بسم اللّه الرّحمن الرّحیم

دانشگاه صنعتی اصفهان ـ دانشکدهٔ مهندسی برق و کامپیوتر
(نیم‌سال تحصیلی ۴۰۲۱)

نظریهٔ زبان‌ها و ماشین‌ها

حسین فلسفین

Leftmost and Rightmost Derivations

A derivation of a string w in a grammar G is a leftmost derivation if at every step the leftmost remaining variable is the one replaced. In other words, a leftmost derivation is a derivation in which the variable replaced at each step is the leftmost one. A rightmost derivation is defined analogously.

Example: ***In the grammar G with productions***

$$S \to a|S + S|S * S|(S),$$

the string $a + a * a$ has the following two leftmost derivations:

$$S \Rightarrow S + S \Rightarrow a + S \Rightarrow a + S * S \Rightarrow a + a * S \Rightarrow a + a * a$$

$$S \Rightarrow S * S \Rightarrow S + S * S \Rightarrow a + S * S \Rightarrow a + a * S \Rightarrow a + a * a$$

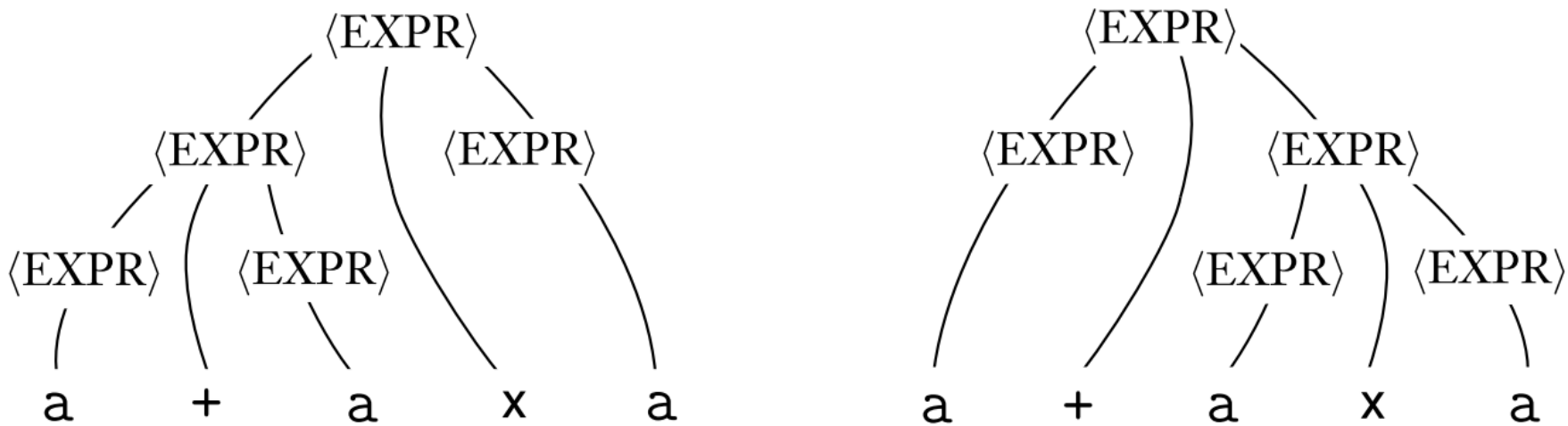
Ambiguity

Sometimes a grammar can generate the same string in several different ways. Such a string will have several different parse trees and thus several different meanings. This result may be undesirable for certain applications, such as programming languages, where a program should have a unique interpretation.

Example: ***consider grammar:***

$$\langle \text{EXPR} \rangle \rightarrow \langle \text{EXPR} \rangle + \langle \text{EXPR} \rangle | \langle \text{EXPR} \rangle \times \langle \text{EXPR} \rangle | (\langle \text{EXPR} \rangle) | a$$

This grammar generates the string $a + a \times a$ ***ambiguously. The following figure shows the two different parse trees.***



In contrast, the following grammar generates exactly the same language, but every generated string has a unique parse tree. Hence it is unambiguous, whereas the previous one was ambiguous.

$$\begin{aligned}
\langle \text{EXPR} \rangle &\rightarrow \langle \text{EXPR} \rangle + \langle \text{TERM} \rangle \mid \langle \text{TERM} \rangle \\
\langle \text{TERM} \rangle &\rightarrow \langle \text{TERM} \rangle \times \langle \text{FACTOR} \rangle \mid \langle \text{FACTOR} \rangle \\
\langle \text{FACTOR} \rangle &\rightarrow (\langle \text{EXPR} \rangle) \mid \texttt{a}
\end{aligned}$$

A context-free grammar G is said to be ambiguous if there exists some $w \in L(G)$ that has at least two distinct derivation trees.

