Formal languages and Automata
形式语言与自动机

Chapter6 SIMPLIFICATION OF CONTEXT-FREE GRAMMARS AND NORMAL FORMS

Beijing University of Posts and Telecommunications

北京邮电大学

刘咏彬 liuyb@bupt.edu.cn

2023.11

Overview - Membership and Parsing in Context-Free Languages

1. The definition of a context-free grammar imposes no restriction whatsoever on the right side of a production.
    - Not necessary
    - Detrimental in some arguments.
2. Grammar Simplification （化简） and Normalization （范式）：
    - Eliminates λ-productions (empty productions) 空产生式
    - Eliminates unit-productions (productions with a single variable).单一产生式
    - Removes useless productions that never appear in string derivations.无用产生式、无用符号
3. Normal Forms for Context-Free Grammars:
    - Introduces two most useful normal forms: Chomsky Normal Form（乔姆斯基范式） and Greibach Normal Form（格里巴克范式）.
    - These forms have numerous practical and theoretical applications.

METHODS FOR TRANSFORMING GRAMMARS

- Handling the Empty String in Grammars: λ-free languages.
- Example
  - Let L be a context-free language and G= (V, T, S, P) a corresponding grammar for L - {λ}. Adding a new variable S_0, making S_0 the start variable,and adding productions $S_0 \rightarrow S \mid \lambda$ to P, generates L.
  - Given any context-free grammar G, there is a method for obtaining $\widehat{G}$ such that $L(\widehat{G}) = L(G) - \{\lambda\}$.
- A Useful Substitution Rule for Simplification:
  - Various rules exist for generating equivalent grammars through substitutions.
  - Simplification here refers to removing certain types of undesirable productions.
  - The process doesn' t always reduce the number of rules.

Substitution Rule(代入)

- **THEOREM 6.1**

Let $G = (V, T, S, P)$ be a context-free grammar. Suppose that P contains a production of the form

$$A \to x_1 B x_2,$$

Assume that A and B are different variables and that

$$B \to y_1 | y_2 | \cdots | y_n$$

is the set of all productions in P that have B as the left side. Let $\widehat{G} = (V, T, S, \widehat{P})$ be the grammar in which $\widehat{P}$ is constructed by deleting

$$A \to x_1 B x_2 \tag{6.1}$$

from P, and adding to it

$$A \to x_1 y_1 x_2 | x_1 y_2 x_2 | \cdots | x_1 y_n x_2.$$

Then

$$L(\widehat{G}) = L(G)$$

EXAMPLE 6.1

Consider $G = (\{A, B\}, \{a, b, c\}, A, P)$ with productions

$$\begin{aligned} A &\to a\,|aaA|\,abBc, \\ B &\to abbA|b. \end{aligned}$$

Using the suggested substitution for the variable B, we get the grammar $\widehat{G}$ with productions

$$\begin{aligned} A &\to a\,|aaA|\,ababbAc|abbc, \\ B &\to abbA|b. \end{aligned}$$

The new grammar $\widehat{G}$ is equivalent to G. The string $aaabbc$ has the derivation

$$A \Rightarrow aaA \Rightarrow aaabBc \Rightarrow aaabbc$$

in G, and the corresponding derivation

$$A \Rightarrow aaA \Rightarrow aaabbc$$

in $\widehat{G}$.

Notice that, in this case, the variable B and its associated productions are still in the grammar even though they can no longer play a part in any derivation. We will next show how such unnecessary productions can be removed from a grammar.

Removing Useless Productions （去除无用产生式）

- An example, in this Grammar “A” is useless
  - $S \to aSb \mid \lambda \mid A$,
  - $A \to aA$,

- **DEFINITION 6.1**

  Let $G = (V, T, S, P)$ be a context-free grammar. A variable $A \in V$ is said to be useful if and only if there is at least one $w \in L(G)$ such that

