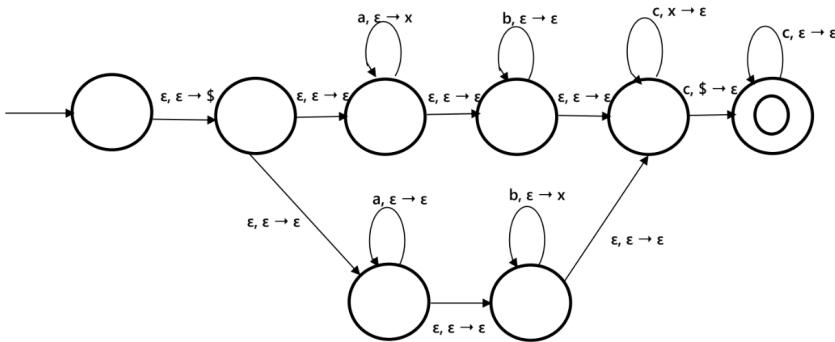
زمانی که استک خالی شود به این معنا است که رشته‌ای داشته‌ایم که به ازای هر ۰، ۲ عدد ۱ داشته است.



$$b. \quad \{a^i b^j c^k \mid k \geq \min(i, j)\}$$

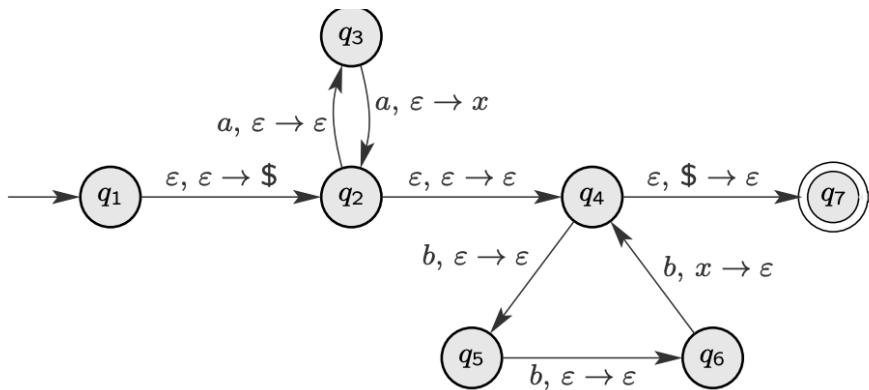
برای اینکه رشته‌ای در این زیان قابل قبول باشد، یا تعداد c ها باید از تعداد a ها بیشتر باشد یا تعداد c ها از تعداد b ها باید بیشتر باشد. بنابراین پاسخ اجتماع این دو حالت است.

به ازای هر a که در ورودی می‌بینیم یک x در استک پوش می‌کنیم. b ها را رد کرده و سپس به ازای هر c یک x پاپ می‌کنیم. اگر رشته به پایان برسد و به انتهای استک رسیده باشیم و همچنان c در ورودی باشد، رشته قابل قبول است. همین کار را برای a ها نیز انجام می‌دهیم.



$$c. \quad \{a^{2n} b^{3n} \mid n \geq 0\}$$

در این قسمت به ازای هر ۲ تا ورودی a یک x در استک پوش می‌کنیم و سپس به ازای هر ۳ تا ورودی b یک x از استک پاپ می‌کنیم. در این صورت تعداد ۰ ها $2n$ و تعداد ۱ ها $3n$ محاسبه می‌شود.

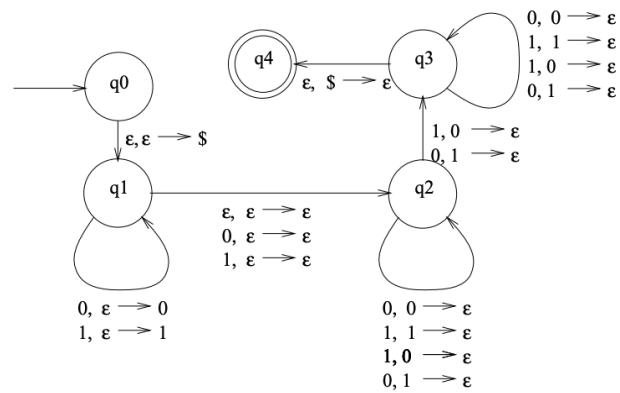


d. $\{ w \mid w \in \{a, b\}^* \text{ به طوری که نامتقارن باشد} \}$

(رشته های متقارن روی زبان a و b عبارت اند از $(\dots, a, b, aa, bb, aba, bab)$:

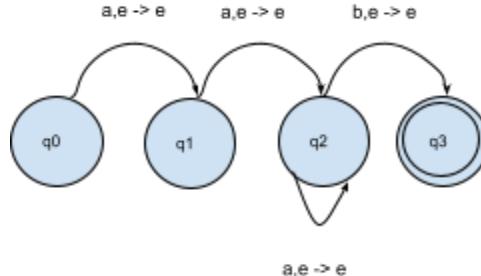
در این قسمت می توانیم از PDA مربوط به رشته هایی که متقارن هستند کمک بگیریم. برای اینکار دقیقاً همه چیز را یکسان قرار می دهیم تنها هنگامی که میخواهیم از استک پاپ کنیم باید مطمئن شویم که همچنان حداقل یک a وجود دارد در صورتی که باید b میبینیم و بر عکس.

ابتدا نصف رشته را خوانده و وارد استک میکنیم. سپس تمام حروف بعدی را می خوانیم. در صورتی که به یک mismatch بخورد کنیم از این استیت خارج شده و وارد استیت بعدی می شویم و حالا هر حرفی در ورودی باشد، آن را با حرف بعدی استک پاپ میکنیم و رشته پذیرفته می شود.

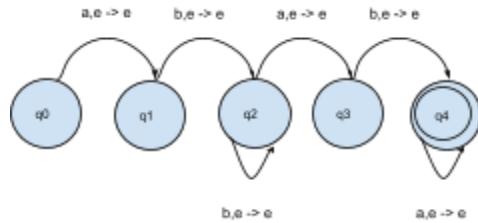


(2) برای هر یک از زبان های منظم زیر یک NPDA رسم کنید.

