دانشگاه صنعتی شریف
دانشکده‌ی مهندسی کامپیوتر

نظریه زبان‌ها و ماشین‌ها

امیررضا آذری - ۹۹۱۰۱۰۸۷

پاسخ تمرین سوم

۱. **پرسش نخست**

پاسخ.

آ)

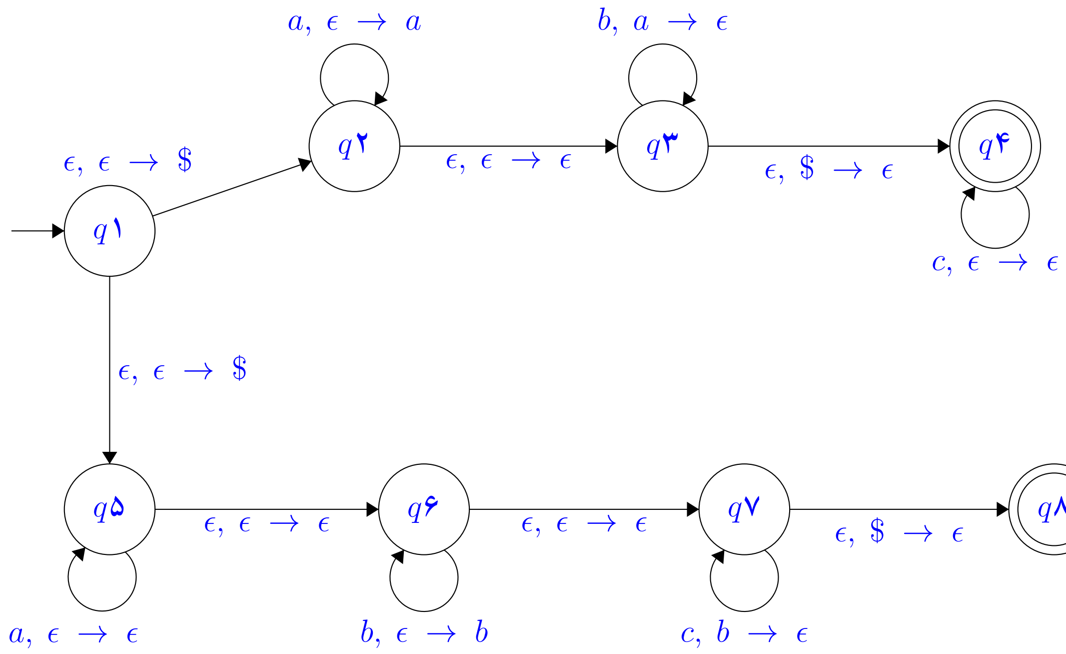
دستور زبان مستقل از متن برابر است با:

$$
\begin{aligned}
S &\rightarrow S_1|S_2\\
S_1 &\rightarrow S_1c|A|\epsilon\\
A &\rightarrow aAb|\epsilon\\
S_2 &\rightarrow aS_2|B|\epsilon\\
B &\rightarrow bBc|\epsilon
\end{aligned}
$$

دقت کنید که A رشته‌ای با تعداد برابر a و b می‌سازد. همچنین B نیز رشته‌ای با تعداد برابر b و c می‌سازد. حالت دیگر این گرامر را می‌توان به شکل زیر نوشت که با هم معادل هستند.

$$
\begin{aligned}
S &\rightarrow S_1C|AS_2\\
S_1 &\rightarrow aS_1b|\epsilon\\
A &\rightarrow aA|\epsilon\\
S_2 &\rightarrow bS_2c|\epsilon\\
C &\rightarrow cC|\epsilon
\end{aligned}
$$

خودکاره پشته‌ای متناظر به این زبان برابر است با:

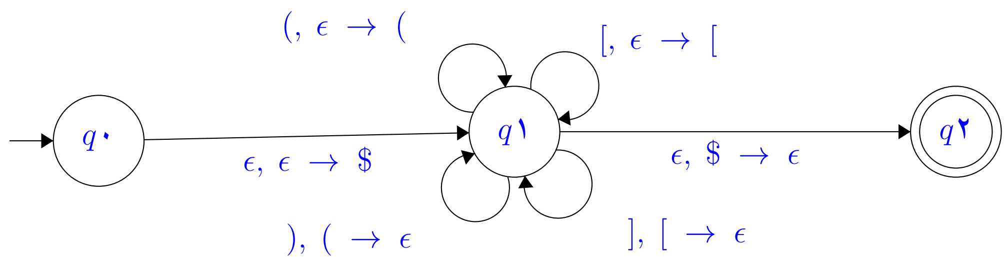


ب)

دستور زبان مستقل از متن برابر است با:

$$S \rightarrow SS \mid (S) \mid [S] \mid \epsilon$$

خودکاره پشته‌ای متناظر به این زبان برابر است با:



ج)

گرامر این زبان برابر است با:

$$S \rightarrow aSb \mid aaSb \mid aaSbbb \mid \epsilon$$

دقت کنید می‌خوهیم که $i \leq ۲j \leq ۳i$ باشد. قاعده اول تعداد برابر a و b تولید می‌کند که در زبان بیان شده است. قاعده دوم به ما کمک می‌کند تا رشته‌هایی که ۲ برابر تعداد b های آن با تعداد هایشa برابر است ایجاد بشود. قاعده سوم نیز بررسی می‌کند تا ۲ برابر تعداد b ها با ۳ برابر تعداد a ها برابر شود. حال با ترکیب این ۳ قاعده، به گرامر زبان مورد نظر می‌رسیم. چون تعداد b ها هیچگاه ۲ برابر تعداد a ها و یا کمتر از نصف آنها

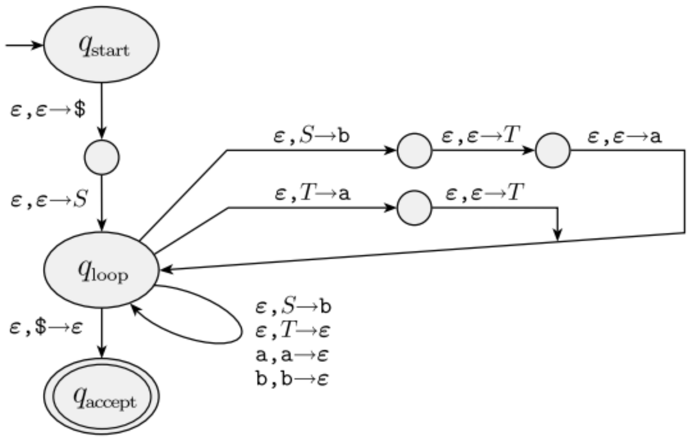
نمی‌شود.
حال طبق مثال زیر از سورس، گرامر را به pda تبدیل می‌کنیم.