* *When we say that a grammar generates a string ambiguously, we mean that the string has two different parse trees, not two different derivations. Two derivations may differ merely in the order in which they replace variables yet not in their overall structure.*

* *Derivation trees show which productions are used in obtaining a sentence, but do not give the order of their application. A parse tree represents the rules applied, but not the order of their application.*

* *A grammar is unambiguous if and only if every word generated has exactly one parse tree.*

تذکر: درایویشن‌هایی که درخت تجزیهٔ آنها یکسان است، ماهیتاً تمایز عمده‌ای با هم ندارند و صرفاً ترتیب بکارگیری رول‌ها در آنها متفاوت است. در مقابل، درایویشن‌هایی که درخت‌های تجزیهٔ آنها متفاوت است، ماهیتاً متفاوت هستند. مثلاً، گرامر

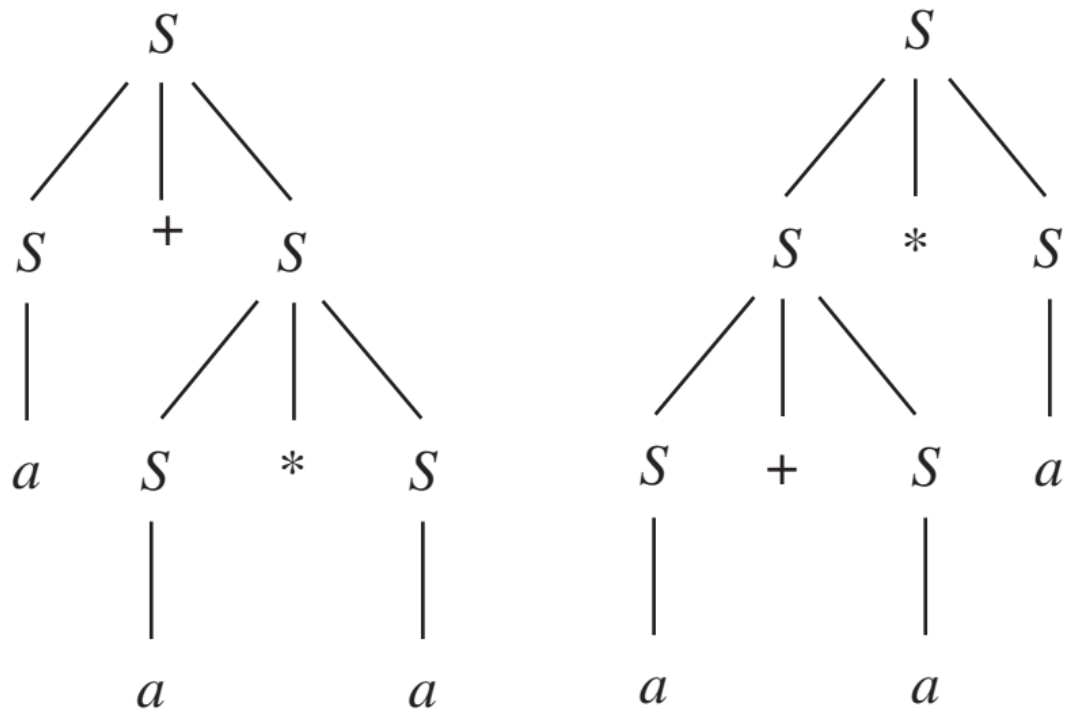
$$S \rightarrow a|S + S|S * S|(S)$$

را در نظر بگیرید.