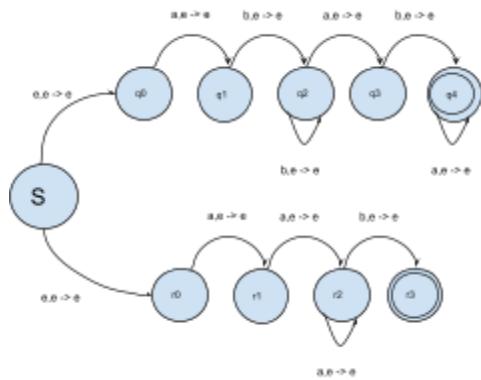
a. $L1 = L(aaa^*b)$



b. $L2 = L(abb^*aba^*)$



c. $L1 \cup L2$



ای رسم کنید که گرامر زیر را بپذیرد. (3

$$S \rightarrow aABB \mid aAA$$

$$A \rightarrow aBB \mid a$$

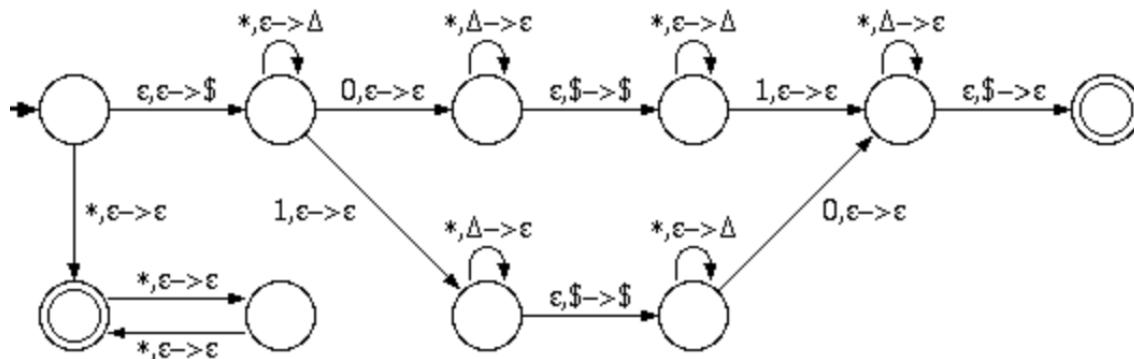
$$B \rightarrow bBB \mid A$$

$((s_0, e, e), (s_1, S))$	$((s_0, e, e), (s_1, S))$
$((s_1, e, S), (s_1, aABB))$	$((s_1, a, S), (s_1, ABB))$
$((s_1, e, S), (s_1, aAA))$	$((s_1, a, S), (s_1, AA))$
$((s_1, e, A), (s_1, aBB))$	$((s_1, a, A), (s_1, BB))$
$((s_1, e, B), (s_1, bBB))$	$((s_1, a, A), (s_1, e))$
$((s_1, e, B), (s_1, A))$	$((s_1, b, B), (s_1, BB))$
$((s_1, a, a), (s_1, e))$	$((s_1, a, B), (s_1, BB))$
$((s_1, b, b), (s_1, e))$	$((s_1, a, B), (s_1, e))$

۴

(4) مشخص کنید هر PDA چه زبانی را می‌پذیرد.

.a



در این PDA می‌بینیم که از اولین استیت دو راه خواهیم داشت. راه اول راهی است که به ازای هر ورودی‌ای که داشته باشیم، به یک استیت accept میرسیم. و سپس با هر ورودی‌ای که داشته باشیم از این استیت در می‌آیم و سپس در ازای هر ورودی دیگری که داشته باشیم مجدداً به استیت accept میرسیم. این مسیر در واقع نشان دهنده تمامی رشته‌های با طول فرد است. در مسیر دوم با هر ورودی‌ای که داشته باشیم ابتدا یک مثلث وارد استک می‌شود.

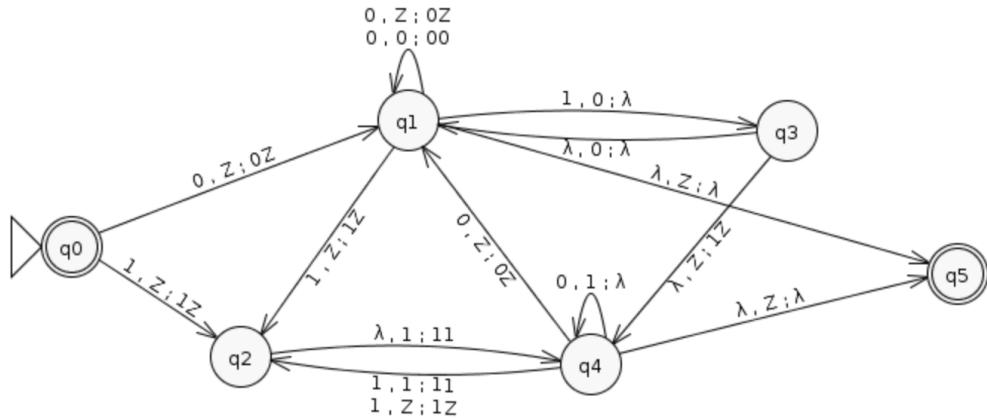
سپس در صورتی که در ورودی ۰ داشته باشیم، آن را خوانده و سپس به ازای هر ورودی دیگری که داشته باشیم مثلث‌ها را پاپ می‌کنیم تا به انتهای رشته برسیم. سپس به ازای تمام ورودی‌های دیگر مجدداً مثلث را پوش کرده و بعد ۱ را از ورودی می‌خوانیم و سپس به ازای تمام ورودی‌های دیگر مثلث را از استک پاپ می‌کنیم. در صورتی که در ورودی پس از پوش کردن مثلث‌ها ۱ داشته باشیم نیز مانند مسیر قبلی خواهد بود و در نیمه دوم به دنبال عدد ۰ خواهیم بود.

این مسیر به منظور تمام رشته‌هایی است که اگر فرض کنیم این رشته به دو قسمت مساوی تقسیم شده باشد، حداقل ۱ بیت آن متفاوت است.