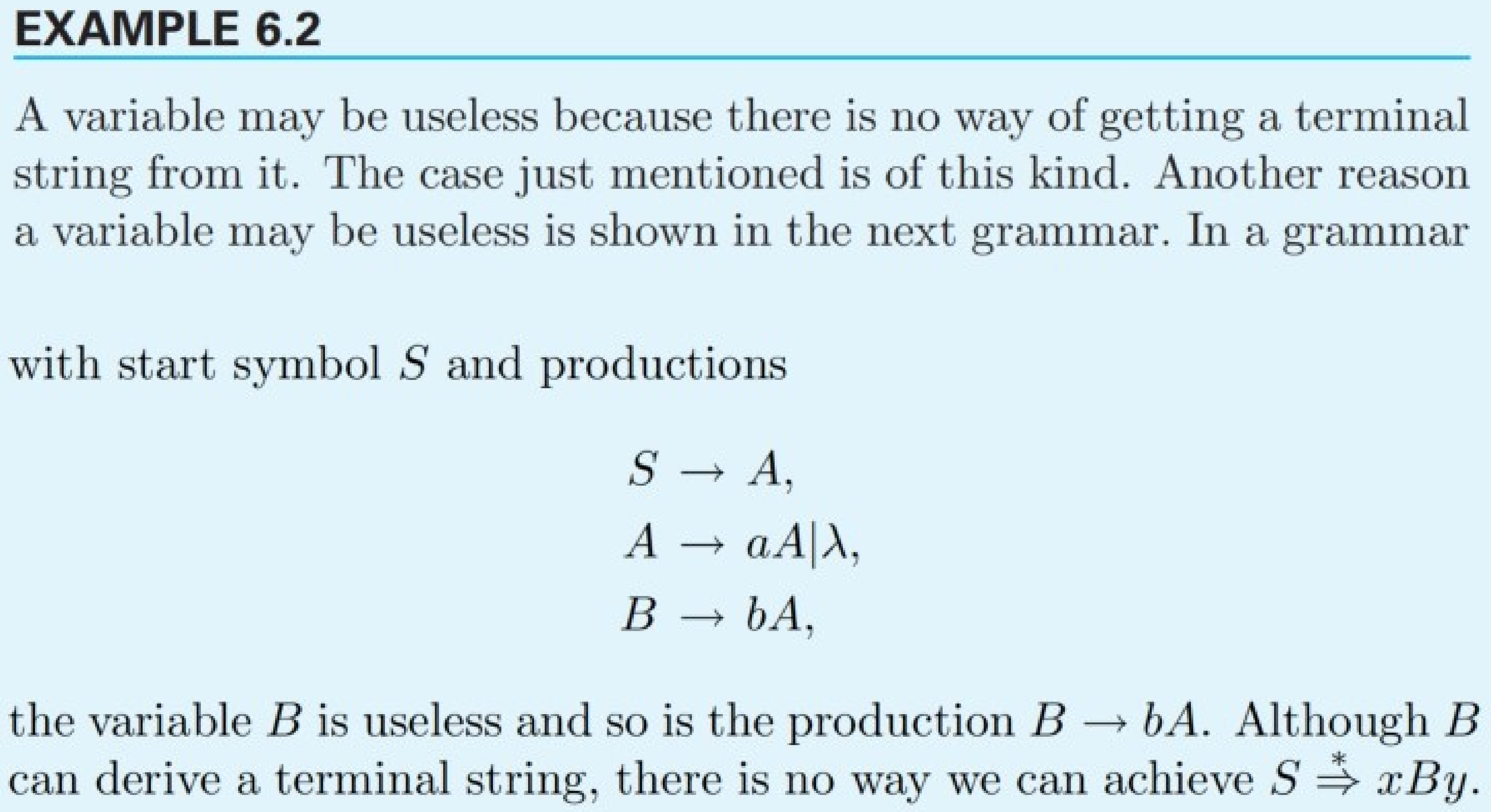
$$S \overset{*}{\Rightarrow} xAy \overset{*}{\Rightarrow} w$$

两类无用符号：
1. 从开始符号推导不出的变量或终极符号
2. 此变量推导不出终极符号串

  with x, y in $(V \cup T)^*$.

  - In words, a variable is useful if and only if it occurs in at least one derivation.
  - A variable that is not useful is called useless.
  - A production is useless if it involves any useless variable.

Removing Useless Productions



EXAMPLE 6.2

A variable may be useless because there is no way of getting a terminal string from it. The case just mentioned is of this kind. Another reason a variable may be useless is shown in the next grammar. In a grammar

with start symbol S and productions

$$\begin{aligned} S &\rightarrow A, \\ A &\rightarrow aA|\lambda, \\ B &\rightarrow bA, \end{aligned}$$

the variable B is useless and so is the production $B \rightarrow bA$. Although B can derive a terminal string, there is no way we can achieve $S \overset{*}{\Rightarrow} xBy$.

Removing Useless Productions

EXAMPLE 6.3

Eliminate useless symbols and productions from $G = (V, T, S, P)$, where $V = \{S, A, B, C\}$ and $T = \{a, b\}$, with P consisting of

$$\begin{aligned} S &\rightarrow aS\,|A|\,C, \\ A &\rightarrow a, \\ B &\rightarrow aa, \\ C &\rightarrow aCb. \end{aligned}$$

First, we identify the set of variables that can lead to a terminal string. Because $A \rightarrow a$ and $B \rightarrow aa$, the variables A and B belong to this set. So does S, because $S \Rightarrow A \Rightarrow a$. However, this argument cannot be made for C, thus identifying it as useless. Removing C and its corresponding productions, we are led to the grammar G_1 with variables $V_1 = \{S, A, B\}$, terminals $T = \{a\}$, and productions

$$\begin{aligned} S &\rightarrow aS|A, \\ A &\rightarrow a, \\ B &\rightarrow aa. \end{aligned}$$

Removing Useless Productions

Next we want to eliminate the variables that cannot be reached from the start variable. For this, we can draw a **dependency graph** for the variables. Dependency graphs are a way of visualizing complex relationships and are found in many applications. For context-free grammars, a dependency graph has its vertices labeled with variables,

with an edge between vertices C and D if and only if there is a production of the form

$$C \rightarrow xDy.$$

A dependency graph for V_1 is shown in Figure 6.1. A variable is useful only if there is a path from the vertex labeled S to the vertex labeled with that variable. In our case, Figure 6.1 shows that B is useless. Removing it and the affected productions and terminals, we are led to the final answer $\widehat{G} = \left(\widehat{V}, \widehat{T}, S, \widehat{P}\right)$ with $\widehat{V} = \{S, A\}, \widehat{T} = \{a\}$, and productions

$$S \rightarrow aS|A,$$
$$A \rightarrow a.$$

The formalization of this process leads to a general construction and the corresponding theorem.