EXAMPLE 2.25

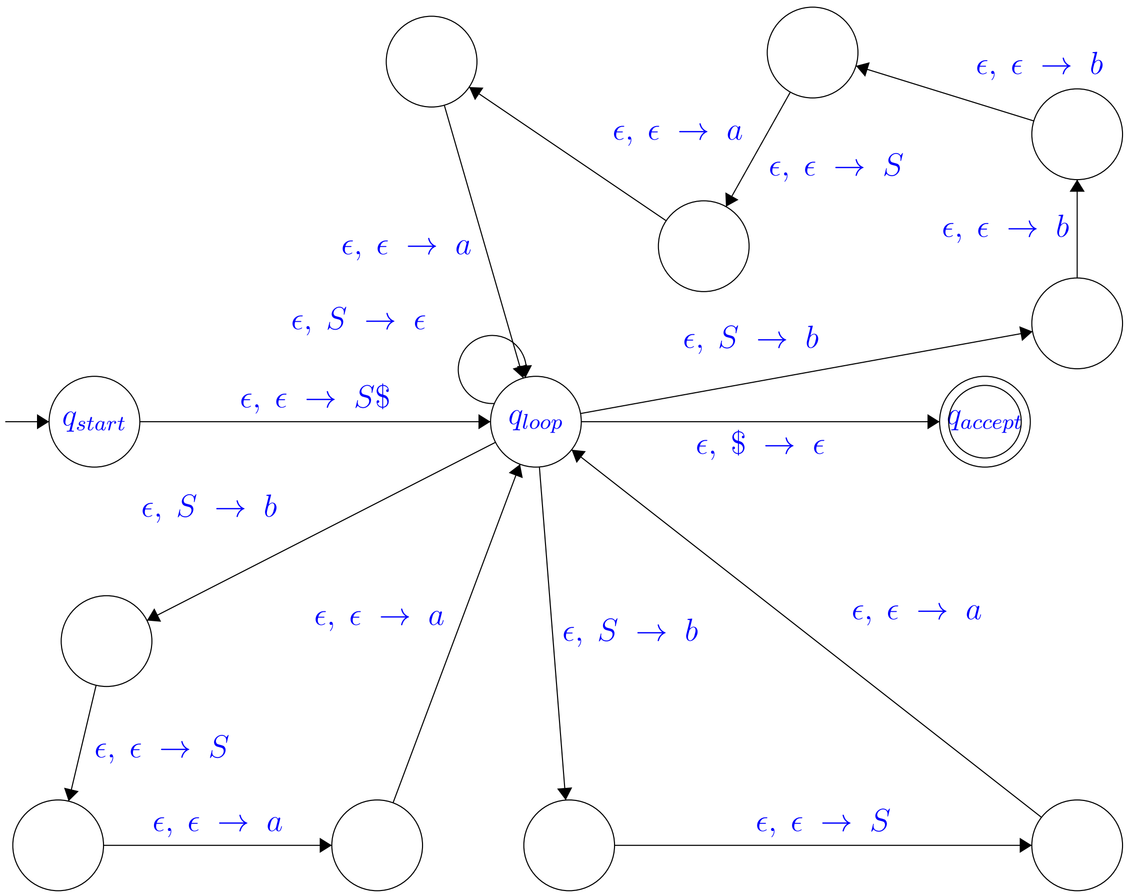
We use the procedure developed in Lemma 2.21 to construct a PDA P_1 from the following CFG G.

$$S \to \mathtt{a}T\mathtt{b} \mid \mathtt{b}$$
$$T \to T\mathtt{a} \mid \varepsilon$$

The transition function is shown in the following diagram.



داریم (دقت کنید مرحله پوش کردن $ و s با هم انجام شده است.) :



۲. **پرسش دوم**

پاسخ.

آ)

ماشین‌های پشته‌ای مربوطه، هر کدام یک زبان منظم را نمایندگی می‌کنند. هم‌چنین می‌دانیم که زبان‌های منظم زیرمجموعه‌ی محض زبان‌های مستقل از متن هستند. در نتیجه برای همه‌ی این زبان‌ها می‌توان یک ماشین پشته‌ای طراحی کرد.

می‌دانیم که در ماشین‌های متناهی، صرفا داشتن نماد ورودی و نیز حالت فعلی ماشین، برای تعیین حالت بعدی کفایت می‌کند. به همین جهت کافی است که از پشته، به همین منظور استفاده کنیم. یک روش آن است که به ازای هر حالت ماشین متناهی، یک نماد در نظر بگیریم. هر بار که ماشین متناهی به یک حالت جدید تغییر حالت می‌دهد، ماشین پشته‌ای که قرار است عملکرد آن را شبیه سازی کند، نماد مربوطه را بالای پشته پوش می‌کند.

حال براي تعيين حالت بعدی، به حالت قبلی که نماد آن در بالای پشته قرار گرفته دسترسی دارد و به همین جهت، می‌تواند برای نماد بعدی تصمیم گیری کند.

ماشین پشته‌ای مربوطه، برحسب اینکه شیوه‌ی پذیرش در آن چگونه باشد، می‌تواند حداکثر با دو حالت پیاده سازی گردد.

برای تعریف کامل داریم:

این ۲ حالت، حالت‌های پذیرش و رد هستند. طبیعتا حالت شروع همان حالت رد میباشد و حالت پذیرش، همان فاینال استیت یا استیت اکسپت است. این دو در الفبای استک نیز تفاوت دارند و شامل کاراکترهای s_i میباشد (به ازای هر حالت در DFA).

با توجه به خواسته سوال برای هر عضو الفبا باید انتقال $r, \epsilon \rightarrow \delta(s., r)$ در نظر بگیریم که r همان عضو الفباست. در هنگام شروع در حالت پذیرش هستیم پس اگر حاصل $(s., r)$ در فاینال استیتهای ما باشد، حالت خود را عوض میکنیم. حال به ازای هر حالت از DPDA و به ازای هر حرف الفبا و هر استیت ماشین، تمامی حالات گفته شده و حالت پذیرش یا رد را در نظر می‌گیریم. بنابراین نشان دادیم که براي هر زبان منظم، يك خودکاره پشته اي قطعي دو حالته موجود است که هیچ گذر ϵ ندارد و هرگز نمادي را از پشته حذف نمي كند.

ب)

ابتدا ۲ عکس از راه های حل شده در کتاب سیپسر را نشان می‌دهیم.

We show that an ambiguous CFG cannot be a DCFG. If G is ambiguous, G derives some string s with at least two different parse trees and therefore s has at least two different rightmost derivations and at least two different leftmost reductions. Compare the steps of two of these different leftmost reductions and locate the first string where these reduction differ. The preceding string must be the same in both reductions, but it must have two different handles. Hence G is not a DCFG.

Step 1 of 1

An **ambiguous grammar** is defined as “**a context free grammar** for which there subsists a string and that string may contains greater than one left-most derivation. **An unambiguous grammar** is defined as “a **context free grammar** for which all **justifiable string** has an individual leftmost derivation.

• As context free grammar (CFG’s) is proper superset of deterministic context-free grammars (DCFG’s). It can be derived from deterministic finite automata and it can be used to generate deterministic context free language.

• DCFG’s (deterministic context-free grammars) are always shows an unambiguous behavior and an unambiguous context free grammar (CFG’s) is an important super class of DCFG’s.

The above statement can be proved by the following way:

• As it is known that **for every pushed down automata** M **there exist an equivalent context free grammar** G**.**

Therefore, M **Recognizes** $L\$ \Rightarrow$ G **generates** $L\$$. But, M **deterministic** $\Rightarrow$ G **is an unambiguous**. Hence, **replacing** $\$$ **by** ε **in** $G \Rightarrow G$ **generates** L .

• Thus, from the above explanation it can be said that **every DCFG is an unambiguous CFG.**

حال ۲ قضیه را نشان می‌دهیم.

.۱

زبان L برای $DPDA$ ای مانند P، یک $L^\epsilon(P)$ هست، اگر و تنها اگر که L خاصیت prefix را داشته باشد و برای DPDA ای مانند P' داشته باشم: $L = L(P')$.

ابتدا طرف رفت را نشان می‌دهیم:

۲ رشته مختلف x, y را از زبان L در نظر بگیرید به طوری که x درواقع prefix رشته y باشد. P بعد از x گیر خواهد کرد و راهی برای پذیرش و اکسپت y وجود نخواهد داشت که تناقض است.

طرف بازگشت:

از استیت اکسپت نباید انتقال معناداری وجود داشته باشد. بنابراین، ما میتونیم آن را حذف کنیم. اثبات کاملتر را مشاهده می‌کنید:

Theorem 18.2

A language L is $L^\epsilon(P)$ for some DPDA P iff L has the prefix property and $L = L(P')$ for some DPDA P'.

Proof.

$(\Rightarrow)$

Let distinct $x, y \in L$ such that x is prefix of y.

P gets stuck after x. no way to accept y. Contradiction.

The usual construction for “by final state” automaton preserves determinism.

$(\Leftarrow)$

There must be no useful outgoing transitions from accepting states.

Therefore, we can delete them if there are any.

Now we can use the usual construction to obtain “by empty stack” DPDA P from P'. $\square$

.۲

اگر زبان L داشته باشیم که $L = l^\epsilon(P)$ برای $DPDA$ای مانند P باشد؛ آنگاه L یک گرامر غیرمبهم دارد.

فرض کنید G گرامر غیرمبهم است. همچنین رشته w در زبان L را در نظر بگیرید.

دقیقاً یک اجرای P روی w وجود دارد. بنابراین، در هر مرحله حداقل یکی از قوانین تولید را می توان در سمت چپ ترین نماد اعمال کرد. به طور کامل توضیح این بخش را در عکس زیر مشاهده می‌نمایید:

> Theorem 18.3
> *If language $L = L^{\epsilon}(P)$ for some DPDA P, then L has an unambiguous grammar.*
>
> Proof.
> Due to theorem 18.1, we construct a grammar that G such that $L(G) = L$.
>
> **claim:** G is unambiguous
> Consider $w \in L$.
> There is exactly one run of P on w.
> Therefore, at each step at least one of the production rules can be applied on the leftmost symbol.(why?)
> At some step, let it be a rule due to the transition $\delta(q, a, X) = \{(r, Y_1 ... Y_k)\}$.
> There may be many production rules generated due the above transition.
> Since there is a unique run, therefore Y_i must be popped at unique stater_i.
> Therefore, exactly one of the production rule will lead to acceptance. □

حال به بخش اصلی می‌رویم که میخواهیم نشان بدهیم اگر زبان L يك خودكاره پشته اي قطعي داشته باشد، آنگاه يك دستور زبان نامبهم براي آن موجود است.

ابتدا تعریف زیر را داریم:

> Prefix property
>
> Definition 18.2
> *A language L has prefix property if there are no two $x, y \in L$ such that x is a prefix of y.*

حال فرض کنید \$ یک سیمبل در L داریم. زبان $L\$$ را در نظر بگیرید. این زبان خاصیت prefix را دارا است.

حال برای یک $DPDA$ مانند P'، $L\$ = L(P')$.

طبق بخش اول، یک $DPFA$ مانند D وجود دارد که $L\$ = L^{\epsilon}(D)$.

طبق بخش دوم یک گرامر نامبهم مانند G' وجود دارد که $L(G') = L\$$.

به دست آوردیم که G در واقع \$ را به غیرپایانه تبدیل کرده و یک پروداکشن $\$ \rightarrow \epsilon$ در G' خواهد گذاشت. مشخصا $L(G) = L$.

هر lmdهای در G، قاعده $\$ \rightarrow \epsilon$ را در اخر اعمال خواهد کرد. بقیه مشتقات مانند G' خواهند بود. بنابراین انتخاب دیگری در میانه مشتق‌گیری در G نمی‌ماند و دقیقا یک حالت مشتق برای \$ وجود خواهد داشت.

بتابراین G غیرمبهم است.

۳. پرسش سوم

پاسخ.

در این سوال، با توجه به الگوریتم تبدیل به فرم نرمال چامسکی و گریباخ عمل می‌نماییم. برای تبدیل به فرم نرمال چامسکی، ابتدا یک قاعده ابتدایی $S. \to S$ اضافه می‌نماییم. سپس طبق گفته‌های سورس عمل خواهیم کرد. داریم:

DEFINITION 2.8

A context-free grammar is in ***Chomsky normal form*** if every rule is of the form

$$A \to BC$$
$$A \to a$$

where a is any terminal and A, B, and C are any variables—except that B and C may not be the start variable. In addition, we permit the rule $S \to \varepsilon$, where S is the start variable.

PROOF First, we add a new start variable S_0 and the rule $S_0 \to S$, where S was the original start variable. This change guarantees that the start variable doesn't occur on the right-hand side of a rule.

Second, we take care of all ε-rules. We remove an ε-rule $A \to \varepsilon$, where A is not the start variable. Then for each occurrence of an A on the right-hand side of a rule, we add a new rule with that occurrence deleted. In other words, if $R \to uAv$ is a rule in which u and v are strings of variables and terminals, we add rule $R \to uv$. We do so for each *occurrence* of an A, so the rule $R \to uAvAw$ causes us to add $R \to uvAw$, $R \to uAvw$, and $R \to uvw$. If we have the rule $R \to A$, we add $R \to \varepsilon$ unless we had previously removed the rule $R \to \varepsilon$. We repeat these steps until we eliminate all ε-rules not involving the start variable.

Third, we handle all unit rules. We remove a unit rule $A \to B$. Then, whenever a rule $B \to u$ appears, we add the rule $A \to u$ unless this was a unit rule previously removed. As before, u is a string of variables and terminals. We repeat these steps until we eliminate all unit rules.

Finally, we convert all remaining rules into the proper form. We replace each rule $A \to u_1u_2 \cdots u_k$, where $k \geq 3$ and each u_i is a variable or terminal symbol,

with the rules $A \to u_1A_1$, $A_1 \to u_2A_2$, $A_2 \to u_3A_3$, ..., and $A_{k-2} \to u_{k-1}u_k$. The A_i's are new variables. We replace any terminal u_i in the preceding rule(s) with the new variable U_i and add the rule $U_i \to u_i$.

برای تبدیل به فرم گریباخ نیز داریم:

Greibach Normal Form

Another useful grammatical form is the **Greibach normal form**. Here we put restrictions not on the length of the right sides of a production, but on the positions in which terminals and variables can appear. Arguments justifying Greibach normal form are a little complicated and not very transparent. Similarly, constructing a grammar in Greibach normal form equivalent to a given context-free grammar is tedious. We therefore deal with this matter very briefly. Nevertheless, Greibach normal form has many theoretical and practical consequences.

DEFINITION 6.5

A context-free grammar is said to be in Greibach normal form if all productions have the form $A \to ax$,
where $a \in T$ and $x \in V^*$.

با توجه به گفته هد درس تنها به پاسخ نهایی هر بخش اکتفا می‌نماییم. همچنین گفته شده که اپسیلون را می‌توانیم در فرم نهایی ننویسیم.

۱.

چامسکی:

$$
\begin{aligned}
S_. &\to AB \mid U_{S_.}B \mid \epsilon \\
S &\to U_{S_.}B \mid AB \\
A &\to U_aU_{S_.} \mid U_aA \mid a \\
B &\to SU_S \mid U_bU_b \mid U_bS \mid SU_b \mid U_aU_{S_.} \mid U_aA \mid a \mid b \\
U_{S_.} &\to AS \\
U_S &\to U_bS \\
U_a &\to a \\
U_b &\to b
\end{aligned}
$$

گریباخ:

برای این بخش از فرم نرمال چامسکی مرحله قبل کمک می‌گیریم:

$$
\begin{array}{rcl}
S_. & \rightarrow & aU_{S_.}B \mid aB \mid aAB \mid aU_{S_.}SB \mid aSB \mid aASB \mid \epsilon \\
S & \rightarrow & aU_{S_.}B \mid aB \mid aAB \mid aU_{S_.}SB \mid aSB \mid aASB \\
A & \rightarrow & aU_{S_.} \mid aA \mid a \\
B & \rightarrow & aU_{S_.}BU_S \mid aBU_S \mid aABU_S \mid aU_{S_.}SBU_S \mid aSBU_S \mid aASBU_S \\
\mid & aU_{S_.}BU_b & \mid aBU_b \mid aABU_b \mid aU_{S_.}SBU_b \mid aSBU_b \mid aASBU_b \mid bU_b \mid bS \mid b \mid aU_{S_.} \mid a \mid aA \\
U_{S_.} & \rightarrow & aU_{S_.}S \mid aS \mid aAS \\
U_S & \rightarrow & bS \\
U_a & \rightarrow & a \\
U_b & \rightarrow & b
\end{array}
$$

دقت کنید که می‌توانستیم برای تبدیل به فرم گریباخ از گرامر اولیه شروع کنیم و دیگر $S_.$ را ننویسیم اما برای راحتی از فرم نرمال چامسکی به دست آمده بهره جستیم. همچنین U_a به نوعی useless است.

۲.

چامسکی:

در این مورد دقت کنید بعد حذف اپسیلون، تعداد زیادی unit داریم. برای مثال، A به C می‌رود و C هم به S می‌رود. پس می‌توان نتیجه گیری کرد که A به S می‌رود و C را حذف کرد. همچنین وقتی A تنها به S می‌رود، می‌توان تمامی Aهای سمت راست را با S جایگزین کرد. همین استدلال را برای B نیز خواهیم داشت.

$$
\begin{array}{rcl}
S_. & \rightarrow & U_۱U_a \mid U_۲U_b \mid SS \mid U_aU_a \mid U_bU_b \mid \epsilon \\
S & \rightarrow & U_۱U_a \mid U_۲U_b \mid SS \mid U_aU_a \mid U_bU_b \\
U_a & \rightarrow & a \\
U_b & \rightarrow & b \\
U_۱ & \rightarrow & U_aS \\
U_۲ & \rightarrow & U_bS
\end{array}
$$

گریباخ:

برای این بخش به دلیل داشتن قاعده $S \rightarrow SS$ باید left recursion را برطرف نماییم.

$$
\begin{array}{rcl}
S_\circ & \rightarrow & aSU_a \mid bSU_b \mid aU_a \mid bU_b \mid aSU_aS \mid bSU_bS \mid aU_aS \mid bU_bS \\
\mid aSU_aFS & \mid & bSU_bFS \mid aU_aFS \mid bU_bFS \mid \epsilon \\
S & \rightarrow & aSU_a \mid bSU_b \mid aU_a \mid bU_b \mid aSU_aF \mid bSU_bF \mid aU_aF \mid bU_bF \\
F & \rightarrow & aSU_a \mid bSU_b \mid aU_a \mid bU_b \mid aSU_aF \mid bSU_bF \mid aU_aF \mid bU_bF \\
\mid aSU_aFF & \mid & bSU_bFF \mid aU_aFF \mid bU_bFF \\
U_a & \rightarrow & a \\
U_b & \rightarrow & b \\
U_۱ & \rightarrow & aS \\
U_۲ & \rightarrow & bS
\end{array}
$$

که در اینجا متغیرهای $U_۱, U_۲$ به نوعی useless هستند.

۳.

چامسکی:

دقت کنید چون در سمت راست هیچ قاعده‌ای S نداریم، می‌توانیم قاعده ابتدایی $S_\circ \rightarrow S$ را اضافه نکنیم.

$$
\begin{array}{rcl}
S & \rightarrow & UA \mid AA \mid U_aA \mid a \\
A & \rightarrow & U_aA \mid a \\
U & \rightarrow & AA \\
U_a & \rightarrow & a
\end{array}
$$

گریباخ:

$$
\begin{array}{rcl}
S & \rightarrow & aAAA \mid aAA \mid aA \mid a \\
A & \rightarrow & aA \mid a \\
U & \rightarrow & aAA \mid aA \\
U_a & \rightarrow & a
\end{array}
$$

که در اینجا متغیرهای U_a, U به نوعی useless هستند.

۴. **پرسش چهارم**

پاسخ.

آ)

مبهم است. رشته aab را در نظر بگیرید:

$$S \to aS \to aaSbS \to aabS \to aab$$
$$S \to aSbS \to aaSbS \to aabS \to aab$$

ب)

مبهم نیست زیرا رشته‌ای یافت نمی‌شود که به ۲ شکل متفاوت مشتق بشود.

ج)

مبهم نیست زیرا رشته‌ای یافت نمی‌شود که به ۲ شکل متفاوت مشتق بشود.
(طبق گفته هد، صرفا اینکه اشاره کنیم مبهم نیست کافی می‌باشد.)

۵. **پرسش پنجم**

پاسخ.

۱.

در تمامی موارد، برهان خلف می‌زنیم و فرض می‌کنیم زبان مستقل از متن است. اما در ادامه به تناقض می‌رسیم.
همچنین در مورد لم تزریق می‌دانیم:

Pumping lemma for context-free languages If A is a context-free language, then there is a number p (the pumping length) where, if s is any string in A of length at least p, then s may be divided into five pieces $s = uvxyz$ satisfying the conditions

1. for each $i \geq 0$, $uv^ixy^iz \in A$,
2. $|vy| > 0$, and
3. $|vxy| \leq p$.

آ)

رشته $a^pb^{p+1}c^{p+2}$ را در نظر بگیرید. حال vy را در نظر بگیرید. اگر تعداد تکرار حرف a در آن بیشتر از b باشد، پامگ به بالا باعث می‌شود رشته جدید عضو زبان نباشد. اگر هم کمتر باشد، پامپ به پایین سبب می‌شود دیگر رشته جدید عضو زبان مد نظر نباشد. بنابراین حتما باید تعداد تکرار a و b در vy برابر باشد. همین استدلال را برای b و c و همینطور a و c می‌توان داشت. درنتیجه در رشته vy تعداد حروف تکرار شده a و b و c باید برابر باشند.
طبق لم تزریق می‌دانیم اندازه رشته vy بزرگتر از صفر می‌باشد. طبق توضیحات نتیجه میگیریم این رشته از تمامی ۳ حروف ما، حداقل یکی را دارا هستند. بنابراین حتی اگر یکی از v یا y تنها شامل یک حرف باشند، دیگری

شامل دو کاراکتر مختلف است. در این حالت به طور مشخص پامپ به بالا، سبب می‌شود ترتیب کاراکترها به هم بریزد و رشته ساخته شده عضو زبان نیست. پس به تناقض رسیدیم.

تصویر زیر راه دیگری است:

1.A: consider $a^{3m}b^{3(m+1)}c^{3(m+2)}$ for pumping lemma with length p.
s = uvxyz, for any $i \geq 0 : uv^ixy^iz$ is in L, $|vy| > 0$, $|vxy| \leq p$
suppose $u = a^m, v = a^{2m}, x = b^{3(m+1)}, y = c^{m+2}, z = b^{2(m+2)}$
however: if $i = 2$, we have: $a^{5m}b^{3(m+1)}c^{4(m+2)}$, and for any m in N we do not have: $5m < 4(m+1)$, e.g. $m = 5$. So, the the condition: "for any $i \geq 0 : uv^ixy^iz$ is in L" is violated. Therefore, L is not CF.

ب)

رشته $a^pb^{p^۲}$ را در نظر بگیرید. اگر vy تنها از a تشکیل شده باشند، داریم:

فرض کنید $|vy| = a^k$. بنابراین $uv^ixy^iz = a^{p+(i-۱)k}$. حال به ازای $i = ۲$ داریم:

$$uv^۲xy^z = a^{p+k}$$
$$p+k \ \leq p+p = ۲p \leq p^۲ + ۲p + ۱ = (p+۱)^۲$$
$$\rightarrow p+k \ < \ (p+۱)^۲$$

بنابراین رشته $uv^۲xy^۲z$ داخل این زبان نیست زیرا از کوچک‌ترین مربع بعد از $p^۲$ کمتر است و با آن برابر نیست. بنابراین چون $k > ۰$، مقدار $(p+۱)^۲$ از $(p+k)^۲$ نیز کوچکتر مساوی خواهد بود. پس به تناقض رسیدیم.

بنابراین رشته vy حتما شامل b هست. اگر هر کدام از رشته‌های y یا v شامل هر دو کاراکتر باشند، در i های بالاتر به وضوح ترتیب کلمات به هم خواهد ریخت و عضو زبان نخواهد بود. بنابراین یا هردو فقط شامل b خواهند بود و یا v تنها شامل a و y تنها شامل b خواهد بود. در حالتی که $vy = b^k$، رشته‌ی $uv^۲xy^۲z$ به طور مشخص عضو زبان نخواهد شد زیرا $a^pb^{p^۲+k}$ که توان مربعی از p نیست.

حال برای حالت آخر باقی مانده داریم:

$$|v| = q, \ |y| = w$$
$$\rightarrow uv^۲xy^۲z = a^{p+q}b^{p^۲+w}$$
$$\xrightarrow{q=۰} p^۲ \neq p^۲ + w$$
$$\xrightarrow{w=۰} (p+q)^۲ \neq p^۲$$
$$\xrightarrow{otherwise} (p+q)^۲ \geq (p+۱)^۲ > p^۲ + ۲p > p^۲ + w$$

که در تمامی حالات به تناقض رسیدیم.

ج)

ابتدا به طور واضح می‌دانیم هر رشته‌ای که از این زبان بگیریم، رشته vy شامل \$ نخواهد بود. زیرا با پامپ بالا یا پایین به هم می‌ریزد. همچنین رشته vy به طور کامل در سمت راست یا چپ \$ نیز قرار ندارد زیرا با پامپ بالا یا پایین، مقادیر را می‌توان بسیار کم یا زیاد کرد که به هم می‌ریزد. همچنین طبق $|vxy| \leq p$، نتیجه می‌شود

تنها حالت ممکن تا اینجا این است که v شامل فقط ۰ و y شامل فقط ۱ باشد. در ضمن رشته را 1^p0^p در نظر بگیرید. و طبیعتا v در سمت چپ \$ و y در سمت راست آ» است. پس برای uv^2xy^2z برای بخش چپ \$ داریم $1^p0^{p+|v|}$. و برای بخش راست داریم $1^{p+|y|}0^{p-1}1$. زیرا سمت راست برابر $b(n+1)$ است. بنابراین داریم:

$$1^p0^{p+|v|} + 1 \;=\; 1^{p+|y|}0^{p-1}1$$
$$\rightarrow 1^p0^{p+|v|-1}1 = 1^{p+|y|}0^{p-1}1$$

از معادلات بالا نتیجه می‌شود که $0 = |y| = |v|$ که طبق ویژگی لم تزریق، به تناقض رسیدیم. تصویر زیر را نیز می‌توان به عنوان پاسخ دیگر دید:

1.C: We know, for a number like A, we need $[\log a] + 1$ bits. So, with increasing the n to $n+1$ we have two forms for the language L:
1. 1(some 0 and 1 with length of $[\log a]$)\$1(some 0 and 1 with length of $[\log a]$) (e.g. $n = 7 \rightarrow 5 = 101, 6 = 110$.
2. 1(some 0 and 1 with length of $[\log a]$)\$1(some 0 and 1 with length of $[\log a] + 1$) (e.g. $n = 7 \rightarrow 7 = 111, 8 = 1000$.
Now, we prove that the first format in not CF. So, we conclude that L is not CF: suppose $u = 1000, v = 100, x = \$1, y = 000, z = 101$
however: if $i = 2$, we have: 1000100100\$1000000101, and obviously, s is not in L. So, the the condition: "for any $i \geq 0 : uv^ixy^iz$ is in L" is violated. Therefore, L is not CF.

برای اثبات این بخش، از لینک‌های زیر کمک گرفته شده است.

- لینک اول
- لینک دوم
- لینک سوم

د)

رشته $a^pb^pc^p$ را در نظر بگیرید. اگر رشته vy شامل b باشد، با پامپ به پایین تعداد b ها کمتر از max می‌شود و این تناقض است. پس b ندارند. همچنین هردوی a یا c را هم ندارند زیرا می‌دانیم $|vxy| \leq p$

پس هر دو یا فقط a دارند یا فقط c. در این حالت نیز با پامپ به بالا، مقدار max از تعداد b بیشتر خواهد شد و تناقض است. بنابریان زبان مستقل از متن نیست. تصویر زیر راه دیگری است:

1.D: consider $a^{m+1}b^{m+1}c^{k+1}$ and we know: $m > k$ for pumping lemma with length p.
s = uvxyz, for any $i \geq 0 : uv^ixy^iz$ is in L, $|vy| > 0$, $|vxy| \leq p$
suppose $u = a^m, v = ab, x = b^{m-1}, y = bc, z = c^k$
however: if $i = 2$, we have: $a^{m+3}b^{m+5}c^{m+3}$. Because $m > k$, So $m + 5 \neq m + 3 = max(m + 3, k + 3)$ So, the the condition: "for any $i \geq 0 : uv^ixy^iz$ is in L" is violated. Therefore, L is not CF.

.۲

آ)

این را در اسلایدهای درس نیز اثبات کردیم.

Theorem

CFLs are closed under reversal.

Proof.

Consider grammar $G = (V, \Sigma, R, S)$ where $B = L(G)$. Construct CFG G' where

$$R' = \{A \to u^{\mathcal{R}} \mid A \to \text{ is a rule in } R\}.$$

Show that each for variable $A \in V$ and $u \in (V \cup \Sigma)^*$

$$A \overset{n}{\Rightarrow}_G u \leftrightarrow A \overset{n}{\Rightarrow}_{G'} u^{\mathcal{R}}.$$

$\overset{n}{\Rightarrow}$ means $\overset{*}{\Rightarrow}$ in exactly k steps. □

Proof Cont.

- Base case $n = 0$: $A \overset{0}{\Rightarrow}_G u$, then $u = A = u^{\mathcal{R}}$. So $A \overset{0}{\Rightarrow}_{G'} u^{\mathcal{R}}$ and vice versa.

□

Proof Cont.

- Inductive step: Assume for all $n > 0$, $A \in V$ and $u \in (V \cup \Sigma)^*$:

$$A \overset{n-1}{\Longrightarrow}_G u \leftrightarrow A \overset{n-1}{\Longrightarrow}_{G'} u^{\mathcal{R}}.$$

- If $A \overset{n}{\Rightarrow}_G u$, then there is $C \in V$ and $x, y, z \in (V \cup \Sigma)^*$ such that $u = xyz$, $A \overset{n-1}{\Longrightarrow}_G xCz$, and $C \Rightarrow_G y$.
- By induction hypothesis: $A \overset{n-1}{\Longrightarrow}_{G'} z^{\mathcal{R}}Cx^{\mathcal{R}}$.
- By construction $C \Rightarrow_{G'} y^{\mathcal{R}}$.
- Thus, $A \overset{n}{\Rightarrow}_{G'} z^{\mathcal{R}}y^{\mathcal{R}}x^{\mathcal{R}} = (xyz)^{\mathcal{R}} = u^{\mathcal{R}}$.
- Repeat for the reverse direction. Then:

$$A \overset{n}{\Rightarrow}_G u \leftrightarrow A \overset{n}{\Rightarrow}_{G'} u^{\mathcal{R}}.$$

Proof Cont.

- Therefore, for $w \in B$

$$S \overset{*}{\Rightarrow}_G w \leftrightarrow S \overset{*}{\Rightarrow}_{G'} w^{\mathcal{R}},$$

which implies $L(G') = B^{\mathcal{R}}$

ب)

این را در اسلایدهای درس نیز اثبات کردیم.

Intersection with a Regular

Theorem

CFLs are closed under intersection with a regular language.

Proof.

- A a CFL recognized by the PDA $M_1 = (Q_1, \Sigma, \Gamma, \delta_1, q_1, F_1)$
- B a regular language recognized by the NFA $M_2 = (Q_2, \Sigma, \Gamma, \delta_2, q_2, F_2)$
- Construct PDA $M = (Q, \Sigma, \Gamma, \delta, q_0, F)$
  - $Q = Q_1 \times Q_2$
  - $q_0 = (q_1, q_2)$
  - $F = F_1 \times F_2$
  - The transition function for all $a \in \Sigma_\varepsilon, b, c \in \Gamma_\varepsilon$:

$$\delta((q,r),a,b) = \{((s,t),c) \mid (s,c) \in \delta_1 \text{ and } t \in \delta_2(r,a)\},$$

  - As M runs on input w, its stack and the first element of its state change according to q_1 whereas the second element of its state changes according to q_2.

$\square$

همچنین راه زیر را هم داریم(بخش a)

(a) Let C be a context-free language and R be a regular language. Let P be the PDA that recognizes C, and D be the DFA that recognizes R. If Q is the set of states of P and Q' is the set of states of D, we construct a PDA P' that recognizes $C \cap R$ with the set of states $Q \times Q'$. P' will do what P does and also keep track of the states of D. It accepts a string w if and only if it stops at a state $q \in F_P \times F_D$, where F_P is the set of accept states of P and F_D is the set of accept states of D. Since $C \cap R$ is recognized by P', it is context free.

(b) Let R be the regular language $\mathtt{a}^*\mathtt{b}^*\mathtt{c}^*$. If A were a CFL then $A \cap R$ would be a CFL by part (a). However, $A \cap R = \{\mathtt{a}^n\mathtt{b}^n\mathtt{c}^n \mid n \geq 0\}$, and Example 2.36 proves that $A \cap R$ is not context free. Thus A is not a CFL.

ج)

برای حل این سوال از این لینک کمک گرفته‌ایم.

در این راه G که گرامر L است و h که یک homomorphism است را در نظر بگیرید. حال یک گرامر برای $h(L)$ ارائه می‌دهیم.

$$G = (V, \Sigma, R, S),\ \ h:\ \Sigma^* \to \Gamma^*$$
$$G' = (V', \Gamma, R', S'),\ add\ V\ to\ V',\ \ S = S'$$

حال تمامی ترمینال های G را متغیر در نظر می‌گیریم. برای هر $a \in \Sigma$ داریم: یک سری متغیر به اسم U_a تولید میکنیم و انها را به V' اضافه مینماییم. انگاه برای هر عضو R، اگر a در rhs ان قاعده امده بود، ان را با U_a جایگزین می‌کنیم. همچنین برای هر U_a، قاعده $h(a)$ را اضافه مینماییم. اکنون در این دستور زبان جدید، هر

رشته ساخته شده الحاقی از U_a ما است که رشته های نگاشت شده ما همانطور که می خواستیم هستند. همچنین برای هر رشته نگاشت شده، فقط رشته اصلی را در نظر بگیرید که آن را به L می سازد و همین کار را در گرامر مستقل از متن ما انجام دهید. رشته را با فرمت U_a خواهید داشت. حالا وقتی آخرین قانون را انجام می دهید، آنها را به فضای نگاشت شده تبدیل می کنید که homomorphism ما است. بنابراین هر دو طرف کار می کنند. در ادامه داریم:

Let $M = (Q, \Delta, \Gamma, \delta, q_0, F)$ be the pda of our context free language (L). The homomorphism is $h : \Sigma^* \to \Delta^*$. Also set $n = max_{a\in\Sigma} \mid h(a) \mid$ be the maximum length of the mapped version of any of our symbols. Let $N = (Q', \Sigma, \Gamma, \delta', q_0', F')$ such that
$Q' = Q \times \Delta^{\leq n}$ which I mean all strings with a length og at most n.
$q_0' = (q_0, \epsilon)$, $F' = F \times \{\epsilon\}$, and

$$\delta'((q,v),x,a) = \begin{cases} \{((q,h(x)),\epsilon)\} & \text{if } v = a = \epsilon \\ \{((p,u,b) \mid (p,b) \in \delta(q,y,a)\} & \text{if } v = yu, x = \epsilon, y \in (\Delta \cup \{\epsilon\}) \end{cases}$$

Also $\delta'() = \emptyset$ for other cases.

حال برای هر w در Σ^* داریم:

$$(q', \epsilon) \to_{P'}^{w'} ((q,v),\sigma) \equiv (q_., \epsilon) \to_{P'}^{w'} (q,\sigma),\ where\ h(w) = w'v.$$
$$\to w \in L(P') \equiv (q'_., \epsilon) \to_{P'}^{w'} ((q,\epsilon),\sigma)\ where\ q \in F.$$
$$\to \equiv (q_., \epsilon \to_{P}^{h(w)} (q,\sigma) \equiv h(w) \in L(P)$$
$$L(P') = h^{-۱}(L(P)) = h^{-۱}(L)$$

راه لینکی که ذکر شد برابر است با:

Inverse Homomorphisms

Recall: For a homomorphism h, $h^{-1}(L) = \{w \mid h(w) \in L\}$

Proposition 8. *If L is a CFL then $h^{-1}(L)$ is a CFL*

Proof Idea
For regular language L: the DFA for $h^{-1}(L)$ on reading a symbol a, simulated the DFA for L on $h(a)$. Can we do the same with PDAs?

- Key idea: store $h(a)$ in a "buffer" and process symbols from $h(a)$ one at a time (according to the transition function of the original PDA), and the next input symbol is processed only after the "buffer" has been emptied.
- Where to store this "buffer"? In the state of the new PDA!

Proof. Let $P = (Q, \Delta, \Gamma, \delta, q_0, F)$ be a PDA such that $L(P) = L$. Let $h : \Sigma^* \to \Delta^*$ be a homomorphism such that $n = \max_{a\in\Sigma} |h(a)|$, i.e., every symbol of Σ is mapped to a string under h of length at most n. Consider the PDA $P' = (Q', \Sigma, \Gamma, \delta', q'_0, F')$ where

- $Q' = Q \times \Delta^{\leq n}$, where $\Delta^{\leq n}$ is the collection of all strings of length at most n over Δ.
- $q'_0 = (q_0, \epsilon)$
- $F' = F \times \{\epsilon\}$
- δ' is given by

$$\delta'((q, v), x, a) = \begin{cases} \{((q, h(x)), \epsilon)\} & \text{if } v = a = \epsilon \\ \{((p, u), b) \mid (p, b) \in \delta(q, y, a)\} & \text{if } v = yu,\ x = \epsilon, \text{ and } y \in \Delta \end{cases}$$

  and $\delta'(\cdot) = \emptyset$ in all other cases.

We can show by induction that for every $w \in \Sigma^*$

$$\langle q'_0, \epsilon\rangle \xrightarrow{w}_{P'} \langle (q, v), \sigma\rangle \text{ iff } \langle q_0, \epsilon\rangle \xrightarrow{w'}_{P} \langle q, \sigma\rangle$$

where $h(w) = w'v$. Again this induction proof is left as an exercise. Now, $w \in L(P')$ iff $\langle q'_0, \epsilon\rangle \xrightarrow{w}_{P'} \langle (q, \epsilon), \sigma\rangle$ where $q \in F$ (by definition of PDA acceptance and F') iff $\langle q_0, \epsilon\rangle \xrightarrow{h(w)}_{P} \langle q, \sigma\rangle$ (by exercise) iff $h(w) \in L(P)$ (by definition of PDA acceptance). Thus, $L(P') = h^{-1}(L(P)) = h^{-1}(L)$. □

۶. پرسش امتیازی

پاسخ.

۱.

برای حل این سوال داریم:

$$
\begin{aligned}
G &= (V,\ \Sigma,\ R,\ S)\\
V &= \{S\}\\
\Sigma &= \{۰,\ ۱,\ \epsilon,\ (,\ ),\ +,\ *,\ \emptyset\}\\
R &= \{\\
S &\rightarrow SS \mid S+S \mid S^* \mid (S)\\
S &\rightarrow \emptyset\\
S &\rightarrow \epsilon\\
S &\rightarrow ۰\\
S &\rightarrow ۱\\
\}
\end{aligned}
$$

۲.

بله مبهم است. برای مثال رشته ۰ + ۰ + ۰ به ۲ حالت تولید می‌شود. برای رفع ابهام قواعد را به شکل زیر تغییر می‌دهیم:

$$
\begin{aligned}
S &\rightarrow S+E \mid E\\
E &\rightarrow EF \mid F\\
F &\rightarrow (S) \mid S^* \mid ۰ \mid ۱ \mid \emptyset \mid \epsilon
\end{aligned}
$$

در واقع برای رفع ابهام در این حالت، به نوعی اولویت‌بندی انجام داده‌ایم و اجازه ندادیم تا مبهم بماند.

موفق باشید :)