بنابر این این PDA در واقع اجتماعی از رشته‌هایی است که یا طول آن‌ها فرد است و یا آنکه با اینکه طولشان زوج است ولی وقتی این رشته را تقسیم بر دو قسمت مساوی می‌کنیم، قطعاً این دو قسمت باهم برابر نیستند. بنابراین می‌توان گفت این PDA زبان زیر را توصیف می‌کند:

$$L = \text{complement of } \{ww \mid w \text{ in } \{0, 1\}^*\}$$

.b



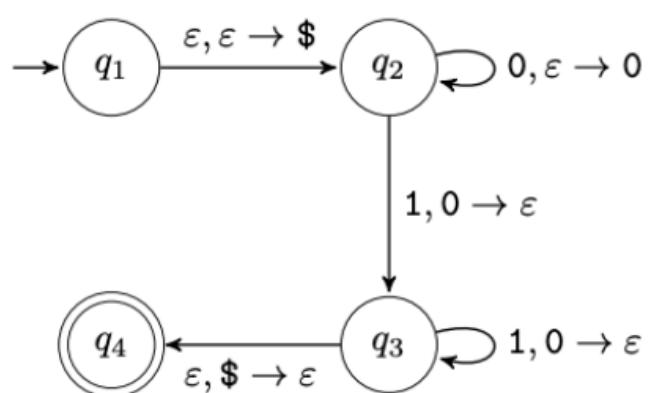
در این PDA زمانی که یک ۰ از ورودی می‌خوانیم اگر استک خالی باشد یا در آن ۰ باشد، یک ۰ در استک پوش می‌کنیم. در غیر این صورت یک ۱ از استک پاپ می‌کنیم. زمانی که ۱ را از ورودی می‌خوانیم اگر استک خالی باشد یا اگر در استک ۱ باشد، دو عدد ۱ به استک پوش می‌کنیم. در غیر این صورت اگر ۲ عدد ۰ در استک داشته باشیم آن دو را پاپ می‌کنیم و اگر ۱ عدد ۰ در استک باشد، آن را پاپ کرده و یک عدد ۱ در استک پوش می‌کنیم.

در واقع این PDA مقدار 2^* (تعداد ۱ ها) - (تعداد ۰ ها) را نگه می‌دارد. به عبارتی می‌توان گفت زیان این

PDA به صورت زیر است:

$$L = \{w \in \{0, 1\}^* \mid w \text{ has twice as many } 0s \text{ as } 1s\}$$

PDA زیر را به گرامر متناظرش تبدیل کنید. (5)



برای تبدیل PDA به گرامر متناظرش ابتدا باید این PDA را تبدیل به یک simplified PDA داده شده در این سوال خودش simplified است زیرا قبل از قبول رشته استک را خالی میکند و همچنین هر گذار یا تنها یک سمبول به استک پوش میکند و یا تنها یک سمبول از استک پاپ میکند و این دو کار را همزمان انجام نمیدهد.

در ادامه طبق اثبات لم ۲۷ کتاب که در صفحه ۱۲۲ کتاب آورده شده است قواعد گرامر را تعریف میکنیم.

start variable : A_{14}

طبق گفته کتاب ابتدا قواعد مدل اول را مینویسیم: برای هر $1, 0 \in \Sigma_\epsilon$ و $u \in \Gamma$ و $q, p, r, s \in Q$ اگر $A_{pq} \rightarrow aA_{rs}b$ عضو گرامر است. برای فهم بهتر این قواعد شامل (r, u) شود و $\delta(s, b, u) \in Q$ شامل (شود، قانون q, ϵ) است. برای زیر جدول را نگاه کنید.

مقدار U	گذاری که U را پوش میکند	گذاری که U را پاپ میکند	قاعده گرامر متناظر
\$	$q_1 \rightarrow q_2$	$q_3 \rightarrow q_4$	$A_{q_1 q_4} \rightarrow A_{q_2 q_3}$
0	$q_2 \rightarrow q_2$	$q_2 \rightarrow q_3$	$A_{23} \rightarrow A_{22}$
0	$q_2 \rightarrow q_2$	$q_3 \rightarrow q_3$	$A_{23} \rightarrow A_{23}$

سپس قواعد مدل دوم را مینویسیم: برای هر $p, q, r \in Q$ یک قانون به صورت $A_{pq} \rightarrow A_{pr}A_{rp}$ به گرامر اضافه میکنیم.