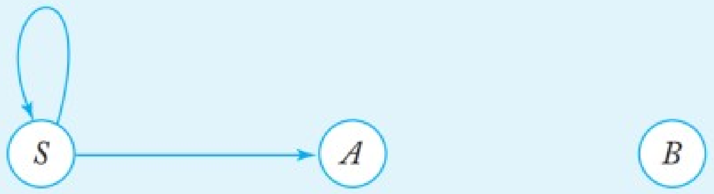
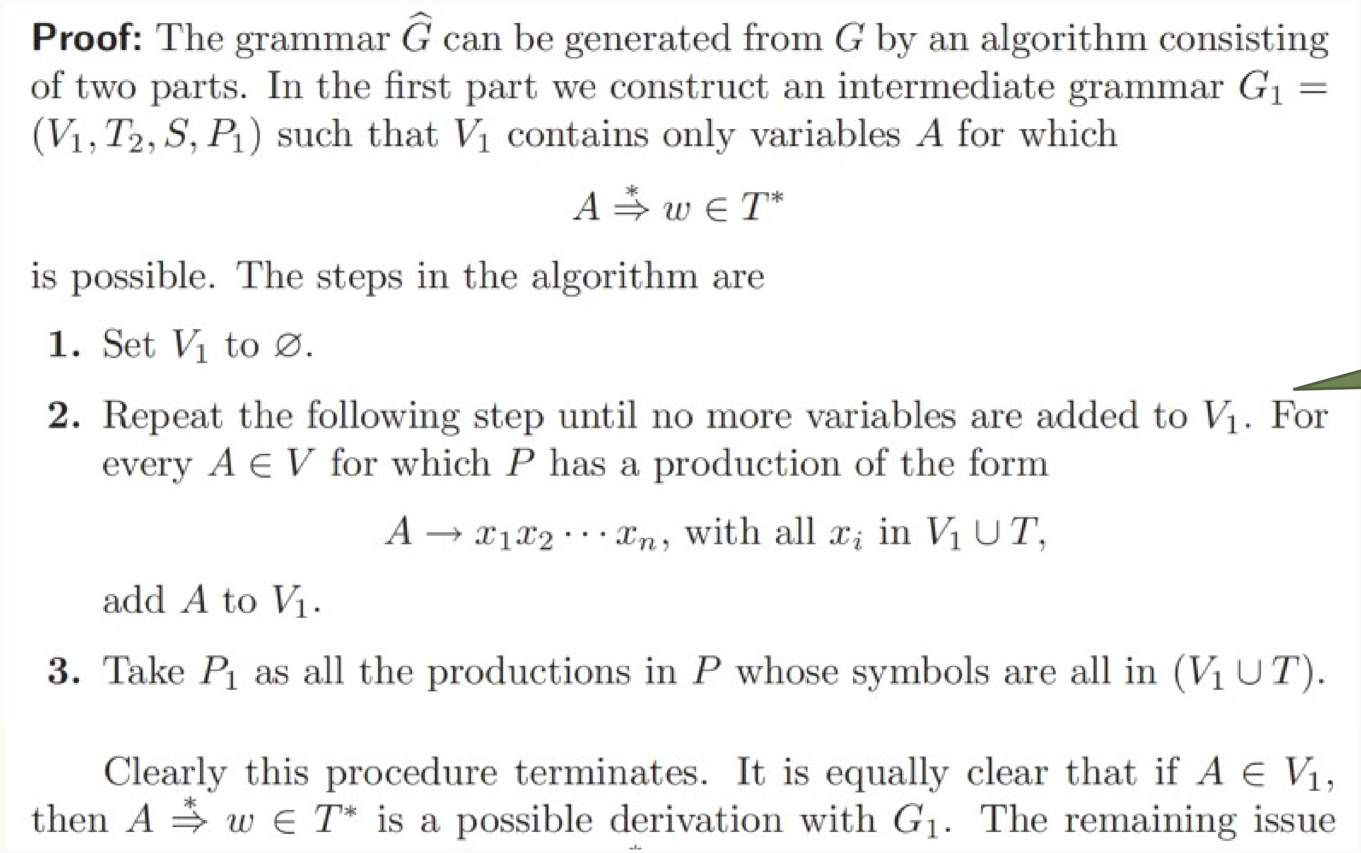


FIGURE 6.1

How to Remove Useless Productions (1)

- **THEOREM 6.2**

  Let $G = (V, T, S, P)$ be a context-free grammar. Then there exists an equivalent grammar $\widehat{G} = (\widehat{V}, \widehat{T}, S, \widehat{P})$ that does not contain any usless variables or productions.



Proof: The grammar $\widehat{G}$ can be generated from G by an algorithm consisting of two parts. In the first part we construct an intermediate grammar $G_1 = (V_1, T_2, S, P_1)$ such that V_1 contains only variables A for which

$$A \overset{*}{\Rightarrow} w \in T^*$$

is possible. The steps in the algorithm are

1. Set V_1 to $\varnothing$.

2. Repeat the following step until no more variables are added to V_1. For every $A \in V$ for which P has a production of the form

$$A \rightarrow x_1 x_2 \cdots x_n, \text{ with all } x_i \text{ in } V_1 \cup T,$$

add A to V_1.

3. Take P_1 as all the productions in P whose symbols are all in $(V_1 \cup T)$.

Clearly this procedure terminates. It is equally clear that if $A \in V_1$, then $A \overset{*}{\Rightarrow} w \in T^*$ is a possible derivation with G_1. The remaining issue

方法:
如何消除推导不出
终极符号串的变量

How to Remove Useless Productions (2)

直至A变量被加入V_1，算法停止

以此类推。。。。

倒数第三层的变量会在第一步加入V_1

倒数第二层的变量会在第一步加入V_1

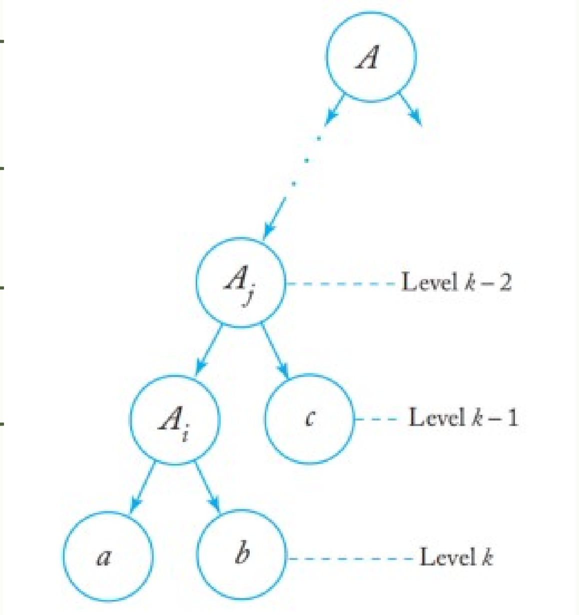
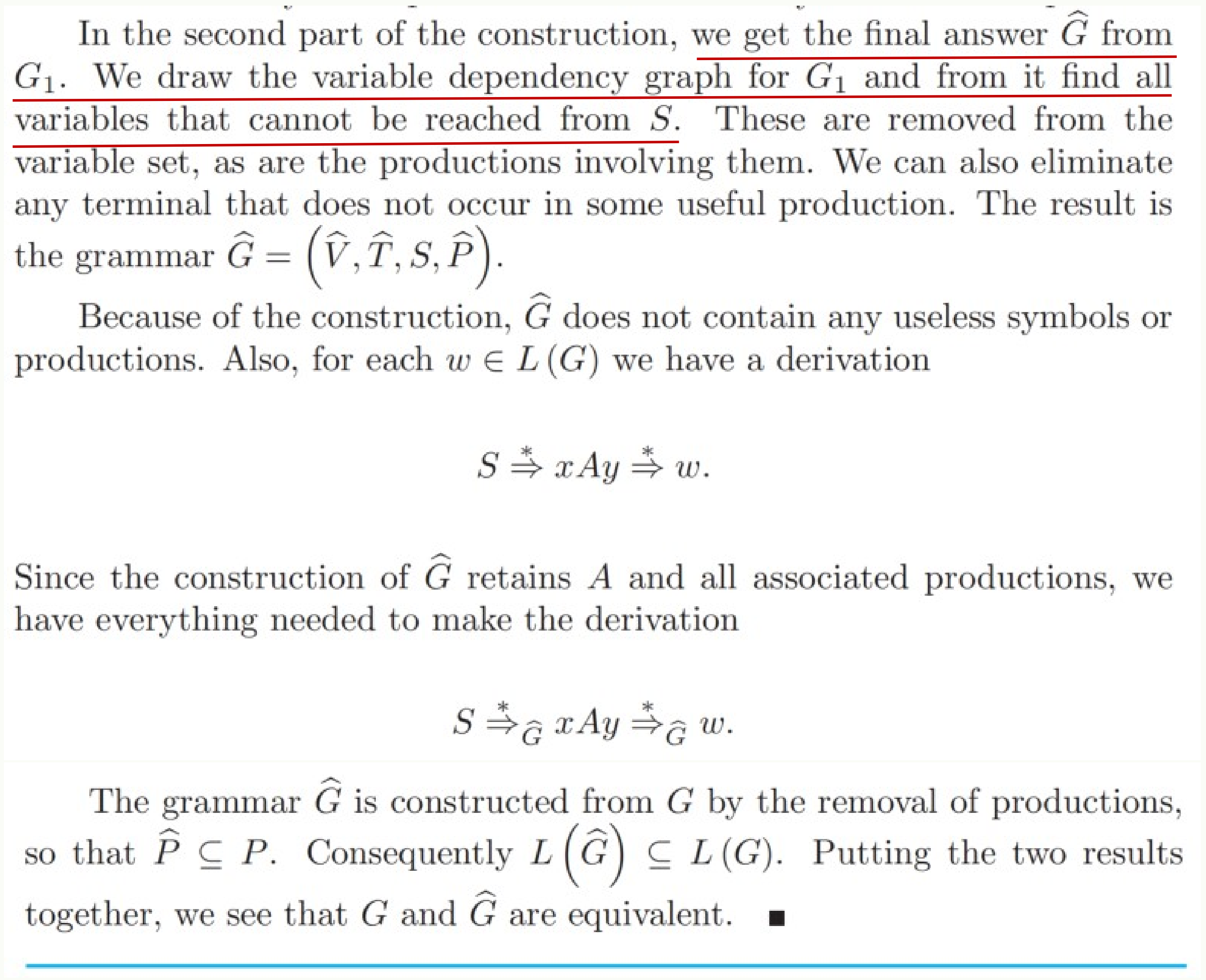


FIGURE 6.2

whether every A for which $A \overset{*}{\Rightarrow} w = ab\cdots$ is added to V_1 before the procedure terminates. To see this, consider any such A and look at the partial derivation tree corresponding to that derivation (Figure 6.2). At level k, there are only terminals, so every variable A_i at level $k-1$ will be added to V_1 on the first pass through Step 2 of the algorithm. Any variable at level $k-2$ will then be added to V_1 on the second pass through Step 2. The third time through Step 2, all variables at level $k-3$ will be added, and so on. The algorithm cannot terminate while there are variables in the tree that are not yet in V_1. Hence A will eventually be added to V_1.

How to Remove Useless Productions (3)



In the second part of the construction, we get the final answer $\widehat{G}$ from G_1. We draw the variable dependency graph for G_1 and from it find all variables that cannot be reached from S. These are removed from the variable set, as are the productions involving them. We can also eliminate any terminal that does not occur in some useful production. The result is the grammar $\widehat{G} = \left(\widehat{V}, \widehat{T}, S, \widehat{P}\right)$.

Because of the construction, $\widehat{G}$ does not contain any useless symbols or productions. Also, for each $w \in L(G)$ we have a derivation

$$S \stackrel{*}{\Rightarrow} xAy \stackrel{*}{\Rightarrow} w.$$

Since the construction of $\widehat{G}$ retains A and all associated productions, we have everything needed to make the derivation

$$S \stackrel{*}{\Rightarrow}_{\widehat{G}} xAy \stackrel{*}{\Rightarrow}_{\widehat{G}} w.$$

The grammar $\widehat{G}$ is constructed from G by the removal of productions, so that $\widehat{P} \subseteq P$. Consequently $L\left(\widehat{G}\right) \subseteq L(G)$. Putting the two results together, we see that G and $\widehat{G}$ are equivalent. ■

Removing λ-Productions

- DEFINITION 6.2

  Any production of a context-free grammar of the form

  $$A \rightarrow \lambda$$

  is called a λ-production（空产生式）. Any variable A for which the derivation

  $$A \overset{*}{\Rightarrow} \lambda \qquad (6.3)$$

  is possible is called nullable（可空变量）.

A grammar may generate a language not containing λ, yet have some λ-productions or nullable variables. In such cases, the λ-productions can be removed.

EXAMPLE 6.4

Consider the grammar

$$S \rightarrow aS_1b,$$
$$S_1 \rightarrow aS_1b|\lambda,$$

with start variable S. This grammar generates the λ-free language $\{a^nb^n : n \geq 1\}$. The λ-production $S_1 \rightarrow \lambda$ can be removed after adding new productions obtained by substituting λ for S_1 where it occurs on the right. Doing this we get the grammar

$$S \rightarrow aS_1b|ab,$$
$$S_1 \rightarrow aS_1b|ab.$$

We can easily show that this new grammar generates the same language as the original one.

In more general situations, substitutions for λ-productions can be made in a similar, although more complicated, manner.

THEOREM 6.3

Let G be any context-free grammar with λ not in $L(G)$. Then there exists an equivalent grammar $\widehat{G}$ having no λ-productions.

How to Remove λ-production

★

Proof: We first find the set V_N of all nullable variables of G, using the following steps.

1. For all productions $A \rightarrow \lambda$, put A into $\underline{V_N}$.

2. Repeat the following step until no further variables are added to V_N.

   For all productions

$$B \rightarrow A_1 A_2 \cdots A_n,$$

   where $A_1, A_2, ..., A_n$ are in V_N, put B into V_N.

Once the set V_N has been found, we are ready to construct $\widehat{P}$. To do so, we look at all productions in P of the form

$$A \rightarrow x_1 x_2 \cdots x_m, m \geq 1,$$

where each $x_i \in V \cup T$. For each such production of P, we put into $\widehat{P}$ that production as well as all those generated by replacing nullable variables with λ in all possible combinations. For example, if x_i and x_j are both nullable, there will be one production in $\widehat{P}$ with x_i replaced with λ, one in which x_j is replaced with λ, and one in which both x_i and x_j are replaced with λ. There is one exception: If all x_i are nullable, the production $A \rightarrow \lambda$ is not put into $\widehat{P}$.

The argument that this grammar $\widehat{G}$ is equivalent to G is straightforward and will be left to the reader. ■

EXAMPLE 6.5

Find a context-free grammar without λ-productions equivalent to the grammar defined by

$$\begin{aligned} S &\rightarrow ABaC, \\ A &\rightarrow BC, \\ B &\rightarrow b|\lambda, \\ C &\rightarrow D|\lambda, \\ D &\rightarrow d. \end{aligned}$$

From the first step of the construction in Theorem 6.3, we find that the nullable variables are A, B, C. Then, following the second step of the construction, we get

$$\begin{aligned} S &\rightarrow ABaC\,|BaC|\,AaC\,|ABa|\,aC\,|Aa|\,Ba|a, \\ A &\rightarrow B\,|C|\,BC, \\ B &\rightarrow b, \\ C &\rightarrow D, \\ D &\rightarrow d. \end{aligned}$$

Removing Unit-Productions

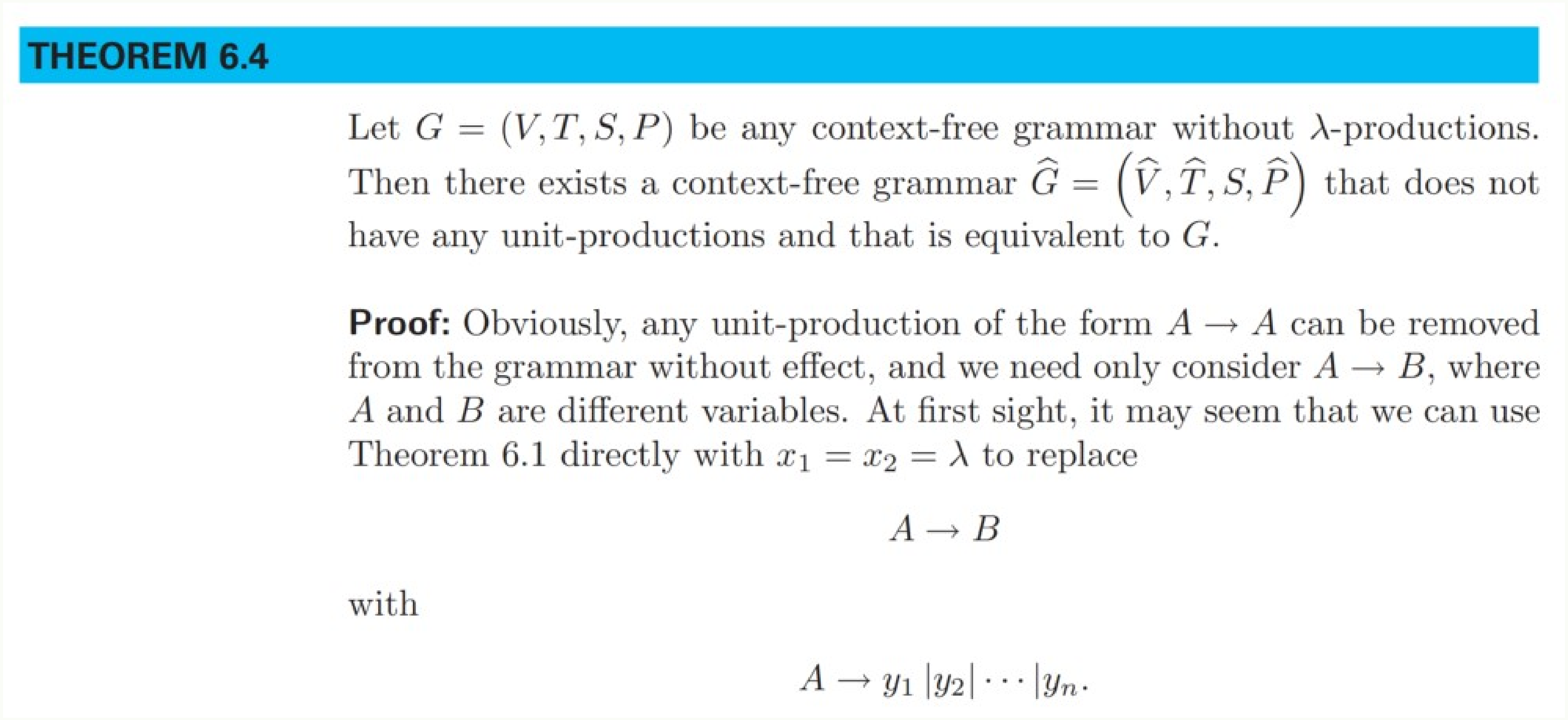
- DEFINITION 6.3

  Any production of a context-free grammar of the form

  $$A \to B,$$

  where $A, B \in V$, is called a unit-production（单一产生式）.

How to Remove Unit-Productions (1)



THEOREM 6.4

Let $G = (V, T, S, P)$ be any context-free grammar without λ-productions. Then there exists a context-free grammar $\widehat{G} = \left(\widehat{V}, \widehat{T}, S, \widehat{P}\right)$ that does not have any unit-productions and that is equivalent to G.

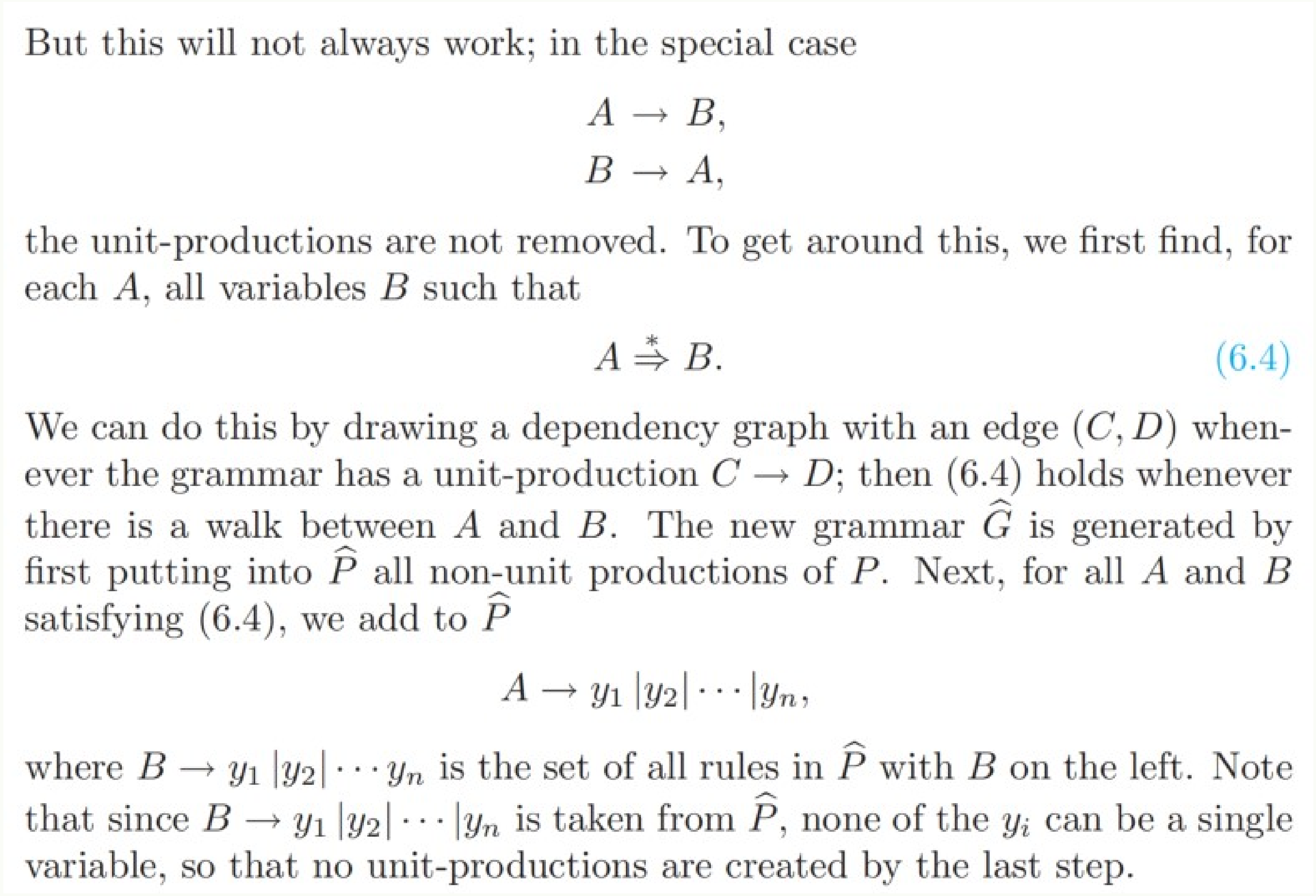
Proof: Obviously, any unit-production of the form $A \rightarrow A$ can be removed from the grammar without effect, and we need only consider $A \rightarrow B$, where A and B are different variables. At first sight, it may seem that we can use Theorem 6.1 directly with $x_1 = x_2 = \lambda$ to replace

$$A \rightarrow B$$

with

$$A \rightarrow y_1 \left| y_2 \right| \cdots \left| y_n \right..$$

How to Remove Unit-Productions (2)



But this will not always work; in the special case

$$A \to B,$$
$$B \to A,$$

the unit-productions are not removed. To get around this, we first find, for each A, all variables B such that

$$A \overset{*}{\Rightarrow} B. \qquad (6.4)$$

We can do this by drawing a dependency graph with an edge (C, D) whenever the grammar has a unit-production $C \to D$; then (6.4) holds whenever there is a walk between A and B. The new grammar $\widehat{G}$ is generated by first putting into $\widehat{P}$ all non-unit productions of P. Next, for all A and B satisfying (6.4), we add to $\widehat{P}$

$$A \to y_1 \,|y_2| \cdots |y_n,$$

where $B \to y_1 \,|y_2| \cdots y_n$ is the set of all rules in $\widehat{P}$ with B on the left. Note that since $B \to y_1 \,|y_2| \cdots |y_n$ is taken from $\widehat{P}$, none of the y_i can be a single variable, so that no unit-productions are created by the last step.

EXAMPLE 6.6

Remove all unit-productions from

$$\begin{aligned} S &\rightarrow Aa|B, \\ B &\rightarrow A|bb, \\ A &\rightarrow a\,|bc|\,B. \end{aligned}$$

The dependency graph for the unit-productions is given in Figure 6.3; we see from it that $S \overset{*}{\Rightarrow} A, S \overset{*}{\Rightarrow} B, B \overset{*}{\Rightarrow} A$, and $A \overset{*}{\Rightarrow} B$. Hence, we add to the original non-unit productions

$$\begin{aligned} S &\rightarrow Aa, \\ A &\rightarrow a|bc, \\ B &\rightarrow bb, \end{aligned}$$

the new rules

$$\begin{aligned} S &\rightarrow a\,|bc|\,bb, \\ A &\rightarrow bb, \\ B &\rightarrow a|bc, \end{aligned}$$

也可合并在一起：
S → Aa | a | bc | bb,
A → a | bc | bb,
B → a | bc | bb

补充例子

- 例：对下面文法进行化简：

  $S \to ABCDE|aB|fF$

  $A \to aBCA|BC|\lambda$

  $B \to b|bB|\lambda$

  $C \to c|cC|\lambda$

  $D \to d|dD|\lambda$

  $E \to e|eE|\lambda$

  $F \to fF$

  $G \to g|gG$

例子

- 例：对下面文法进行化简：

  $S \rightarrow ABCDE|aB|fF$
  $A \rightarrow aBCA|BC|\lambda$
  $B \rightarrow b|bB|\lambda$
  $C \rightarrow c|cC|\lambda$
  $D \rightarrow d|dD|\lambda$
  $E \rightarrow e|eE|\lambda$
  $F \rightarrow fF$
  $G \rightarrow g|gG$

1、去无用符号

（1）删除派生不出终极符号行的变量，F是无用符号，$V_1=\{S，A，B，C，D，E\}$

（2）删除不出现在任何句型中的语法符号，G，f，g为无用符号

$\widehat{V}=\{S，A，B，C，D，E\}$

$\widehat{T}=\{a,b,c,d,e\}$

化简结果：

$S \rightarrow ABCDE|aB$
$A \rightarrow aBCA|BC|\lambda$
$B \rightarrow b|bB|\lambda$
$C \rightarrow c|cC|\lambda$
$D \rightarrow d|dD|\lambda$
$E \rightarrow e|eE|\lambda$

例子

S→ABCDE|aB
A→aBCA|BC|λ
B→b|bB|λ
C→c|cC|λ
D→d|dD|λ
E→e|eE|λ

- 2、去空产生式

（1）求可空变量集

- V_N={S，A，B，C，D，E}

（2）先考虑对S的产生式进行去空产生式化简：

- S→ABCDE
- S→BCDE|ACDE|ABDE|ABCE|ABCD（一个变量用空替代）
- S→ABC|ABD|ABE|ACD|ACE|ADE|BCD|BCE|BDE|CDE（两个变量用空替代）
- S→AB|AC|AD|AE|BC|BD|BE|CD|CE|DE（三个变量用空替代）
- S→A|B|C|D|E（四个变量用空替代）
- S→aB|a
- S→λ (开始符号的空产生式单独保留)

（3）考虑对其他变量产生式进行去空产生式化简：

- A→aBCA|aCA|aBA|aBC|aB|aC|aA|BC|B|C|a
- B→b|bB　C→c|cC　D→d|dD　E→e|eE

例子

A→aBCA|aCA|aBA|aBC|aB|aC|aA|BC|B|C|a
B→b|bB　C→c|cC　D→d|dD　E→e|eE

- 3、去单一产生式
  - 在2中化简得到的产生式组中，S→A|B|C|D|E、A→B|C是单一产生式，下面将这种产生式去除。
- 首先考虑去掉A→B|C
    - A→aBCA|aCA|aBA|aBC|aB|aC|aA|BC|a
    - A→b|bB　A→c|cC
  - 合并：A→aBCA|aCA|aBA|aBC|aB|aC|aA|BC|a|b|bB|c|cC
- 再考虑去掉S→A|B|C|D|E
  - S→aBCA|aCA|aBA|aBC S→b|bB S→c|cC S→d|dD S→e|eE
- 合并:
  - S→ aBCA|aCA|aBA|aBC
  - S→a|b|c|d|e|aA|aB|aC|bB|cC|dD|eE|BC

例子

- 将化简后的产生式合并，最后化简文法为：
  - S→ABCDE|BCDE|ACDE|ABDE|ABCE|ABCD|ABC|ABD|ABE|ACD|ACE|ADE|BCD|BCE|BDE|CDE|AB|AC|AD|AE|BC|BD|BE|CD|CE|DE|a|b|c|d|e|aA|aB|aC|bB|cC|dD|eE|BC|aBCA|aCA|aBA|aBC|λ
  - A→aBCA|aCA|aBA|aBC|aB|aC|aA|BC|a|b|bB|c|cC
  - B→b|bB   C→c|cC
  - D→d|dD   E→e|eE
- 4、运用**THEOREM 6.2**的方法，发现无“无用符号”,以上文法就是原文法的化简结果

END