در این گرامر، دو درایویشن زیر برای رشتهٔ $a + a \times a$، ماهیتاً متفاوت هستند چون درخت‌های تجزیهٔ نظیر آن‌دو متفاوت است:

$$S \Rightarrow S + S \Rightarrow a + S \Rightarrow a + S * S \Rightarrow a + a * S \Rightarrow a + a * a$$

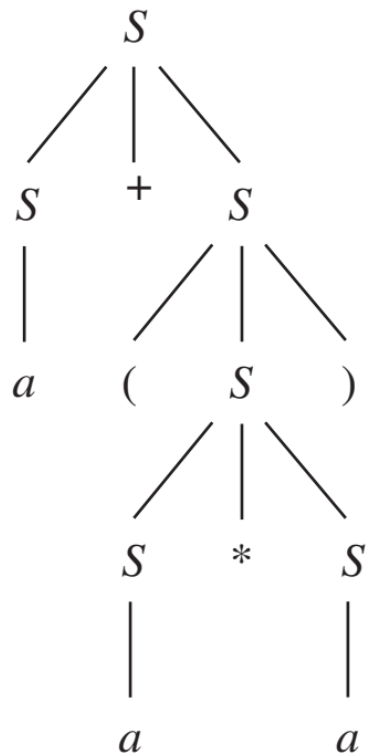
$$S \Rightarrow S * S \Rightarrow S + S * S \Rightarrow a + S * S \Rightarrow a + a * S \Rightarrow a + a * a$$



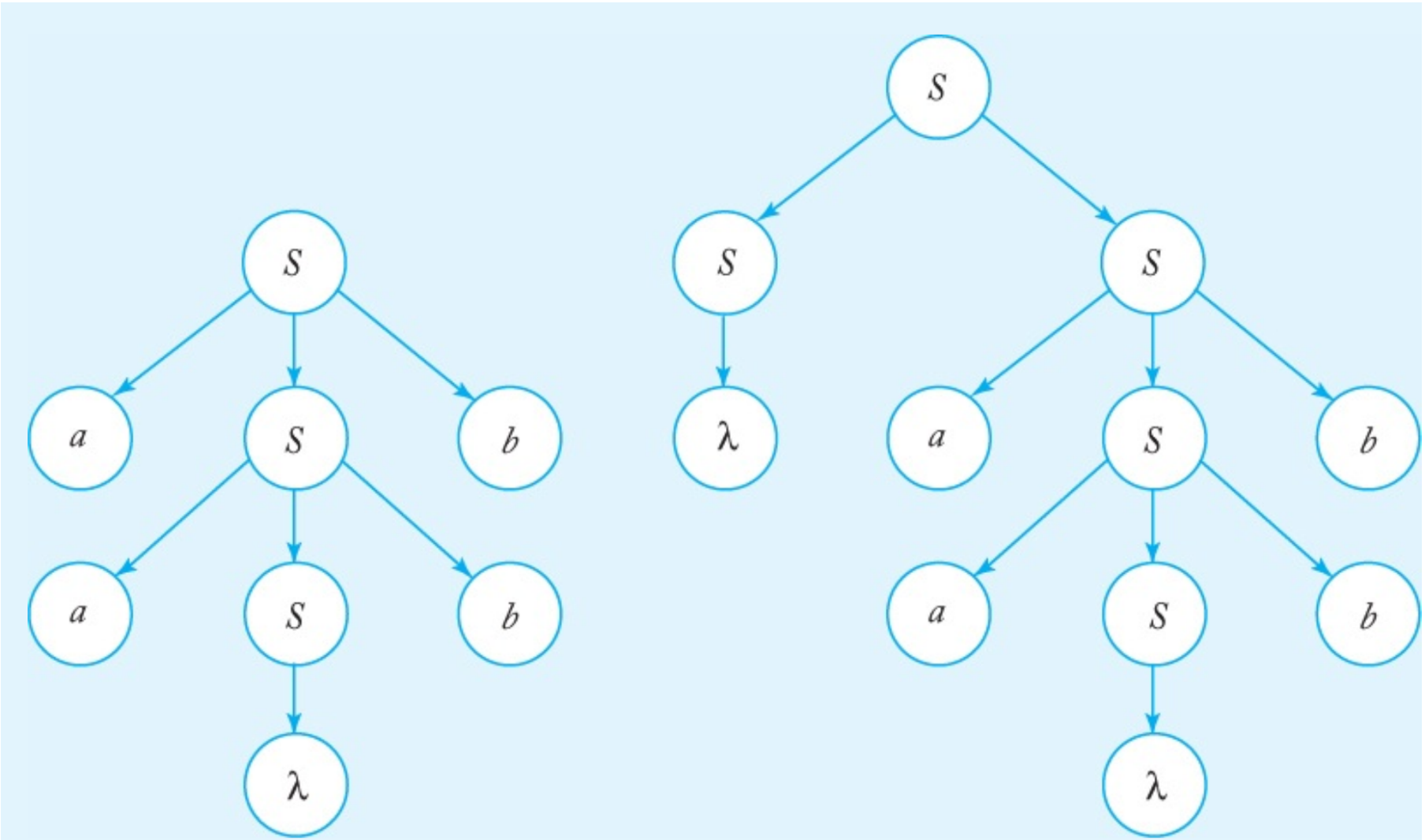
اما دو درایویشن زیر برای رشتهٔ $a + (a \times a)$ ماهیتاً متمایز نیستند چون درخت تجزیهٔ نظیر آنها یکسان است. تمایز این‌دو صرفاً در ترتیب بکارگیری رول‌ها است:

$$S \Rightarrow S+S \Rightarrow a+S \Rightarrow a+(S) \Rightarrow a+(S*S) \Rightarrow a+(a*S) \Rightarrow a+(a*a)$$

$$S \Rightarrow S+S \Rightarrow S+(S) \Rightarrow S+(S*S) \Rightarrow S+(a*S) \Rightarrow S+(a*a) \Rightarrow a+(a*a)$$



Example: ***The grammar*** $S \rightarrow aSb|SS|\varepsilon$ ***is ambiguous. The sentence*** $aabb$ ***has the two derivation trees:***



مفهوم ابهام با بهره‌گیری از دو مفهوم *LMD* و *RMD* نیز قابل بیان است:

If G is a context-free grammar, then for every $x \in L(G)$, these three statements are equivalent:
1. x has more than one derivation tree.
2. x has more than one leftmost derivation.
3. x has more than one rightmost derivation.

Alternatively, ambiguity implies the existence of two or more LMDs (RMDs). In fact, a string w is derived ambiguously in CFG G if it has two or more different LMDs (RMDs). Grammar G is ambiguous if it generates some string ambiguously. A grammar G is said to be unambiguous if every word $w \in L(G)$ has exactly one LMD (RMD) and ambiguous otherwise.

مفهوم ذاتاً مبهم بودن که به یک زبان اطلاق می‌شود:

If L is a context-free language for which there exists an unambiguous grammar, then L is said to be unambiguous. If every grammar that generates L is ambiguous, then the language is called inherently ambiguous. Indeed, sometimes when we have an ambiguous grammar we can find an unambiguous grammar that generates the same language. Some context-free languages, however, can be generated only by ambiguous grammars. Such languages are called inherently ambiguous.

It can be shown that, the language $\{a^i b^j c^k | i = j \text{ or } j = k\}$ is inherently ambiguous.

Example: *The language* $L = \{a^n b^n c^m\} \cup \{a^n b^m c^m\}$*, with* n *and* m *nonnegative, is an inherently ambiguous context-free language. That* L *is context-free is easy to show. Notice that* $L = L_1 \cup L_2$*, where* L_1 *is generated by*

$$S_1 \to S_1 c | A, \quad A \to aAb | \varepsilon$$

and L_2 *is given by an analogous grammar with start symbol* S_2 *and productions*

$$S_2 \to aS_2 | B, \quad B \to bBc | \varepsilon$$

Then L *is generated by the combination of these two grammars with the additional production* $S \to S_1 | S_2$*. The grammar is ambiguous since the string* $a^n b^n c^n$ *has two distinct derivations, one starting with* $S \Rightarrow S_1$*, the other with* $S \Rightarrow S_2$*. It does not, of course, follow from this that* L *is inherently ambiguous as there might exist some other unambiguous grammars for it. A rigorous argument is quite technical.*

Chomsky Normal Form

When working with context-free grammars, it is often convenient to have them in simplified form. One of the simplest and most useful forms is called the Chomsky normal form. In general, a context-free grammar can be transformed to a grammar in Chomsky normal form without changing the language to be generated.

تعریف فرم نرمال چامسکی

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

$$A \rightarrow BC$$
$$A \rightarrow a$$

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 $S \rightarrow \varepsilon$, where S is the start variable.

The ε-rule $S \rightarrow \varepsilon$ is permitted only for the start symbol S.

Theorem: *Any context-free language is generated by a context-free grammar in Chomsky normal form.*

Proof Idea: *We can convert any grammar G into Chomsky normal form. The conversion has several stages wherein rules that violate the conditions are replaced with equivalent ones that are satisfactory.*

- ☞ *First, we add a new start variable.*
- ☞ *Then, we eliminate all ε-rules of the form $A \to \varepsilon$.*
- ☞ *We also eliminate all unit (renaming) rules of the form $A \to B$.*
- ☞ *In both cases we patch up the grammar to be sure that it still generates the same language.*
- ☞ *Finally, we convert the remaining rules into the proper form. A rule $A \to u_1 u_2 \cdots u_k$ in which the length of the right-hand side k is more than or equal to 3 is called a long rule.*

پس روند در مجموع به‌صورت ذیل است:

We transform G to an equivalent context-free grammar in Chomsky normal form by deleting the ε-rules, the renaming rules, and finally the long rules from the rules of G. We shall explain the idea for deleting the three types of rules.

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. Observe that this change does not change the language to be generated and that the start symbol S_0 never appears on the right-hand side of any 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 (deletion of renaming 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. In fact, for each renaming rule $A \to B$, the deletion of $A \to B$ is applied for all the rules that take the form of $B \to u$, where $u \in (V \cup \Sigma)^*$. By the deletion, we eliminate the rule $A \to B$ and add all the rules that are expressed as $A \to u$ provided that $A \to u$ was not deleted so far. For each renaming rule this type of deletion and addition will be applied concerning all the remaining rules, including ones that are added while applying other deletions.*

☞ *Finally, we convert all remaining rules into the proper form. We replace each rule $A \to u_1 u_2 \cdots u_k$, where $k \geq 3$ and each u_i is a variable or terminal symbol, with the rules $A \to u_1 A_1$, $A_1 \to u_2 A_2$,*

$A_2 \rightarrow u_3 A_3$, ..., ***and*** $A_{k-2} \rightarrow u_{k-1}u_k$. ***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*** $U_i \rightarrow u_i$.

نکات مهمی دربارهٔ حذف ε-***rule***ها که باید مجدداً متذکر شوم:

☞ *Since in Chomsky normal form $S \to \varepsilon$ is permitted for the start symbol S, the ε-rules we must delete are the ones with variables on the left-hand side other than the start symbol.*

☞ *Once the deletion of $A \to \varepsilon$ is applied for all the rules in the grammar, the ε-rule $A \to \varepsilon$ will never be taken up again in the later deletion process.*

☞ *After the deletion of all the ε rules for all the rules is done, the only ε rule that may possibly be left is $S_0 \to \varepsilon$.*

Example: ***The series of grammars presented illustrates the steps in the conversion. Rules shown in bold have just been added. Rules shown in gray have just been removed.***

1. The original CFG G_6 is shown on the left. The result of applying the first step to make a new start variable appears on the right.

$$S \to ASA \mid \mathtt{a}B$$
$$A \to B \mid S$$
$$B \to \mathtt{b} \mid \varepsilon$$

$$\boldsymbol{S_0 \to S}$$
$$S \to ASA \mid \mathtt{a}B$$
$$A \to B \mid S$$
$$B \to \mathtt{b} \mid \varepsilon$$

2. Remove ε-rules $B \to \varepsilon$, shown on the left, and $A \to \varepsilon$, shown on the right.

$$S_0 \to S$$
$$S \to ASA \mid \mathtt{a}B \mid \mathbf{a}$$
$$A \to B \mid S \mid \boldsymbol{\varepsilon}$$
$$B \to \mathtt{b} \mid \varepsilon$$

$$S_0 \to S$$
$$S \to ASA \mid \mathtt{a}B \mid \mathtt{a} \mid \boldsymbol{SA} \mid \boldsymbol{AS} \mid \boldsymbol{S}$$
$$A \to B \mid S \mid \varepsilon$$
$$B \to \mathtt{b}$$

ادامهٔ مثال:

3a. Remove unit rules $S \to S$, shown on the left, and $S_0 \to S$, shown on the right.

$$\begin{aligned} S_0 &\to S \\ S &\to ASA \mid \mathtt{a}B \mid \mathtt{a} \mid SA \mid AS \mid S \\ A &\to B \mid S \\ B &\to \mathtt{b} \end{aligned} \qquad \begin{aligned} S_0 &\to S \mid \boldsymbol{ASA} \mid \mathbf{a}\boldsymbol{B} \mid \mathbf{a} \mid \boldsymbol{SA} \mid \boldsymbol{AS} \\ S &\to ASA \mid \mathtt{a}B \mid \mathtt{a} \mid SA \mid AS \\ A &\to B \mid S \\ B &\to \mathtt{b} \end{aligned}$$

3b. Remove unit rules $A \to B$ and $A \to S$.

$$\begin{aligned} S_0 &\to ASA \mid \mathtt{a}B \mid \mathtt{a} \mid SA \mid AS \\ S &\to ASA \mid \mathtt{a}B \mid \mathtt{a} \mid SA \mid AS \\ A &\to B \mid S \mid \mathbf{b} \\ B &\to \mathtt{b} \end{aligned} \qquad \begin{aligned} S_0 &\to ASA \mid \mathtt{a}B \mid \mathtt{a} \mid SA \mid AS \\ S &\to ASA \mid \mathtt{a}B \mid \mathtt{a} \mid SA \mid AS \\ A &\to S \mid \mathtt{b} \mid \boldsymbol{ASA} \mid \mathbf{a}\boldsymbol{B} \mid \mathbf{a} \mid \boldsymbol{SA} \mid \boldsymbol{AS} \\ B &\to \mathtt{b} \end{aligned}$$

ادامهٔ مثال:

4. Convert the remaining rules into the proper form by adding additional variables and rules. The final grammar in Chomsky normal form is equivalent to G_6. (Actually the procedure given in Theorem 2.9 produces several variables U_i and several rules $U_i \to \mathtt{a}$. We simplified the resulting grammar by using a single variable U and rule $U \to \mathtt{a}$.)

$$
\begin{aligned}
S_0 &\to AA_1 \mid UB \mid \mathtt{a} \mid SA \mid AS \\
S &\to AA_1 \mid UB \mid \mathtt{a} \mid SA \mid AS \\
A &\to \mathtt{b} \mid AA_1 \mid UB \mid \mathtt{a} \mid SA \mid AS \\
A_1 &\to SA \\
U &\to \mathtt{a} \\
B &\to \mathtt{b}
\end{aligned}
$$

Example: ***We transform the grammar*** $S \to aS|aSbS|\varepsilon$ ***to an equivalent context-free grammar which is in Chomsky normal form in the way described above.***

First, ***introducing the start symbol*** S_0***, we modify the grammar as follows:***

$$S_0 \to S, S \to aS|aSbS|\varepsilon.$$

Next, ***we delete*** $S \to \varepsilon$ ***by applying what is described above to the three rules: By applying to*** $S_0 \to S$ ***we have*** $S_0 \to S|\varepsilon$***. By applying to*** $S \to aS$ ***we have*** $S \to aS|a$***. By applying to*** $S \to aSbS$ ***we have*** $S \to aSbS|abS|aSb|ab$***. Putting them together, we have***

$$S_0 \to S|\varepsilon, S \to aS|a|aSbS|abS|aSb|ab.$$

Next, ***to delete*** $S_0 \to S$***, we add*** $S_0 \to w$ ***for each rule of the form*** $S \to w$***, thereby obtaining rules as follows:***

$$S_0 \to aS|a|aSbS|abS|aSb|ab|\varepsilon, S \to aS|a|aSbS|abS|aSb|ab.$$

Finally, we delete long rules. Deleting $S \to aSbS$***, we add instead***

$$S \to X_aA_2, A_2 \to SA_3, A_3 \to X_bS, X_a \to a, \text{ and } X_b \to b.$$

In this way we have the equivalent rules in Chomsky normal form as follows:

$$\begin{aligned}
S_0 &\to X_aS|a|X_aA_2|X_aB_2|X_aC_2|X_aX_b|\varepsilon,\\
S &\to X_aS|a|X_aA_2|X_aB_2|X_aC_2|X_aX_b,\\
A_2 &\to SA_3,\\
A_3 &\to X_bS,\\
B_2 &\to X_bS,\\
C_2 &\to SX_b,\\
X_a &\to a,\\
X_b &\to b.
\end{aligned}$$

تمرین:

Transform the grammar

$$S \to aSaaA|A, \quad A \to abA|bb$$

into Chomsky normal form.

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, the Greibach normal form has several uses, both practical and theoretical.

Any CFG can be written in a special form called Greibach normal form. (For every CFG G, there exists an equivalent grammar $\widehat{G}$ in Greibach normal form.) Each production has one of the following forms: $A \to aB_1B_2\cdots B_n$, $A \to a$, $S \to \varepsilon$, where S is the start symbol, a is a terminal, and each B_k is a variable not equal to S. ε can occur only with the production $S \to \varepsilon$.