$$\begin{array}{llll}
 A_{14} \rightarrow A_{12}A_{24} & A_{11} \rightarrow A_{11}A_{11} & A_{14} \rightarrow A_{11}A_{14} & A_{11} \rightarrow A_{13}A_{31} \\
 A_{14} \rightarrow A_{13}A_{34} & A_{11} \rightarrow A_{12}A_{21} & A_{14} \rightarrow A_{14}A_{44} & A_{11} \rightarrow A_{14}A_{41} \\
 & & A_{13} \rightarrow A_{12}A_{23} & \text{و به همین صورت این} \\
 & & A_{13} \rightarrow A_{11}A_{13} & \text{قواعد ادامه دارند.} \\
 & & A_{13} \rightarrow A_{13}A_{33} & \\
 & & A_{13} \rightarrow A_{14}A_{43} & \\
 & & A_{12} \rightarrow A_{12}A_{22} & \\
 & & A_{12} \rightarrow A_{11}A_{12} & \\
 & & A_{12} \rightarrow A_{13}A_{32} &
 \end{array}$$

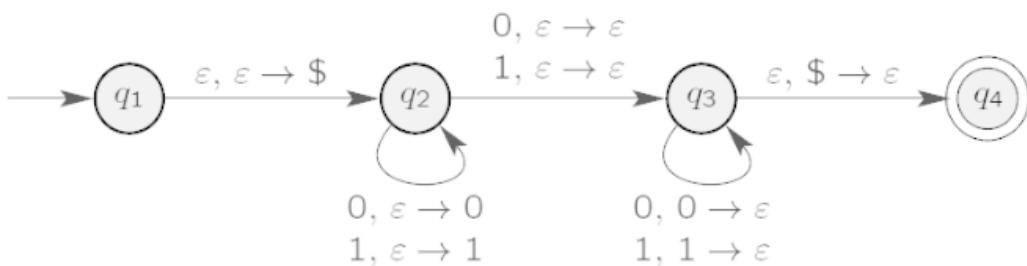
و در نهایت قواعد مدل سوم: برای هر $p \in Q$ یک قانون به صورت $\epsilon \rightarrow A_{pp}$ به گرامر اضافه میکنیم.

$$A_{11} \rightarrow \epsilon$$

$$A_{22} \rightarrow \epsilon$$

$$A_{33} \rightarrow \epsilon$$

$$A_{44} \rightarrow \epsilon \quad \text{PDA زیر را در نظر بگیرید.} \quad (6)$$



a. توضیح دهید چرا زبانی که این PDA میپذیرد با $L = \{w \mid w \in \{0, 1\}^*\}$ متفاوت است.

در این PDA گذار از q_2 به خودش n سمبول اول را از ورودی میخواند و در استک پوش میکند و گذار q_3 به خودش نیز با هر بار پاپ کردن از استک، n سمبول انتهایی را با n سمبول اول موجود در استک match میکند. همچنین در گذار از q_2 به q_3 $n+1$ ام و یا عنصر میانی رشته مشخص میشود و از آنجایی که طول رشته را فرد میکند لازم نیست با چیزی match شود در نتیجه چیزی در استک پوش و یا از آن پاپ نمیکند. پس نتیجه میگیریم تفاوتی که زبانی که PDA میپذیرد با زبان L دارد این است که طول رشته مورد نظر تنها میتواند فرد باشد پس زبانی که PDA میپذیرد زبان زیر میباشد:

$$B = \{ w \in \{0, 1\}^* \mid w = w^R \text{ and the length of } w \text{ is odd} \}$$

b. تنها یک گذار به این PDA اضافه کنید تا زبان L را بپذیرد.
در این بخش باید گذاری اضافه کنیم که PDA رشته های با طول زوج را نیز که شرط مورد نظر را دارند بپذیرد پس گذاری از q_2 به q_3 اضافه میکنیم که این بار سمبولی را به عنوان سمبول میانی چک نکند و تنها از q_2 که

n سمبول اول را پوش می کند به q_3 که سمبول انتهایی match میکند منتقل کند. در نتیجه به PDA زیر میرسیم که زبان مورد نظر L را میپذیرد:

