Formal Languages, Automata and Codes

Oleg Gutik

Lecture 17

We first raise an issue that is somewhat of a nuisance with grammars and languages in general: the presence of the empty word. The empty word plays a rather singular role in many theorems and proofs, and it is often necessary to give it special attention. We prefer to remove it from consideration altogether, looking only at languages that do not contain λ. In doing so, we do not lose generality, as we see from the following considerations. Let L be any context-free language, and let $G = (V, T, S, P)$ be a context-free grammar for $L - \{\lambda\}$. Then the grammar we obtain by adding to V the new variable S_0, making S_0 the start variable, and adding to P the productions

$$S_0 \to S|\lambda$$

generates L. Therefore, any nontrivial conclusion we can make for $L - \{\lambda\}$ will almost certainly transfer to L. Also, given any context-free grammar G, there is a method for obtaining $\widehat{G}$ such that $L(\widehat{G}) = L(G) - \{\lambda\}$. Consequently, for all practical purposes, there is no difference between context-free languages that include λ and those that do not. For the rest of this chapter, unless otherwise stated, we will restrict our discussion to λ-free languages.

We first raise an issue that is somewhat of a nuisance with grammars and languages in general: the presence of the empty word. The empty word plays a rather singular role in many theorems and proofs, and it is often necessary to give it special attention. We prefer to remove it from consideration altogether, looking only at languages that do not contain λ. In doing so, we do not lose generality, as we see from the following considerations. Let L be any context-free language, and let $G = (V, T, S, P)$ be a context-free grammar for $L - \{\lambda\}$. Then the grammar we obtain by adding to V the new variable S_0, making S_0 the start variable, and adding to P the productions

$$S_0 \to S|\lambda$$

generates L. Therefore, any nontrivial conclusion we can make for $L - \{\lambda\}$ will almost certainly transfer to L. Also, given any context-free grammar G, there is a method for obtaining $\widehat{G}$ such that $L(\widehat{G}) = L(G) - \{\lambda\}$. Consequently, for all practical purposes, there is no difference between context-free languages that include λ and those that do not. For the rest of this chapter, unless otherwise stated, we will restrict our discussion to λ-free languages.

We first raise an issue that is somewhat of a nuisance with grammars and languages in general: the presence of the empty word. The empty word plays a rather singular role in many theorems and proofs, and it is often necessary to give it special attention. We prefer to remove it from consideration altogether, looking only at languages that do not contain λ. In doing so, we do not lose generality, as we see from the following considerations. Let L be any context-free language, and let $G = (V, T, S, P)$ be a context-free grammar for $L - \{\lambda\}$. Then the grammar we obtain by adding to V the new variable S_0, making S_0 the start variable, and adding to P the productions

$$S_0 \to S|\lambda$$

generates L. Therefore, any nontrivial conclusion we can make for $L - \{\lambda\}$ will almost certainly transfer to L. Also, given any context-free grammar G, there is a method for obtaining $\widehat{G}$ such that $L(\widehat{G}) = L(G) - \{\lambda\}$. Consequently, for all practical purposes, there is no difference between context-free languages that include λ and those that do not. For the rest of this chapter, unless otherwise stated, we will restrict our discussion to λ-free languages.

We first raise an issue that is somewhat of a nuisance with grammars and languages in general: the presence of the empty word. The empty word plays a rather singular role in many theorems and proofs, and it is often necessary to give it special attention. We prefer to remove it from consideration altogether, looking only at languages that do not contain λ. In doing so, we do not lose generality, as we see from the following considerations. Let L be any context-free language, and let $G = (V, T, S, P)$ be a context-free grammar for $L - \{\lambda\}$. Then the grammar we obtain by adding to V the new variable S_0, making S_0 the start variable, and adding to P the productions

$$S_0 \rightarrow S|\lambda$$

generates L. Therefore, any nontrivial conclusion we can make for $L - \{\lambda\}$ will almost certainly transfer to L. Also, given any context-free grammar G, there is a method for obtaining $\widehat{G}$ such that $L(\widehat{G}) = L(G) - \{\lambda\}$. Consequently, for all practical purposes, there is no difference between context-free languages that include λ and those that do not. For the rest of this chapter, unless otherwise stated, we will restrict our discussion to λ-free languages.

We first raise an issue that is somewhat of a nuisance with grammars and languages in general: the presence of the empty word. The empty word plays a rather singular role in many theorems and proofs, and it is often necessary to give it special attention. We prefer to remove it from consideration altogether, looking only at languages that do not contain λ. In doing so, we do not lose generality, as we see from the following considerations. Let L be any context-free language, and let $G = (V, T, S, P)$ be a context-free grammar for $L - \{\lambda\}$. Then the grammar we obtain by adding to V the new variable S_0, making S_0 the start variable, and adding to P the productions

$$S_0 \to S|\lambda$$

generates L. Therefore, any nontrivial conclusion we can make for $L - \{\lambda\}$ will almost certainly transfer to L. Also, given any context-free grammar G, there is a method for obtaining $\widehat{G}$ such that $L(\widehat{G}) = L(G) - \{\lambda\}$. Consequently, for all practical purposes, there is no difference between context-free languages that include λ and those that do not. For the rest of this chapter, unless otherwise stated, we will restrict our discussion to λ-free languages.

We first raise an issue that is somewhat of a nuisance with grammars and languages in general: the presence of the empty word. The empty word plays a rather singular role in many theorems and proofs, and it is often necessary to give it special attention. We prefer to remove it from consideration altogether, looking only at languages that do not contain λ. In doing so, we do not lose generality, as we see from the following considerations. Let L be any context-free language, and let $G = (V, T, S, P)$ be a context-free grammar for $L - \{\lambda\}$. Then the grammar we obtain by adding to V the new variable S_0, making S_0 the start variable, and adding to P the productions

$$S_0 \to S|\lambda$$

generates L. Therefore, any nontrivial conclusion we can make for $L - \{\lambda\}$ will almost certainly transfer to L. Also, given any context-free grammar G, there is a method for obtaining $\widehat{G}$ such that $L(\widehat{G}) = L(G) - \{\lambda\}$. Consequently, for all practical purposes, there is no difference between context-free languages that include λ and those that do not. For the rest of this chapter, unless otherwise stated, we will restrict our discussion to λ-free languages.

We first raise an issue that is somewhat of a nuisance with grammars and languages in general: the presence of the empty word. The empty word plays a rather singular role in many theorems and proofs, and it is often necessary to give it special attention. We prefer to remove it from consideration altogether, looking only at languages that do not contain λ. In doing so, we do not lose generality, as we see from the following considerations. Let L be any context-free language, and let $G = (V, T, S, P)$ be a context-free grammar for $L - \{\lambda\}$. Then the grammar we obtain by adding to V the new variable S_0, making S_0 the start variable, and adding to P the productions

$$S_0 \to S|\lambda$$

generates L. Therefore, any nontrivial conclusion we can make for $L - \{\lambda\}$ will almost certainly transfer to L. Also, given any context-free grammar G, there is a method for obtaining $\widehat{G}$ such that $L(\widehat{G}) = L(G) - \{\lambda\}$. Consequently, for all practical purposes, there is no difference between context-free languages that include λ and those that do not. For the rest of this chapter, unless otherwise stated, we will restrict our discussion to λ-free languages.

We first raise an issue that is somewhat of a nuisance with grammars and languages in general: the presence of the empty word. The empty word plays a rather singular role in many theorems and proofs, and it is often necessary to give it special attention. We prefer to remove it from consideration altogether, looking only at languages that do not contain λ. In doing so, we do not lose generality, as we see from the following considerations. Let L be any context-free language, and let $G = (V, T, S, P)$ be a context-free grammar for $L - \{\lambda\}$. Then the grammar we obtain by adding to V the new variable S_0, making S_0 the start variable, and adding to P the productions

$$S_0 \to S|\lambda$$

generates L. Therefore, any nontrivial conclusion we can make for $L - \{\lambda\}$ will almost certainly transfer to L. Also, given any context-free grammar G, there is a method for obtaining $\widehat{G}$ such that $L(\widehat{G}) = L(G) - \{\lambda\}$. Consequently, for all practical purposes, there is no difference between context-free languages that include λ and those that do not. For the rest of this chapter, unless otherwise stated, we will restrict our discussion to λ-free languages.

We first raise an issue that is somewhat of a nuisance with grammars and languages in general: the presence of the empty word. The empty word plays a rather singular role in many theorems and proofs, and it is often necessary to give it special attention. We prefer to remove it from consideration altogether, looking only at languages that do not contain λ. In doing so, we do not lose generality, as we see from the following considerations. Let L be any context-free language, and let $G = (V, T, S, P)$ be a context-free grammar for $L - \{\lambda\}$. Then the grammar we obtain by adding to V the new variable S_0, making S_0 the start variable, and adding to P the productions

$$S_0 \to S|\lambda$$

generates L. Therefore, any nontrivial conclusion we can make for $L - \{\lambda\}$ will almost certainly transfer to L. Also, given any context-free grammar G, there is a method for obtaining $\widehat{G}$ such that $L(\widehat{G}) = L(G) - \{\lambda\}$. Consequently, for all practical purposes, there is no difference between context-free languages that include λ and those that do not. For the rest of this chapter, unless otherwise stated, we will restrict our discussion to λ-free languages.

We first raise an issue that is somewhat of a nuisance with grammars and languages in general: the presence of the empty word. The empty word plays a rather singular role in many theorems and proofs, and it is often necessary to give it special attention. We prefer to remove it from consideration altogether, looking only at languages that do not contain λ. In doing so, we do not lose generality, as we see from the following considerations. Let L be any context-free language, and let $G = (V, T, S, P)$ be a context-free grammar for $L - \{\lambda\}$. Then the grammar we obtain by adding to V the new variable S_0, making S_0 the start variable, and adding to P the productions

$$S_0 \to S|\lambda$$

generates L. Therefore, any nontrivial conclusion we can make for $L - \{\lambda\}$ will almost certainly transfer to L. Also, given any context-free grammar G, there is a method for obtaining $\widehat{G}$ such that $L(\widehat{G}) = L(G) - \{\lambda\}$. Consequently, for all practical purposes, there is no difference between context-free languages that include λ and those that do not. For the rest of this chapter, unless otherwise stated, we will restrict our discussion to λ-free languages.

We first raise an issue that is somewhat of a nuisance with grammars and languages in general: the presence of the empty word. The empty word plays a rather singular role in many theorems and proofs, and it is often necessary to give it special attention. We prefer to remove it from consideration altogether, looking only at languages that do not contain λ. In doing so, we do not lose generality, as we see from the following considerations. Let L be any context-free language, and let $G = (V, T, S, P)$ be a context-free grammar for $L - \{\lambda\}$. Then the grammar we obtain by adding to V the new variable S_0, making S_0 the start variable, and adding to P the productions

$$S_0 \to S|\lambda$$

generates L. Therefore, any nontrivial conclusion we can make for $L - \{\lambda\}$ will almost certainly transfer to L. Also, given any context-free grammar G, there is a method for obtaining $\widehat{G}$ such that $L(\widehat{G}) = L(G) - \{\lambda\}$. Consequently, for all practical purposes, there is no difference between context-free languages that include λ and those that do not. For the rest of this chapter, unless otherwise stated, we will restrict our discussion to λ-free languages.

We first raise an issue that is somewhat of a nuisance with grammars and languages in general: the presence of the empty word. The empty word plays a rather singular role in many theorems and proofs, and it is often necessary to give it special attention. We prefer to remove it from consideration altogether, looking only at languages that do not contain λ. In doing so, we do not lose generality, as we see from the following considerations. Let L be any context-free language, and let $G = (V, T, S, P)$ be a context-free grammar for $L - \{\lambda\}$. Then the grammar we obtain by adding to V the new variable S_0, making S_0 the start variable, and adding to P the productions

$$S_0 \to S|\lambda$$

generates L. Therefore, any nontrivial conclusion we can make for $L - \{\lambda\}$ will almost certainly transfer to L. Also, given any context-free grammar G, there is a method for obtaining $\widehat{G}$ such that $L(\widehat{G}) = L(G) - \{\lambda\}$. Consequently, for all practical purposes, there is no difference between context-free languages that include λ and those that do not. For the rest of this chapter, unless otherwise stated, we will restrict our discussion to λ-free languages.

We first raise an issue that is somewhat of a nuisance with grammars and languages in general: the presence of the empty word. The empty word plays a rather singular role in many theorems and proofs, and it is often necessary to give it special attention. We prefer to remove it from consideration altogether, looking only at languages that do not contain λ. In doing so, we do not lose generality, as we see from the following considerations. Let L be any context-free language, and let $G = (V, T, S, P)$ be a context-free grammar for $L - \{\lambda\}$. Then the grammar we obtain by adding to V the new variable S_0, making S_0 the start variable, and adding to P the productions

$$S_0 \to S|\lambda$$

generates L. Therefore, any nontrivial conclusion we can make for $L - \{\lambda\}$ will almost certainly transfer to L. Also, given any context-free grammar G, there is a method for obtaining $\widehat{G}$ such that $L(\widehat{G}) = L(G) - \{\lambda\}$. Consequently, for all practical purposes, there is no difference between context-free languages that include λ and those that do not. For the rest of this chapter, unless otherwise stated, we will restrict our discussion to λ-free languages.

We first raise an issue that is somewhat of a nuisance with grammars and languages in general: the presence of the empty word. The empty word plays a rather singular role in many theorems and proofs, and it is often necessary to give it special attention. We prefer to remove it from consideration altogether, looking only at languages that do not contain λ. In doing so, we do not lose generality, as we see from the following considerations. Let L be any context-free language, and let $G = (V, T, S, P)$ be a context-free grammar for $L - \{\lambda\}$. Then the grammar we obtain by adding to V the new variable S_0, making S_0 the start variable, and adding to P the productions

$$S_0 \to S|\lambda$$

generates L. Therefore, any nontrivial conclusion we can make for $L - \{\lambda\}$ will almost certainly transfer to L. Also, given any context-free grammar G, there is a method for obtaining $\widehat{G}$ such that $L(\widehat{G}) = L(G) - \{\lambda\}$. Consequently, for all practical purposes, there is no difference between context-free languages that include λ and those that do not. For the rest of this chapter, unless otherwise stated, we will restrict our discussion to λ-free languages.

We first raise an issue that is somewhat of a nuisance with grammars and languages in general: the presence of the empty word. The empty word plays a rather singular role in many theorems and proofs, and it is often necessary to give it special attention. We prefer to remove it from consideration altogether, looking only at languages that do not contain λ. In doing so, we do not lose generality, as we see from the following considerations. Let L be any context-free language, and let $G = (V, T, S, P)$ be a context-free grammar for $L - \{\lambda\}$. Then the grammar we obtain by adding to V the new variable S_0, making S_0 the start variable, and adding to P the productions

$$S_0 \to S|\lambda$$

generates L. Therefore, any nontrivial conclusion we can make for $L - \{\lambda\}$ will almost certainly transfer to L. Also, given any context-free grammar G, there is a method for obtaining $\widehat{G}$ such that $L(\widehat{G}) = L(G) - \{\lambda\}$. Consequently, for all practical purposes, there is no difference between context-free languages that include λ and those that do not. For the rest of this chapter, unless otherwise stated, we will restrict our discussion to λ-free languages.

We first raise an issue that is somewhat of a nuisance with grammars and languages in general: the presence of the empty word. The empty word plays a rather singular role in many theorems and proofs, and it is often necessary to give it special attention. We prefer to remove it from consideration altogether, looking only at languages that do not contain λ. In doing so, we do not lose generality, as we see from the following considerations. Let L be any context-free language, and let $G = (V, T, S, P)$ be a context-free grammar for $L - \{\lambda\}$. Then the grammar we obtain by adding to V the new variable S_0, making S_0 the start variable, and adding to P the productions

$$S_0 \to S|\lambda$$

generates L. Therefore, any nontrivial conclusion we can make for $L - \{\lambda\}$ will almost certainly transfer to L. Also, given any context-free grammar G, there is a method for obtaining $\widehat{G}$ such that $L(\widehat{G}) = L(G) - \{\lambda\}$. Consequently, for all practical purposes, there is no difference between context-free languages that include λ and those that do not. For the rest of this chapter, unless otherwise stated, we will restrict our discussion to λ-free languages.

We first raise an issue that is somewhat of a nuisance with grammars and languages in general: the presence of the empty word. The empty word plays a rather singular role in many theorems and proofs, and it is often necessary to give it special attention. We prefer to remove it from consideration altogether, looking only at languages that do not contain λ. In doing so, we do not lose generality, as we see from the following considerations. Let L be any context-free language, and let $G = (V, T, S, P)$ be a context-free grammar for $L - \{\lambda\}$. Then the grammar we obtain by adding to V the new variable S_0, making S_0 the start variable, and adding to P the productions

$$S_0 \to S|\lambda$$

generates L. Therefore, any nontrivial conclusion we can make for $L - \{\lambda\}$ will almost certainly transfer to L. Also, given any context-free grammar G, there is a method for obtaining $\widehat{G}$ such that $L(\widehat{G}) = L(G) - \{\lambda\}$. Consequently, for all practical purposes, there is no difference between context-free languages that include λ and those that do not. For the rest of this chapter, unless otherwise stated, we will restrict our discussion to λ-free languages.

We first raise an issue that is somewhat of a nuisance with grammars and languages in general: the presence of the empty word. The empty word plays a rather singular role in many theorems and proofs, and it is often necessary to give it special attention. We prefer to remove it from consideration altogether, looking only at languages that do not contain λ. In doing so, we do not lose generality, as we see from the following considerations. Let L be any context-free language, and let $G = (V, T, S, P)$ be a context-free grammar for $L - \{\lambda\}$. Then the grammar we obtain by adding to V the new variable S_0, making S_0 the start variable, and adding to P the productions

$$S_0 \to S|\lambda$$

generates L. Therefore, any nontrivial conclusion we can make for $L - \{\lambda\}$ will almost certainly transfer to L. Also, given any context-free grammar G, there is a method for obtaining $\widehat{G}$ such that $L(\widehat{G}) = L(G) - \{\lambda\}$. Consequently, for all practical purposes, there is no difference between context-free languages that include λ and those that do not. For the rest of this chapter, unless otherwise stated, we will restrict our discussion to λ-free languages.

We first raise an issue that is somewhat of a nuisance with grammars and languages in general: the presence of the empty word. The empty word plays a rather singular role in many theorems and proofs, and it is often necessary to give it special attention. We prefer to remove it from consideration altogether, looking only at languages that do not contain λ. In doing so, we do not lose generality, as we see from the following considerations. Let L be any context-free language, and let $G = (V, T, S, P)$ be a context-free grammar for $L - \{\lambda\}$. Then the grammar we obtain by adding to V the new variable S_0, making S_0 the start variable, and adding to P the productions

$$S_0 \to S|\lambda$$

generates L. Therefore, any nontrivial conclusion we can make for $L - \{\lambda\}$ will almost certainly transfer to L. Also, given any context-free grammar G, there is a method for obtaining $\widehat{G}$ such that $L(\widehat{G}) = L(G) - \{\lambda\}$. Consequently, for all practical purposes, there is no difference between context-free languages that include λ and those that do not. For the rest of this chapter, unless otherwise stated, we will restrict our discussion to λ-free languages.

We first raise an issue that is somewhat of a nuisance with grammars and languages in general: the presence of the empty word. The empty word plays a rather singular role in many theorems and proofs, and it is often necessary to give it special attention. We prefer to remove it from consideration altogether, looking only at languages that do not contain λ. In doing so, we do not lose generality, as we see from the following considerations. Let L be any context-free language, and let $G = (V, T, S, P)$ be a context-free grammar for $L - \{\lambda\}$. Then the grammar we obtain by adding to V the new variable S_0, making S_0 the start variable, and adding to P the productions

$$S_0 \to S|\lambda$$

generates L. Therefore, any nontrivial conclusion we can make for $L - \{\lambda\}$ will almost certainly transfer to L. Also, given any context-free grammar G, there is a method for obtaining $\widehat{G}$ such that $L(\widehat{G}) = L(G) - \{\lambda\}$. Consequently, for all practical purposes, there is no difference between context-free languages that include λ and those that do not. For the rest of this chapter, unless otherwise stated, we will restrict our discussion to λ-free languages.

A Useful Substitution Rule

Many rules govern generating equivalent grammars by means of substitutions. Here we give one that is very useful for simplifying grammars in various ways. We shall not define the term simplification precisely, but we shall use it nevertheless. What we mean by it is the removal of certain types of undesirable productions; the process does not necessarily result in an actual reduction of the number of rules.

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

$$A \to x_1 B x_2 \tag{1}$$

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

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

A Useful Substitution Rule

Many rules govern generating equivalent grammars by means of substitutions. Here we give one that is very useful for simplifying grammars in various ways. We shall not define the term simplification precisely, but we shall use it nevertheless. What we mean by it is the removal of certain types of undesirable productions; the process does not necessarily result in an actual reduction of the number of rules.

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

$$A \to x_1 B x_2 \tag{1}$$

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

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

A Useful Substitution Rule

Many rules govern generating equivalent grammars by means of substitutions. Here we give one that is very useful for simplifying grammars in various ways. We shall not define the term simplification precisely, but we shall use it nevertheless. What we mean by it is the removal of certain types of undesirable productions; the process does not necessarily result in an actual reduction of the number of rules.

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

$$A \to x_1 B x_2 \tag{1}$$

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

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

A Useful Substitution Rule

Many rules govern generating equivalent grammars by means of substitutions. Here we give one that is very useful for simplifying grammars in various ways. We shall not define the term simplification precisely, but we shall use it nevertheless. What we mean by it is the removal of certain types of undesirable productions; the process does not necessarily result in an actual reduction of the number of rules.

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

$$A \to x_1 B x_2 \qquad (1)$$

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

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

A Useful Substitution Rule

Many rules govern generating equivalent grammars by means of substitutions. Here we give one that is very useful for simplifying grammars in various ways. We shall not define the term simplification precisely, but we shall use it nevertheless. What we mean by it is the removal of certain types of undesirable productions; the process does not necessarily result in an actual reduction of the number of rules.

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

$$A \to x_1 B x_2 \tag{1}$$

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

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

A Useful Substitution Rule

Many rules govern generating equivalent grammars by means of substitutions. Here we give one that is very useful for simplifying grammars in various ways. We shall not define the term simplification precisely, but we shall use it nevertheless. What we mean by it is the removal of certain types of undesirable productions; the process does not necessarily result in an actual reduction of the number of rules.

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

$$A \to x_1 B x_2 \tag{1}$$

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

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

A Useful Substitution Rule

Many rules govern generating equivalent grammars by means of substitutions. Here we give one that is very useful for simplifying grammars in various ways. We shall not define the term simplification precisely, but we shall use it nevertheless. What we mean by it is the removal of certain types of undesirable productions; the process does not necessarily result in an actual reduction of the number of rules.

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

$$A \to x_1 B x_2 \tag{1}$$

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

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

A Useful Substitution Rule

Many rules govern generating equivalent grammars by means of substitutions. Here we give one that is very useful for simplifying grammars in various ways. We shall not define the term simplification precisely, but we shall use it nevertheless. What we mean by it is the removal of certain types of undesirable productions; the process does not necessarily result in an actual reduction of the number of rules.

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

$$A \to x_1 B x_2 \tag{1}$$

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

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

A Useful Substitution Rule

Many rules govern generating equivalent grammars by means of substitutions. Here we give one that is very useful for simplifying grammars in various ways. We shall not define the term simplification precisely, but we shall use it nevertheless. What we mean by it is the removal of certain types of undesirable productions; the process does not necessarily result in an actual reduction of the number of rules.

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

$$A \to x_1 B x_2 \tag{1}$$

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

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

A Useful Substitution Rule

Many rules govern generating equivalent grammars by means of substitutions. Here we give one that is very useful for simplifying grammars in various ways. We shall not define the term simplification precisely, but we shall use it nevertheless. What we mean by it is the removal of certain types of undesirable productions; the process does not necessarily result in an actual reduction of the number of rules.

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

$$A \to x_1 B x_2 \tag{1}$$

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

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

A Useful Substitution Rule

Many rules govern generating equivalent grammars by means of substitutions. Here we give one that is very useful for simplifying grammars in various ways. We shall not define the term simplification precisely, but we shall use it nevertheless. What we mean by it is the removal of certain types of undesirable productions; the process does not necessarily result in an actual reduction of the number of rules.

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

$$A \to x_1 B x_2 \tag{1}$$

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

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

A Useful Substitution Rule

Many rules govern generating equivalent grammars by means of substitutions. Here we give one that is very useful for simplifying grammars in various ways. We shall not define the term simplification precisely, but we shall use it nevertheless. What we mean by it is the removal of certain types of undesirable productions; the process does not necessarily result in an actual reduction of the number of rules.

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

$$A \to x_1 B x_2 \tag{1}$$

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

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

A Useful Substitution Rule

Many rules govern generating equivalent grammars by means of substitutions. Here we give one that is very useful for simplifying grammars in various ways. We shall not define the term simplification precisely, but we shall use it nevertheless. What we mean by it is the removal of certain types of undesirable productions; the process does not necessarily result in an actual reduction of the number of rules.

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

$$A \to x_1 B x_2 \tag{1}$$

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

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

A Useful Substitution Rule

Many rules govern generating equivalent grammars by means of substitutions. Here we give one that is very useful for simplifying grammars in various ways. We shall not define the term simplification precisely, but we shall use it nevertheless. What we mean by it is the removal of certain types of undesirable productions; the process does not necessarily result in an actual reduction of the number of rules.

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

$$A \to x_1 B x_2 \tag{1}$$

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

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

A Useful Substitution Rule

Many rules govern generating equivalent grammars by means of substitutions. Here we give one that is very useful for simplifying grammars in various ways. We shall not define the term simplification precisely, but we shall use it nevertheless. What we mean by it is the removal of certain types of undesirable productions; the process does not necessarily result in an actual reduction of the number of rules.

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

$$A \to x_1 B x_2 \tag{1}$$

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

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

A Useful Substitution Rule

Many rules govern generating equivalent grammars by means of substitutions. Here we give one that is very useful for simplifying grammars in various ways. We shall not define the term simplification precisely, but we shall use it nevertheless. What we mean by it is the removal of certain types of undesirable productions; the process does not necessarily result in an actual reduction of the number of rules.

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

$$A \to x_1 B x_2 \tag{1}$$

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

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

A Useful Substitution Rule

Many rules govern generating equivalent grammars by means of substitutions. Here we give one that is very useful for simplifying grammars in various ways. We shall not define the term simplification precisely, but we shall use it nevertheless. What we mean by it is the removal of certain types of undesirable productions; the process does not necessarily result in an actual reduction of the number of rules.

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

$$A \to x_1 B x_2 \tag{1}$$

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

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

A Useful Substitution Rule

Many rules govern generating equivalent grammars by means of substitutions. Here we give one that is very useful for simplifying grammars in various ways. We shall not define the term simplification precisely, but we shall use it nevertheless. What we mean by it is the removal of certain types of undesirable productions; the process does not necessarily result in an actual reduction of the number of rules.

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

$$A \to x_1 B x_2 \tag{1}$$

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

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

A Useful Substitution Rule

Many rules govern generating equivalent grammars by means of substitutions. Here we give one that is very useful for simplifying grammars in various ways. We shall not define the term simplification precisely, but we shall use it nevertheless. What we mean by it is the removal of certain types of undesirable productions; the process does not necessarily result in an actual reduction of the number of rules.

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

$$A \to x_1 B x_2 \tag{1}$$

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

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

A Useful Substitution Rule

Many rules govern generating equivalent grammars by means of substitutions. Here we give one that is very useful for simplifying grammars in various ways. We shall not define the term simplification precisely, but we shall use it nevertheless. What we mean by it is the removal of certain types of undesirable productions; the process does not necessarily result in an actual reduction of the number of rules.

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

$$A \to x_1 B x_2 \tag{1}$$

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

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

A Useful Substitution Rule

Many rules govern generating equivalent grammars by means of substitutions. Here we give one that is very useful for simplifying grammars in various ways. We shall not define the term simplification precisely, but we shall use it nevertheless. What we mean by it is the removal of certain types of undesirable productions; the process does not necessarily result in an actual reduction of the number of rules.

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

$$A \to x_1 B x_2 \tag{1}$$

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

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

Proof. Suppose that $w \in L(G)$, so that
$$S \stackrel{*}{\Rightarrow}_G w.$$
The subscript on the derivation sign $\Rightarrow$ is used here to distinguish between derivations with different grammars. If this derivation does not involve the production (1)
$$A \to x_1 B x_2, \tag{1}$$
then obviously
$$S \stackrel{*}{\Rightarrow}_{\widehat{G}} w.$$
If it does, then look at the derivation the first time (6.1) is used. The B so introduced eventually has to be replaced; we lose nothing by assuming that this is done immediately. Thus
$$S \stackrel{*}{\Rightarrow}_G u_1 A u_2 \Rightarrow_G u_1 x_1 B x_2 u_2 \Rightarrow_G u_1 x_1 y_j x_2 u_2.$$
But with grammar $\widehat{G}$ we can get
$$S \stackrel{*}{\Rightarrow}_{\widehat{G}} u_1 A u_2 \Rightarrow_{\widehat{G}} u_1 x_1 y_j x_2 u_2.$$
Thus we can reach the same sentential form with G and $\widehat{G}$. If (1) is used again later, we can repeat the argument. It follows then, by induction on the number of times the production is applied, that
$$S \stackrel{*}{\Rightarrow}_{\widehat{G}} w.$$
Therefore, if $w \in L(G)$, then $w \in L(\widehat{G})$.

By similar reasoning, we can show that if $w \in L(\widehat{G})$, then $w \in L(G)$, completing the proof. ■

Proof. Suppose that $w \in L(G)$, so that

$$S \overset{*}{\Rightarrow}_G w.$$

The subscript on the derivation sign $\Rightarrow$ is used here to distinguish between derivations with different grammars. If this derivation does not involve the production (1)

$$A \to x_1 B x_2, \tag{1}$$

then obviously

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

If it does, then look at the derivation the first time (6.1) is used. The B so introduced eventually has to be replaced; we lose nothing by assuming that this is done immediately. Thus

$$S \overset{*}{\Rightarrow}_G u_1 A u_2 \Rightarrow_G u_1 x_1 B x_2 u_2 \Rightarrow_G u_1 x_1 y_j x_2 u_2.$$

But with grammar $\widehat{G}$ we can get

$$S \overset{*}{\Rightarrow}_{\widehat{G}} u_1 A u_2 \Rightarrow_{\widehat{G}} u_1 x_1 y_j x_2 u_2.$$

Thus we can reach the same sentential form with G and $\widehat{G}$. If (1) is used again later, we can repeat the argument. It follows then, by induction on the number of times the production is applied, that

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

Therefore, if $w \in L(G)$, then $w \in L(\widehat{G})$.

By similar reasoning, we can show that if $w \in L(\widehat{G})$, then $w \in L(G)$, completing the proof. ■

Proof. Suppose that $w \in L(G)$, so that

$$S \overset{*}{\Rightarrow}_G w.$$

The subscript on the derivation sign $\Rightarrow$ is used here to distinguish between derivations with different grammars. If this derivation does not involve the production (1)

$$A \to x_1 B x_2, \tag{1}$$

then obviously

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

If it does, then look at the derivation the first time (6.1) is used. The B so introduced eventually has to be replaced; we lose nothing by assuming that this is done immediately. Thus

$$S \overset{*}{\Rightarrow}_G u_1 A u_2 \Rightarrow_G u_1 x_1 B x_2 u_2 \Rightarrow_G u_1 x_1 y_j x_2 u_2.$$

But with grammar $\widehat{G}$ we can get

$$S \overset{*}{\Rightarrow}_{\widehat{G}} u_1 A u_2 \Rightarrow_{\widehat{G}} u_1 x_1 y_j x_2 u_2.$$

Thus we can reach the same sentential form with G and $\widehat{G}$. If (1) is used again later, we can repeat the argument. It follows then, by induction on the number of times the production is applied, that

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

Therefore, if $w \in L(G)$, then $w \in L(\widehat{G})$.

By similar reasoning, we can show that if $w \in L(\widehat{G})$, then $w \in L(G)$, completing the proof. ∎

Proof. Suppose that $w \in L(G)$, so that

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

The subscript on the derivation sign $\Rightarrow$ is used here to distinguish between derivations with different grammars. If this derivation does not involve the production (1)

$$A \to x_1 B x_2, \tag{1}$$

then obviously

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

If it does, then look at the derivation the first time (6.1) is used. The B so introduced eventually has to be replaced; we lose nothing by assuming that this is done immediately. Thus

$$S \stackrel{*}{\Rightarrow}_G u_1 A u_2 \Rightarrow_G u_1 x_1 B x_2 u_2 \Rightarrow_G u_1 x_1 y_j x_2 u_2.$$

But with grammar $\widehat{G}$ we can get

$$S \stackrel{*}{\Rightarrow}_{\widehat{G}} u_1 A u_2 \Rightarrow_{\widehat{G}} u_1 x_1 y_j x_2 u_2.$$

Thus we can reach the same sentential form with G and $\widehat{G}$. If (1) is used again later, we can repeat the argument. It follows then, by induction on the number of times the production is applied, that

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

Therefore, if $w \in L(G)$, then $w \in L(\widehat{G})$.

By similar reasoning, we can show that if $w \in L(\widehat{G})$, then $w \in L(G)$, completing the proof. ■

Proof. Suppose that $w \in L(G)$, so that

$$S \overset{*}{\Rightarrow}_G w.$$

The subscript on the derivation sign $\Rightarrow$ is used here to distinguish between derivations with different grammars. If this derivation does not involve the production (1)

$$A \to x_1 B x_2, \tag{1}$$

then obviously

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

If it does, then look at the derivation the first time (6.1) is used. The B so introduced eventually has to be replaced; we lose nothing by assuming that this is done immediately. Thus

$$S \overset{*}{\Rightarrow}_G u_1 A u_2 \Rightarrow_G u_1 x_1 B x_2 u_2 \Rightarrow_G u_1 x_1 y_j x_2 u_2.$$

But with grammar $\widehat{G}$ we can get

$$S \overset{*}{\Rightarrow}_{\widehat{G}} u_1 A u_2 \Rightarrow_{\widehat{G}} u_1 x_1 y_j x_2 u_2.$$

Thus we can reach the same sentential form with G and $\widehat{G}$. If (1) is used again later, we can repeat the argument. It follows then, by induction on the number of times the production is applied, that

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

Therefore, if $w \in L(G)$, then $w \in L(\widehat{G})$.

By similar reasoning, we can show that if $w \in L(\widehat{G})$, then $w \in L(G)$, completing the proof. ∎

Proof. Suppose that $w \in L(G)$, so that

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

The subscript on the derivation sign $\Rightarrow$ is used here to distinguish between derivations with different grammars. If this derivation does not involve the production (1)

$$A \to x_1 B x_2, \tag{1}$$

then obviously

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

If it does, then look at the derivation the first time (6.1) is used. The B so introduced eventually has to be replaced; we lose nothing by assuming that this is done immediately. Thus

$$S \stackrel{*}{\Rightarrow}_G u_1 A u_2 \Rightarrow_G u_1 x_1 B x_2 u_2 \Rightarrow_G u_1 x_1 y_j x_2 u_2.$$

But with grammar $\widehat{G}$ we can get

$$S \stackrel{*}{\Rightarrow}_{\widehat{G}} u_1 A u_2 \Rightarrow_{\widehat{G}} u_1 x_1 y_j x_2 u_2.$$

Thus we can reach the same sentential form with G and $\widehat{G}$. If (1) is used again later, we can repeat the argument. It follows then, by induction on the number of times the production is applied, that

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

Therefore, if $w \in L(G)$, then $w \in L(\widehat{G})$.

By similar reasoning, we can show that if $w \in L(\widehat{G})$, then $w \in L(G)$, completing the proof. ■

Proof. Suppose that $w \in L(G)$, so that

$$S \overset{*}{\Rightarrow}_G w.$$

The subscript on the derivation sign $\Rightarrow$ is used here to distinguish between derivations with different grammars. If this derivation does not involve the production (1)

$$A \to x_1 B x_2, \tag{1}$$

then obviously

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

If it does, then look at the derivation the first time (6.1) is used. The B so introduced eventually has to be replaced; we lose nothing by assuming that this is done immediately. Thus

$$S \overset{*}{\Rightarrow}_G u_1 A u_2 \Rightarrow_G u_1 x_1 B x_2 u_2 \Rightarrow_G u_1 x_1 y_j x_2 u_2.$$

But with grammar $\widehat{G}$ we can get

$$S \overset{*}{\Rightarrow}_{\widehat{G}} u_1 A u_2 \Rightarrow_{\widehat{G}} u_1 x_1 y_j x_2 u_2.$$

Thus we can reach the same sentential form with G and $\widehat{G}$. If (1) is used again later, we can repeat the argument. It follows then, by induction on the number of times the production is applied, that

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

Therefore, if $w \in L(G)$, then $w \in L(\widehat{G})$.

By similar reasoning, we can show that if $w \in L(\widehat{G})$, then $w \in L(G)$, completing the proof. ■

Proof. Suppose that $w \in L(G)$, so that

$$S \overset{*}{\Rightarrow}_G w.$$

The subscript on the derivation sign $\Rightarrow$ is used here to distinguish between derivations with different grammars. If this derivation does not involve the production (1)

$$A \to x_1 B x_2, \tag{1}$$

then obviously

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

If it does, then look at the derivation the first time (6.1) is used. The B so introduced eventually has to be replaced; we lose nothing by assuming that this is done immediately. Thus

$$S \overset{*}{\Rightarrow}_G u_1 A u_2 \Rightarrow_G u_1 x_1 B x_2 u_2 \Rightarrow_G u_1 x_1 y_j x_2 u_2.$$

But with grammar $\widehat{G}$ we can get

$$S \overset{*}{\Rightarrow}_{\widehat{G}} u_1 A u_2 \Rightarrow_{\widehat{G}} u_1 x_1 y_j x_2 u_2.$$

Thus we can reach the same sentential form with G and $\widehat{G}$. If (1) is used again later, we can repeat the argument. It follows then, by induction on the number of times the production is applied, that

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

Therefore, if $w \in L(G)$, then $w \in L(\widehat{G})$.

By similar reasoning, we can show that if $w \in L(\widehat{G})$, then $w \in L(G)$, completing the proof. ∎

Proof. Suppose that $w \in L(G)$, so that

$$S \overset{*}{\Rightarrow}_G w.$$

The subscript on the derivation sign $\Rightarrow$ is used here to distinguish between derivations with different grammars. If this derivation does not involve the production (1)

$$A \to x_1 B x_2, \tag{1}$$

then obviously

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

If it does, then look at the derivation the first time (6.1) is used. The B so introduced eventually has to be replaced; we lose nothing by assuming that this is done immediately. Thus

$$S \overset{*}{\Rightarrow}_G u_1 A u_2 \Rightarrow_G u_1 x_1 B x_2 u_2 \Rightarrow_G u_1 x_1 y_j x_2 u_2.$$

But with grammar $\widehat{G}$ we can get

$$S \overset{*}{\Rightarrow}_{\widehat{G}} u_1 A u_2 \Rightarrow_{\widehat{G}} u_1 x_1 y_j x_2 u_2.$$

Thus we can reach the same sentential form with G and $\widehat{G}$. If (1) is used again later, we can repeat the argument. It follows then, by induction on the number of times the production is applied, that

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

Therefore, if $w \in L(G)$, then $w \in L(\widehat{G})$.

By similar reasoning, we can show that if $w \in L(\widehat{G})$, then $w \in L(G)$, completing the proof. ∎

Proof. Suppose that $w \in L(G)$, so that

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

The subscript on the derivation sign $\Rightarrow$ is used here to distinguish between derivations with different grammars. If this derivation does not involve the production (1)

$$A \to x_1 B x_2, \tag{1}$$

then obviously

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

If it does, then look at the derivation the first time (6.1) is used. The B so introduced eventually has to be replaced; we lose nothing by assuming that this is done immediately. Thus

$$S \stackrel{*}{\Rightarrow}_G u_1 A u_2 \Rightarrow_G u_1 x_1 B x_2 u_2 \Rightarrow_G u_1 x_1 y_j x_2 u_2.$$

But with grammar $\widehat{G}$ we can get

$$S \stackrel{*}{\Rightarrow}_{\widehat{G}} u_1 A u_2 \Rightarrow_{\widehat{G}} u_1 x_1 y_j x_2 u_2.$$

Thus we can reach the same sentential form with G and $\widehat{G}$. If (1) is used again later, we can repeat the argument. It follows then, by induction on the number of times the production is applied, that

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

Therefore, if $w \in L(G)$, then $w \in L(\widehat{G})$.

By similar reasoning, we can show that if $w \in L(\widehat{G})$, then $w \in L(G)$, completing the proof. ■

6.1 Methods for Transforming Grammars

Proof. Suppose that $w \in L(G)$, so that

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

The subscript on the derivation sign $\Rightarrow$ is used here to distinguish between derivations with different grammars. If this derivation does not involve the production (1)

$$A \to x_1 B x_2, \tag{1}$$

then obviously

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

If it does, then look at the derivation the first time (6.1) is used. The B so introduced eventually has to be replaced; we lose nothing by assuming that this is done immediately. Thus

$$S \stackrel{*}{\Rightarrow}_G u_1 A u_2 \Rightarrow_G u_1 x_1 B x_2 u_2 \Rightarrow_G u_1 x_1 y_j x_2 u_2.$$

But with grammar $\widehat{G}$ we can get

$$S \stackrel{*}{\Rightarrow}_{\widehat{G}} u_1 A u_2 \Rightarrow_{\widehat{G}} u_1 x_1 y_j x_2 u_2.$$

Thus we can reach the same sentential form with G and $\widehat{G}$. If (1) is used again later, we can repeat the argument. It follows then, by induction on the number of times the production is applied, that

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

Therefore, if $w \in L(G)$, then $w \in L(\widehat{G})$.

By similar reasoning, we can show that if $w \in L(\widehat{G})$, then $w \in L(G)$, completing the proof. ■

Proof. Suppose that $w \in L(G)$, so that

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

The subscript on the derivation sign $\Rightarrow$ is used here to distinguish between derivations with different grammars. If this derivation does not involve the production (1)

$$A \to x_1 B x_2, \quad (1)$$

then obviously

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

If it does, then look at the derivation the first time (6.1) is used. The B so introduced eventually has to be replaced; we lose nothing by assuming that this is done immediately. Thus

$$S \stackrel{*}{\Rightarrow}_G u_1 A u_2 \Rightarrow_G u_1 x_1 B x_2 u_2 \Rightarrow_G u_1 x_1 y_j x_2 u_2.$$

But with grammar $\widehat{G}$ we can get

$$S \stackrel{*}{\Rightarrow}_{\widehat{G}} u_1 A u_2 \Rightarrow_{\widehat{G}} u_1 x_1 y_j x_2 u_2.$$

Thus we can reach the same sentential form with G and $\widehat{G}$. If (1) is used again later, we can repeat the argument. It follows then, by induction on the number of times the production is applied, that

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

Therefore, if $w \in L(G)$, then $w \in L(\widehat{G})$.

By similar reasoning, we can show that if $w \in L(\widehat{G})$, then $w \in L(G)$, completing the proof. ■

Proof. Suppose that $w \in L(G)$, so that

$$S \overset{*}{\Rightarrow}_G w.$$

The subscript on the derivation sign $\Rightarrow$ is used here to distinguish between derivations with different grammars. If this derivation does not involve the production (1)

$$A \to x_1 B x_2, \tag{1}$$

then obviously

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

If it does, then look at the derivation the first time (6.1) is used. The B so introduced eventually has to be replaced; we lose nothing by assuming that this is done immediately. Thus

$$S \overset{*}{\Rightarrow}_G u_1 A u_2 \Rightarrow_G u_1 x_1 B x_2 u_2 \Rightarrow_G u_1 x_1 y_j x_2 u_2.$$

But with grammar $\widehat{G}$ we can get

$$S \overset{*}{\Rightarrow}_{\widehat{G}} u_1 A u_2 \Rightarrow_{\widehat{G}} u_1 x_1 y_j x_2 u_2.$$

Thus we can reach the same sentential form with G and $\widehat{G}$. If (1) is used again later, we can repeat the argument. It follows then, by induction on the number of times the production is applied, that

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

Therefore, if $w \in L(G)$, then $w \in L(\widehat{G})$.

By similar reasoning, we can show that if $w \in L(\widehat{G})$, then $w \in L(G)$, completing the proof. ∎

Proof. Suppose that $w \in L(G)$, so that

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

The subscript on the derivation sign $\Rightarrow$ is used here to distinguish between derivations with different grammars. If this derivation does not involve the production (1)

$$A \to x_1 B x_2, \tag{1}$$

then obviously

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

If it does, then look at the derivation the first time (6.1) is used. The B so introduced eventually has to be replaced; we lose nothing by assuming that this is done immediately. Thus

$$S \stackrel{*}{\Rightarrow}_G u_1 A u_2 \Rightarrow_G u_1 x_1 B x_2 u_2 \Rightarrow_G u_1 x_1 y_j x_2 u_2.$$

But with grammar $\widehat{G}$ we can get

$$S \stackrel{*}{\Rightarrow}_{\widehat{G}} u_1 A u_2 \Rightarrow_{\widehat{G}} u_1 x_1 y_j x_2 u_2.$$

Thus we can reach the same sentential form with G and $\widehat{G}$. If (1) is used again later, we can repeat the argument. It follows then, by induction on the number of times the production is applied, that

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

Therefore, if $w \in L(G)$, then $w \in L(\widehat{G})$.

By similar reasoning, we can show that if $w \in L(\widehat{G})$, then $w \in L(G)$, completing the proof. ■

Proof. Suppose that $w \in L(G)$, so that

$$S \overset{*}{\Rightarrow}_G w.$$

The subscript on the derivation sign $\Rightarrow$ is used here to distinguish between derivations with different grammars. If this derivation does not involve the production (1)

$$A \to x_1 B x_2, \tag{1}$$

then obviously

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

If it does, then look at the derivation the first time (6.1) is used. The B so introduced eventually has to be replaced; we lose nothing by assuming that this is done immediately. Thus

$$S \overset{*}{\Rightarrow}_G u_1 A u_2 \Rightarrow_G u_1 x_1 B x_2 u_2 \Rightarrow_G u_1 x_1 y_j x_2 u_2.$$

But with grammar $\widehat{G}$ we can get

$$S \overset{*}{\Rightarrow}_{\widehat{G}} u_1 A u_2 \Rightarrow_{\widehat{G}} u_1 x_1 y_j x_2 u_2.$$

Thus we can reach the same sentential form with G and $\widehat{G}$. If (1) is used again later, we can repeat the argument. It follows then, by induction on the number of times the production is applied, that

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

Therefore, if $w \in L(G)$, then $w \in L(\widehat{G})$.

By similar reasoning, we can show that if $w \in L(\widehat{G})$, then $w \in L(G)$, completing the proof. ■

Proof. Suppose that $w \in L(G)$, so that

$$S \overset{*}{\Rightarrow}_G w.$$

The subscript on the derivation sign $\Rightarrow$ is used here to distinguish between derivations with different grammars. If this derivation does not involve the production (1)

$$A \to x_1 B x_2, \tag{1}$$

then obviously

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

If it does, then look at the derivation the first time (6.1) is used. The B so introduced eventually has to be replaced; we lose nothing by assuming that this is done immediately. Thus

$$S \overset{*}{\Rightarrow}_G u_1 A u_2 \Rightarrow_G u_1 x_1 B x_2 u_2 \Rightarrow_G u_1 x_1 y_j x_2 u_2.$$

But with grammar $\widehat{G}$ we can get

$$S \overset{*}{\Rightarrow}_{\widehat{G}} u_1 A u_2 \Rightarrow_{\widehat{G}} u_1 x_1 y_j x_2 u_2.$$

Thus we can reach the same sentential form with G and $\widehat{G}$. If (1) is used again later, we can repeat the argument. It follows then, by induction on the number of times the production is applied, that

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

Therefore, if $w \in L(G)$, then $w \in L(\widehat{G})$.

By similar reasoning, we can show that if $w \in L(\widehat{G})$, then $w \in L(G)$, completing the proof. ■

Proof. Suppose that $w \in L(G)$, so that

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

The subscript on the derivation sign $\Rightarrow$ is used here to distinguish between derivations with different grammars. If this derivation does not involve the production (1)

$$A \to x_1 B x_2, \tag{1}$$

then obviously

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

If it does, then look at the derivation the first time (6.1) is used. The B so introduced eventually has to be replaced; we lose nothing by assuming that this is done immediately. Thus

$$S \stackrel{*}{\Rightarrow}_G u_1 A u_2 \Rightarrow_G u_1 x_1 B x_2 u_2 \Rightarrow_G u_1 x_1 y_j x_2 u_2.$$

But with grammar $\widehat{G}$ we can get

$$S \stackrel{*}{\Rightarrow}_{\widehat{G}} u_1 A u_2 \Rightarrow_{\widehat{G}} u_1 x_1 y_j x_2 u_2.$$

Thus we can reach the same sentential form with G and $\widehat{G}$. If (1) is used again later, we can repeat the argument. It follows then, by induction on the number of times the production is applied, that

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

Therefore, if $w \in L(G)$, then $w \in L(\widehat{G})$.

By similar reasoning, we can show that if $w \in L(\widehat{G})$, then $w \in L(G)$, completing the proof. ∎

Proof. Suppose that $w \in L(G)$, so that

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

The subscript on the derivation sign $\Rightarrow$ is used here to distinguish between derivations with different grammars. If this derivation does not involve the production (1)

$$A \to x_1 B x_2, \tag{1}$$

then obviously

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

If it does, then look at the derivation the first time (6.1) is used. The B so introduced eventually has to be replaced; we lose nothing by assuming that this is done immediately. Thus

$$S \stackrel{*}{\Rightarrow}_G u_1 A u_2 \Rightarrow_G u_1 x_1 B x_2 u_2 \Rightarrow_G u_1 x_1 y_j x_2 u_2.$$

But with grammar $\widehat{G}$ we can get

$$S \stackrel{*}{\Rightarrow}_{\widehat{G}} u_1 A u_2 \Rightarrow_{\widehat{G}} u_1 x_1 y_j x_2 u_2.$$

Thus we can reach the same sentential form with G and $\widehat{G}$. If (1) is used again later, we can repeat the argument. It follows then, by induction on the number of times the production is applied, that

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

Therefore, if $w \in L(G)$, then $w \in L(\widehat{G})$.

By similar reasoning, we can show that if $w \in L(\widehat{G})$, then $w \in L(G)$, completing the proof. ■

Proof. Suppose that $w \in L(G)$, so that

$$S \overset{*}{\Rightarrow}_G w.$$

The subscript on the derivation sign $\Rightarrow$ is used here to distinguish between derivations with different grammars. If this derivation does not involve the production (1)

$$A \to x_1 B x_2, \qquad (1)$$

then obviously

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

If it does, then look at the derivation the first time (6.1) is used. The B so introduced eventually has to be replaced; we lose nothing by assuming that this is done immediately. Thus

$$S \overset{*}{\Rightarrow}_G u_1 A u_2 \Rightarrow_G u_1 x_1 B x_2 u_2 \Rightarrow_G u_1 x_1 y_j x_2 u_2.$$

But with grammar $\widehat{G}$ we can get

$$S \overset{*}{\Rightarrow}_{\widehat{G}} u_1 A u_2 \Rightarrow_{\widehat{G}} u_1 x_1 y_j x_2 u_2.$$

Thus we can reach the same sentential form with G and $\widehat{G}$. If (1) is used again later, we can repeat the argument. It follows then, by induction on the number of times the production is applied, that

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

Therefore, if $w \in L(G)$, then $w \in L(\widehat{G})$.

By similar reasoning, we can show that if $w \in L(\widehat{G})$, then $w \in L(G)$, completing the proof. ∎

Proof. Suppose that $w \in L(G)$, so that

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

The subscript on the derivation sign $\Rightarrow$ is used here to distinguish between derivations with different grammars. If this derivation does not involve the production (1)

$$A \to x_1 B x_2, \tag{1}$$

then obviously

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

If it does, then look at the derivation the first time (6.1) is used. The B so introduced eventually has to be replaced; we lose nothing by assuming that this is done immediately. Thus

$$S \stackrel{*}{\Rightarrow}_G u_1 A u_2 \Rightarrow_G u_1 x_1 B x_2 u_2 \Rightarrow_G u_1 x_1 y_j x_2 u_2.$$

But with grammar $\widehat{G}$ we can get

$$S \stackrel{*}{\Rightarrow}_{\widehat{G}} u_1 A u_2 \Rightarrow_{\widehat{G}} u_1 x_1 y_j x_2 u_2.$$

Thus we can reach the same sentential form with G and $\widehat{G}$. If (1) is used again later, we can repeat the argument. It follows then, by induction on the number of times the production is applied, that

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

Therefore, if $w \in L(G)$, then $w \in L(\widehat{G})$.

By similar reasoning, we can show that if $w \in L(\widehat{G})$, then $w \in L(G)$, completing the proof. ■

Proof. Suppose that $w \in L(G)$, so that

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

The subscript on the derivation sign $\Rightarrow$ is used here to distinguish between derivations with different grammars. If this derivation does not involve the production (1)

$$A \to x_1 B x_2, \tag{1}$$

then obviously

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

If it does, then look at the derivation the first time (6.1) is used. The B so introduced eventually has to be replaced; we lose nothing by assuming that this is done immediately. Thus

$$S \stackrel{*}{\Rightarrow}_G u_1 A u_2 \Rightarrow_G u_1 x_1 B x_2 u_2 \Rightarrow_G u_1 x_1 y_j x_2 u_2.$$

But with grammar $\widehat{G}$ we can get

$$S \stackrel{*}{\Rightarrow}_{\widehat{G}} u_1 A u_2 \Rightarrow_{\widehat{G}} u_1 x_1 y_j x_2 u_2.$$

Thus we can reach the same sentential form with G and $\widehat{G}$. If (1) is used again later, we can repeat the argument. It follows then, by induction on the number of times the production is applied, that

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

Therefore, if $w \in L(G)$, then $w \in L(\widehat{G})$.

By similar reasoning, we can show that if $w \in L(\widehat{G})$, then $w \in L(G)$, completing the proof. ■

Proof. Suppose that $w \in L(G)$, so that
$$S \overset{*}{\Rightarrow}_G w.$$
The subscript on the derivation sign $\Rightarrow$ is used here to distinguish between derivations with different grammars. If this derivation does not involve the production (1)
$$A \to x_1 B x_2, \tag{1}$$
then obviously
$$S \overset{*}{\Rightarrow}_{\widehat{G}} w.$$
If it does, then look at the derivation the first time (6.1) is used. The B so introduced eventually has to be replaced; we lose nothing by assuming that this is done immediately. Thus
$$S \overset{*}{\Rightarrow}_G u_1 A u_2 \Rightarrow_G u_1 x_1 B x_2 u_2 \Rightarrow_G u_1 x_1 y_j x_2 u_2.$$
But with grammar $\widehat{G}$ we can get
$$S \overset{*}{\Rightarrow}_{\widehat{G}} u_1 A u_2 \Rightarrow_{\widehat{G}} u_1 x_1 y_j x_2 u_2.$$
Thus we can reach the same sentential form with G and $\widehat{G}$. If (1) is used again later, we can repeat the argument. It follows then, by induction on the number of times the production is applied, that
$$S \overset{*}{\Rightarrow}_{\widehat{G}} w.$$
Therefore, if $w \in L(G)$, then $w \in L(\widehat{G})$.

By similar reasoning, we can show that if $w \in L(\widehat{G})$, then $w \in L(G)$, completing the proof. ■

Proof. Suppose that $w \in L(G)$, so that

$$S \overset{*}{\Rightarrow}_G w.$$

The subscript on the derivation sign $\Rightarrow$ is used here to distinguish between derivations with different grammars. If this derivation does not involve the production (1)

$$A \to x_1 B x_2, \tag{1}$$

then obviously

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

If it does, then look at the derivation the first time (6.1) is used. The B so introduced eventually has to be replaced; we lose nothing by assuming that this is done immediately. Thus

$$S \overset{*}{\Rightarrow}_G u_1 A u_2 \Rightarrow_G u_1 x_1 B x_2 u_2 \Rightarrow_G u_1 x_1 y_j x_2 u_2.$$

But with grammar $\widehat{G}$ we can get

$$S \overset{*}{\Rightarrow}_{\widehat{G}} u_1 A u_2 \Rightarrow_{\widehat{G}} u_1 x_1 y_j x_2 u_2.$$

Thus we can reach the same sentential form with G and $\widehat{G}$. If (1) is used again later, we can repeat the argument. It follows then, by induction on the number of times the production is applied, that

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

Therefore, if $w \in L(G)$, then $w \in L(\widehat{G})$.

By similar reasoning, we can show that if $w \in L(\widehat{G})$, then $w \in L(G)$, completing the proof. ■

Proof. Suppose that $w \in L(G)$, so that

$$S \overset{*}{\Rightarrow}_G w.$$

The subscript on the derivation sign $\Rightarrow$ is used here to distinguish between derivations with different grammars. If this derivation does not involve the production (1)

$$A \to x_1 B x_2, \tag{1}$$

then obviously

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

If it does, then look at the derivation the first time (6.1) is used. The B so introduced eventually has to be replaced; we lose nothing by assuming that this is done immediately. Thus

$$S \overset{*}{\Rightarrow}_G u_1 A u_2 \Rightarrow_G u_1 x_1 B x_2 u_2 \Rightarrow_G u_1 x_1 y_j x_2 u_2.$$

But with grammar $\widehat{G}$ we can get

$$S \overset{*}{\Rightarrow}_{\widehat{G}} u_1 A u_2 \Rightarrow_{\widehat{G}} u_1 x_1 y_j x_2 u_2.$$

Thus we can reach the same sentential form with G and $\widehat{G}$. If (1) is used again later, we can repeat the argument. It follows then, by induction on the number of times the production is applied, that

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

Therefore, if $w \in L(G)$, then $w \in L(\widehat{G})$.

By similar reasoning, we can show that if $w \in L(\widehat{G})$, then $w \in L(G)$, completing the proof. ■

Theorem 6.1 is a simple and quite intuitive substitution rule: A production $A \to x_1 B x_2$ can be eliminated from a grammar if we put in its place the set of productions in which B is replaced by all strings it derives in one step. In this result, it is necessary that A and B be different variables.

Example 6.1

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

$$A \to a|aaA|abBc,$$

$$B \to abbA|b.$$

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

$$A \to a|aaA|ababbAc|abbc,$$

$$B \to abbA|b.$$

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 shall next show how such unnecessary productions can be removed from a grammar.

Theorem 6.1 is a simple and quite intuitive substitution rule: A production $A \to x_1 B x_2$ can be eliminated from a grammar if we put in its place the set of productions in which B is replaced by all strings it derives in one step. In this result, it is necessary that A and B be different variables.

Example 6.1

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

$$A \to a|aaA|abBc,$$

$$B \to abbA|b.$$

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

$$A \to a|aaA|ababbAc|abbc,$$

$$B \to abbA|b.$$

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 shall next show how such unnecessary productions can be removed from a grammar.

6.1 Methods for Transforming Grammars

Theorem 6.1 is a simple and quite intuitive substitution rule: A production $A \to x_1 B x_2$ can be eliminated from a grammar if we put in its place the set of productions in which B is replaced by all strings it derives in one step. In this result, it is necessary that A and B be different variables.

Example 6.1

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

$$A \to a|aaA|abBc,$$

$$B \to abbA|b.$$

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

$$A \to a|aaA|ababbAc|abbc,$$

$$B \to abbA|b.$$

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 shall next show how such unnecessary productions can be removed from a grammar.

6.1 Methods for Transforming Grammars

Theorem 6.1 is a simple and quite intuitive substitution rule: A production $A \to x_1 B x_2$ can be eliminated from a grammar if we put in its place the set of productions in which B is replaced by all strings it derives in one step. In this result, it is necessary that A and B be different variables.

Example 6.1

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

$$A \to a|aaA|abBc,$$

$$B \to abbA|b.$$

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

$$A \to a|aaA|ababbAc|abbc,$$

$$B \to abbA|b.$$

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 shall next show how such unnecessary productions can be removed from a grammar.

6.1 Methods for Transforming Grammars

Theorem 6.1 is a simple and quite intuitive substitution rule: A production $A \to x_1 B x_2$ can be eliminated from a grammar if we put in its place the set of productions in which B is replaced by all strings it derives in one step. In this result, it is necessary that A and B be different variables.

Example 6.1

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

$$A \to a|aaA|abBc,$$

$$B \to abbA|b.$$

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

$$A \to a|aaA|ababbAc|abbc,$$

$$B \to abbA|b.$$

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 shall next show how such unnecessary productions can be removed from a grammar.

Theorem 6.1 is a simple and quite intuitive substitution rule: A production $A \to x_1 B x_2$ can be eliminated from a grammar if we put in its place the set of productions in which B is replaced by all strings it derives in one step. In this result, it is necessary that A and B be different variables.

Example 6.1

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

$$A \to a|aaA|abBc,$$

$$B \to abbA|b.$$

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

$$A \to a|aaA|ababbAc|abbc,$$

$$B \to abbA|b.$$

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 shall next show how such unnecessary productions can be removed from a grammar.

Theorem 6.1 is a simple and quite intuitive substitution rule: A production $A \to x_1 B x_2$ can be eliminated from a grammar if we put in its place the set of productions in which B is replaced by all strings it derives in one step. In this result, it is necessary that A and B be different variables.

Example 6.1

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

$$A \to a|aaA|abBc,$$

$$B \to abbA|b.$$

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

$$A \to a|aaA|ababbAc|abbc,$$

$$B \to abbA|b.$$

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 shall next show how such unnecessary productions can be removed from a grammar.

Theorem 6.1 is a simple and quite intuitive substitution rule: A production $A \to x_1 B x_2$ can be eliminated from a grammar if we put in its place the set of productions in which B is replaced by all strings it derives in one step. In this result, it is necessary that A and B be different variables.

Example 6.1

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

$$A \to a|aaA|abBc,$$

$$B \to abbA|b.$$

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

$$A \to a|aaA|ababbAc|abbc,$$

$$B \to abbA|b.$$

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 shall next show how such unnecessary productions can be removed from a grammar.

Theorem 6.1 is a simple and quite intuitive substitution rule: A production $A \to x_1 B x_2$ can be eliminated from a grammar if we put in its place the set of productions in which B is replaced by all strings it derives in one step. In this result, it is necessary that A and B be different variables.

Example 6.1

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

$$A \to a|aaA|abBc,$$

$$B \to abbA|b.$$

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

$$A \to a|aaA|ababbAc|abbc,$$

$$B \to abbA|b.$$

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 shall next show how such unnecessary productions can be removed from a grammar.

Theorem 6.1 is a simple and quite intuitive substitution rule: A production $A \to x_1 B x_2$ can be eliminated from a grammar if we put in its place the set of productions in which B is replaced by all strings it derives in one step. In this result, it is necessary that A and B be different variables.

Example 6.1

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

$$A \to a|aaA|abBc,$$

$$B \to abbA|b.$$

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

$$A \to a|aaA|ababbAc|abbc,$$

$$B \to abbA|b.$$

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 shall next show how such unnecessary productions can be removed from a grammar.

Theorem 6.1 is a simple and quite intuitive substitution rule: A production $A \to x_1 B x_2$ can be eliminated from a grammar if we put in its place the set of productions in which B is replaced by all strings it derives in one step. In this result, it is necessary that A and B be different variables.

Example 6.1

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

$$A \to a|aaA|abBc,$$

$$B \to abbA|b.$$

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

$$A \to a|aaA|ababbAc|abbc,$$

$$B \to abbA|b.$$

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 shall next show how such unnecessary productions can be removed from a grammar.

Theorem 6.1 is a simple and quite intuitive substitution rule: A production $A \to x_1 B x_2$ can be eliminated from a grammar if we put in its place the set of productions in which B is replaced by all strings it derives in one step. In this result, it is necessary that A and B be different variables.

Example 6.1

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

$$A \to a|aaA|abBc,$$
$$B \to abbA|b.$$

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

$$A \to a|aaA|ababbAc|abbc,$$
$$B \to abbA|b.$$

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 shall next show how such unnecessary productions can be removed from a grammar.

Theorem 6.1 is a simple and quite intuitive substitution rule: A production $A \to x_1 B x_2$ can be eliminated from a grammar if we put in its place the set of productions in which B is replaced by all strings it derives in one step. In this result, it is necessary that A and B be different variables.

Example 6.1

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

$$A \to a|aaA|abBc,$$
$$B \to abbA|b.$$

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

$$A \to a|aaA|ababbAc|abbc,$$
$$B \to abbA|b.$$

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 shall next show how such unnecessary productions can be removed from a grammar.

Theorem 6.1 is a simple and quite intuitive substitution rule: A production $A \to x_1 B x_2$ can be eliminated from a grammar if we put in its place the set of productions in which B is replaced by all strings it derives in one step. In this result, it is necessary that A and B be different variables.

Example 6.1

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

$$A \to a|aaA|abBc,$$

$$B \to abbA|b.$$

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

$$A \to a|aaA|ababbAc|abbc,$$

$$B \to abbA|b.$$

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 shall next show how such unnecessary productions can be removed from a grammar.

Theorem 6.1 is a simple and quite intuitive substitution rule: A production $A \to x_1 B x_2$ can be eliminated from a grammar if we put in its place the set of productions in which B is replaced by all strings it derives in one step. In this result, it is necessary that A and B be different variables.

Example 6.1

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

$$A \to a|aaA|abBc,$$

$$B \to abbA|b.$$

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

$$A \to a|aaA|ababbAc|abbc,$$

$$B \to abbA|b.$$

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 shall next show how such unnecessary productions can be removed from a grammar.

Theorem 6.1 is a simple and quite intuitive substitution rule: A production $A \to x_1 B x_2$ can be eliminated from a grammar if we put in its place the set of productions in which B is replaced by all strings it derives in one step. In this result, it is necessary that A and B be different variables.

Example 6.1

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

$$A \to a|aaA|abBc,$$

$$B \to abbA|b.$$

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

$$A \to a|aaA|ababbAc|abbc,$$

$$B \to abbA|b.$$

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 shall next show how such unnecessary productions can be removed from a grammar.

Theorem 6.1 is a simple and quite intuitive substitution rule: A production $A \to x_1 B x_2$ can be eliminated from a grammar if we put in its place the set of productions in which B is replaced by all strings it derives in one step. In this result, it is necessary that A and B be different variables.

Example 6.1

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

$$A \to a|aaA|abBc,$$

$$B \to abbA|b.$$

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

$$A \to a|aaA|ababbAc|abbc,$$

$$B \to abbA|b.$$

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 shall next show how such unnecessary productions can be removed from a grammar.

Theorem 6.1 is a simple and quite intuitive substitution rule: A production $A \to x_1 B x_2$ can be eliminated from a grammar if we put in its place the set of productions in which B is replaced by all strings it derives in one step. In this result, it is necessary that A and B be different variables.

Example 6.1

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

$$A \to a|aaA|abBc,$$

$$B \to abbA|b.$$

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

$$A \to a|aaA|ababbAc|abbc,$$

$$B \to abbA|b.$$

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 shall next show how such unnecessary productions can be removed from a grammar.

Theorem 6.1 is a simple and quite intuitive substitution rule: A production $A \to x_1 B x_2$ can be eliminated from a grammar if we put in its place the set of productions in which B is replaced by all strings it derives in one step. In this result, it is necessary that A and B be different variables.

Example 6.1

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

$$A \to a|aaA|abBc,$$

$$B \to abbA|b.$$

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

$$A \to a|aaA|ababbAc|abbc,$$

$$B \to abbA|b.$$

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 shall next show how such unnecessary productions can be removed from a grammar.

Removing Useless Productions

One invariably wants to remove productions from a grammar that can never take part in any derivation. For example, in the grammar whose entire production set is

$$S \to aSb|\lambda|A,$$

$$A \to aA,$$

the production $S \to A$ clearly plays no role, as A cannot be transformed into a terminal string. While A can occur in a string derived from S, this can never lead to a sentence. Removing this production leaves the language unaffected and is a simplification by any definition.

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

$$S \overset{*}{\Rightarrow} xAy \overset{*}{\Rightarrow} w. \qquad (2)$$

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

One invariably wants to remove productions from a grammar that can never take part in any derivation. For example, in the grammar whose entire production set is

$$S \to aSb|\lambda|A,$$

$$A \to aA,$$

the production $S \to A$ clearly plays no role, as A cannot be transformed into a terminal string. While A can occur in a string derived from S, this can never lead to a sentence. Removing this production leaves the language unaffected and is a simplification by any definition.

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

$$S \stackrel{*}{\Rightarrow} xAy \stackrel{*}{\Rightarrow} w. \qquad (2)$$

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

One invariably wants to remove productions from a grammar that can never take part in any derivation. For example, in the grammar whose entire production set is

$$S \to aSb|\lambda|A,$$

$$A \to aA,$$

the production $S \to A$ clearly plays no role, as A cannot be transformed into a terminal string. While A can occur in a string derived from S, this can never lead to a sentence. Removing this production leaves the language unaffected and is a simplification by any definition.

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

$$S \stackrel{*}{\Rightarrow} xAy \stackrel{*}{\Rightarrow} w. \tag{2}$$

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

One invariably wants to remove productions from a grammar that can never take part in any derivation. For example, in the grammar whose entire production set is

$$S \to aSb|\lambda|A,$$

$$A \to aA,$$

the production $S \to A$ clearly plays no role, as A cannot be transformed into a terminal string. While A can occur in a string derived from S, this can never lead to a sentence. Removing this production leaves the language unaffected and is a simplification by any definition.

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

$$S \stackrel{*}{\Rightarrow} xAy \stackrel{*}{\Rightarrow} w. \qquad (2)$$

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

One invariably wants to remove productions from a grammar that can never take part in any derivation. For example, in the grammar whose entire production set is

$$S \to aSb|\lambda|A,$$

$$A \to aA,$$

the production $S \to A$ clearly plays no role, as A cannot be transformed into a terminal string. While A can occur in a string derived from S, this can never lead to a sentence. Removing this production leaves the language unaffected and is a simplification by any definition.

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

$$S \stackrel{*}{\Rightarrow} xAy \stackrel{*}{\Rightarrow} w. \quad (2)$$

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

One invariably wants to remove productions from a grammar that can never take part in any derivation. For example, in the grammar whose entire production set is

$$S \to aSb|\lambda|A,$$

$$A \to aA,$$

the production $S \to A$ clearly plays no role, as A cannot be transformed into a terminal string. While A can occur in a string derived from S, this can never lead to a sentence. Removing this production leaves the language unaffected and is a simplification by any definition.

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

$$S \stackrel{*}{\Rightarrow} xAy \stackrel{*}{\Rightarrow} w. \quad (2)$$

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.

6.1 Methods for Transforming Grammars

Removing Useless Productions

One invariably wants to remove productions from a grammar that can never take part in any derivation. For example, in the grammar whose entire production set is

$$S \to aSb|\lambda|A,$$

$$A \to aA,$$

the production $S \to A$ clearly plays no role, as A cannot be transformed into a terminal string. While A can occur in a string derived from S, this can never lead to a sentence. Removing this production leaves the language unaffected and is a simplification by any definition.

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

$$S \stackrel{*}{\Rightarrow} xAy \stackrel{*}{\Rightarrow} w. \tag{2}$$

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

One invariably wants to remove productions from a grammar that can never take part in any derivation. For example, in the grammar whose entire production set is

$$S \to aSb|\lambda|A,$$

$$A \to aA,$$

the production $S \to A$ clearly plays no role, as A cannot be transformed into a terminal string. While A can occur in a string derived from S, this can never lead to a sentence. Removing this production leaves the language unaffected and is a simplification by any definition.

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

$$S \stackrel{*}{\Rightarrow} xAy \stackrel{*}{\Rightarrow} w. \qquad (2)$$

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

One invariably wants to remove productions from a grammar that can never take part in any derivation. For example, in the grammar whose entire production set is

$$S \to aSb|\lambda|A,$$

$$A \to aA,$$

the production $S \to A$ clearly plays no role, as A cannot be transformed into a terminal string. While A can occur in a string derived from S, this can never lead to a sentence. Removing this production leaves the language unaffected and is a simplification by any definition.

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

$$S \stackrel{*}{\Rightarrow} xAy \stackrel{*}{\Rightarrow} w. \quad (2)$$

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

One invariably wants to remove productions from a grammar that can never take part in any derivation. For example, in the grammar whose entire production set is

$$S \to aSb|\lambda|A,$$

$$A \to aA,$$

the production $S \to A$ clearly plays no role, as A cannot be transformed into a terminal string. While A can occur in a string derived from S, this can never lead to a sentence. Removing this production leaves the language unaffected and is a simplification by any definition.

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

$$S \stackrel{*}{\Rightarrow} xAy \stackrel{*}{\Rightarrow} w. \tag{2}$$

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

One invariably wants to remove productions from a grammar that can never take part in any derivation. For example, in the grammar whose entire production set is

$$S \to aSb|\lambda|A,$$

$$A \to aA,$$

the production $S \to A$ clearly plays no role, as A cannot be transformed into a terminal string. While A can occur in a string derived from S, this can never lead to a sentence. Removing this production leaves the language unaffected and is a simplification by any definition.

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

$$S \stackrel{*}{\Rightarrow} xAy \stackrel{*}{\Rightarrow} w. \tag{2}$$

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

One invariably wants to remove productions from a grammar that can never take part in any derivation. For example, in the grammar whose entire production set is

$$S \to aSb|\lambda|A,$$

$$A \to aA,$$

the production $S \to A$ clearly plays no role, as A cannot be transformed into a terminal string. While A can occur in a string derived from S, this can never lead to a sentence. Removing this production leaves the language unaffected and is a simplification by any definition.

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

$$S \stackrel{*}{\Rightarrow} xAy \stackrel{*}{\Rightarrow} w. \quad (2)$$

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

One invariably wants to remove productions from a grammar that can never take part in any derivation. For example, in the grammar whose entire production set is

$$S \to aSb|\lambda|A,$$

$$A \to aA,$$

the production $S \to A$ clearly plays no role, as A cannot be transformed into a terminal string. While A can occur in a string derived from S, this can never lead to a sentence. Removing this production leaves the language unaffected and is a simplification by any definition.

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

$$S \overset{*}{\Rightarrow} xAy \overset{*}{\Rightarrow} w. \tag{2}$$

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

One invariably wants to remove productions from a grammar that can never take part in any derivation. For example, in the grammar whose entire production set is

$$S \to aSb|\lambda|A,$$

$$A \to aA,$$

the production $S \to A$ clearly plays no role, as A cannot be transformed into a terminal string. While A can occur in a string derived from S, this can never lead to a sentence. Removing this production leaves the language unaffected and is a simplification by any definition.

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

$$S \stackrel{*}{\Rightarrow} xAy \stackrel{*}{\Rightarrow} w. \qquad (2)$$

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

One invariably wants to remove productions from a grammar that can never take part in any derivation. For example, in the grammar whose entire production set is

$$S \to aSb|\lambda|A,$$

$$A \to aA,$$

the production $S \to A$ clearly plays no role, as A cannot be transformed into a terminal string. While A can occur in a string derived from S, this can never lead to a sentence. Removing this production leaves the language unaffected and is a simplification by any definition.

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

$$S \overset{*}{\Rightarrow} xAy \overset{*}{\Rightarrow} w. \tag{2}$$

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

One invariably wants to remove productions from a grammar that can never take part in any derivation. For example, in the grammar whose entire production set is

$$S \to aSb|\lambda|A,$$

$$A \to aA,$$

the production $S \to A$ clearly plays no role, as A cannot be transformed into a terminal string. While A can occur in a string derived from S, this can never lead to a sentence. Removing this production leaves the language unaffected and is a simplification by any definition.

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

$$S \stackrel{*}{\Rightarrow} xAy \stackrel{*}{\Rightarrow} w. \quad (2)$$

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

One invariably wants to remove productions from a grammar that can never take part in any derivation. For example, in the grammar whose entire production set is

$$S \to aSb|\lambda|A,$$

$$A \to aA,$$

the production $S \to A$ clearly plays no role, as A cannot be transformed into a terminal string. While A can occur in a string derived from S, this can never lead to a sentence. Removing this production leaves the language unaffected and is a simplification by any definition.

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

$$S \stackrel{*}{\Rightarrow} xAy \stackrel{*}{\Rightarrow} w. \tag{2}$$

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

One invariably wants to remove productions from a grammar that can never take part in any derivation. For example, in the grammar whose entire production set is

$$S \to aSb|\lambda|A,$$

$$A \to aA,$$

the production $S \to A$ clearly plays no role, as A cannot be transformed into a terminal string. While A can occur in a string derived from S, this can never lead to a sentence. Removing this production leaves the language unaffected and is a simplification by any definition.

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

$$S \overset{*}{\Rightarrow} xAy \overset{*}{\Rightarrow} w. \tag{2}$$

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.

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

$$S \to A,$$

$$A \to aA|\lambda,$$

$$B \to bA,$$

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

This example illustrates the two reasons why a variable is useless: either because it cannot be reached from the start symbol or because it cannot derive a terminal word. A procedure for removing useless variables and productions is based on recognizing these two situations. Before we present the general case and the corresponding theorem, let us look at another example.

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

$$S \to A,$$
$$A \to aA|\lambda,$$
$$B \to bA,$$

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

This example illustrates the two reasons why a variable is useless: either because it cannot be reached from the start symbol or because it cannot derive a terminal word. A procedure for removing useless variables and productions is based on recognizing these two situations. Before we present the general case and the corresponding theorem, let us look at another example.

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

$$S \to A,$$

$$A \to aA|\lambda,$$

$$B \to bA,$$

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

This example illustrates the two reasons why a variable is useless: either because it cannot be reached from the start symbol or because it cannot derive a terminal word. A procedure for removing useless variables and productions is based on recognizing these two situations. Before we present the general case and the corresponding theorem, let us look at another example.

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

$$S \to A,$$
$$A \to aA|\lambda,$$
$$B \to bA,$$

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

This example illustrates the two reasons why a variable is useless: either because it cannot be reached from the start symbol or because it cannot derive a terminal word. A procedure for removing useless variables and productions is based on recognizing these two situations. Before we present the general case and the corresponding theorem, let us look at another example.

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

$$S \to A,$$

$$A \to aA|\lambda,$$

$$B \to bA,$$

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

This example illustrates the two reasons why a variable is useless: either because it cannot be reached from the start symbol or because it cannot derive a terminal word. A procedure for removing useless variables and productions is based on recognizing these two situations. Before we present the general case and the corresponding theorem, let us look at another example.

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

$$S \to A,$$
$$A \to aA|\lambda,$$
$$B \to bA,$$

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

This example illustrates the two reasons why a variable is useless: either because it cannot be reached from the start symbol or because it cannot derive a terminal word. A procedure for removing useless variables and productions is based on recognizing these two situations. Before we present the general case and the corresponding theorem, let us look at another example.

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

$$S \to A,$$
$$A \to aA|\lambda,$$
$$B \to bA,$$

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

This example illustrates the two reasons why a variable is useless: either because it cannot be reached from the start symbol or because it cannot derive a terminal word. A procedure for removing useless variables and productions is based on recognizing these two situations. Before we present the general case and the corresponding theorem, let us look at another example.

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

$$S \to A,$$
$$A \to aA|\lambda,$$
$$B \to bA,$$

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

This example illustrates the two reasons why a variable is useless: either because it cannot be reached from the start symbol or because it cannot derive a terminal word. A procedure for removing useless variables and productions is based on recognizing these two situations. Before we present the general case and the corresponding theorem, let us look at another example.

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

$$S \to A,$$
$$A \to aA|\lambda,$$
$$B \to bA,$$

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

This example illustrates the two reasons why a variable is useless: either because it cannot be reached from the start symbol or because it cannot derive a terminal word. A procedure for removing useless variables and productions is based on recognizing these two situations. Before we present the general case and the corresponding theorem, let us look at another example.

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

$$S \to A,$$
$$A \to aA|\lambda,$$
$$B \to bA,$$

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

This example illustrates the two reasons why a variable is useless: either because it cannot be reached from the start symbol or because it cannot derive a terminal word. A procedure for removing useless variables and productions is based on recognizing these two situations. Before we present the general case and the corresponding theorem, let us look at another example.

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

$$S \to A,$$
$$A \to aA|\lambda,$$
$$B \to bA,$$

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

This example illustrates the two reasons why a variable is useless: either because it cannot be reached from the start symbol or because it cannot derive a terminal word. A procedure for removing useless variables and productions is based on recognizing these two situations. Before we present the general case and the corresponding theorem, let us look at another example.

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

$$S \to A,$$
$$A \to aA|\lambda,$$
$$B \to bA,$$

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

This example illustrates the two reasons why a variable is useless: either because it cannot be reached from the start symbol or because it cannot derive a terminal word. A procedure for removing useless variables and productions is based on recognizing these two situations. Before we present the general case and the corresponding theorem, let us look at another example.

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

$$S \to A,$$
$$A \to aA|\lambda,$$
$$B \to bA,$$

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

This example illustrates the two reasons why a variable is useless: either because it cannot be reached from the start symbol or because it cannot derive a terminal word. A procedure for removing useless variables and productions is based on recognizing these two situations. Before we present the general case and the corresponding theorem, let us look at another example.

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

$$S \to aS|A|C,$$
$$A \to a,$$
$$B \to aa,$$
$$C \to aCb.$$

First, we identify the set of variables that can lead to a terminal word. Since $A \to a$ and $B \to aa$, 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

$$S \to aS|A,$$
$$A \to a,$$
$$B \to aa.$$

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

$$S \to aS|A|C,$$

$$A \to a,$$

$$B \to aa,$$

$$C \to aCb.$$

First, we identify the set of variables that can lead to a terminal word. Since $A \to a$ and $B \to aa$, 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

$$S \to aS|A,$$

$$A \to a,$$

$$B \to aa.$$

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

$$S \to aS|A|C,$$
$$A \to a,$$
$$B \to aa,$$
$$C \to aCb.$$

First, we identify the set of variables that can lead to a terminal word. Since $A \to a$ and $B \to aa$, 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

$$S \to aS|A,$$
$$A \to a,$$
$$B \to aa.$$

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

$$S \to aS|A|C,$$
$$A \to a,$$
$$B \to aa,$$
$$C \to aCb.$$

First, we identify the set of variables that can lead to a terminal word. Since $A \to a$ and $B \to aa$, 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

$$S \to aS|A,$$
$$A \to a,$$
$$B \to aa.$$

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

$$S \to aS|A|C,$$
$$A \to a,$$
$$B \to aa,$$
$$C \to aCb.$$

First, we identify the set of variables that can lead to a terminal word. Since $A \to a$ and $B \to aa$, 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

$$S \to aS|A,$$
$$A \to a,$$
$$B \to aa.$$

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

$$S \to aS|A|C,$$

$$A \to a,$$

$$B \to aa,$$

$$C \to aCb.$$

First, we identify the set of variables that can lead to a terminal word. Since $A \to a$ and $B \to aa$, 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

$$S \to aS|A,$$

$$A \to a,$$

$$B \to aa.$$

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

$$S \to aS|A|C,$$
$$A \to a,$$
$$B \to aa,$$
$$C \to aCb.$$

First, we identify the set of variables that can lead to a terminal word. Since $A \to a$ and $B \to aa$, 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

$$S \to aS|A,$$
$$A \to a,$$
$$B \to aa.$$

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

$$S \to aS|A|C,$$
$$A \to a,$$
$$B \to aa,$$
$$C \to aCb.$$

First, we identify the set of variables that can lead to a terminal word. Since $A \to a$ and $B \to aa$, 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

$$S \to aS|A,$$
$$A \to a,$$
$$B \to aa.$$

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

$$S \to aS|A|C,$$
$$A \to a,$$
$$B \to aa,$$
$$C \to aCb.$$

First, we identify the set of variables that can lead to a terminal word. Since $A \to a$ and $B \to aa$, 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

$$S \to aS|A,$$
$$A \to a,$$
$$B \to aa.$$

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

$$S \to aS|A|C,$$

$$A \to a,$$

$$B \to aa,$$

$$C \to aCb.$$

First, we identify the set of variables that can lead to a terminal word. Since $A \to a$ and $B \to aa$, 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

$$S \to aS|A,$$

$$A \to a,$$

$$B \to aa.$$

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

$$S \to aS|A|C,$$
$$A \to a,$$
$$B \to aa,$$
$$C \to aCb.$$

First, we identify the set of variables that can lead to a terminal word. Since $A \to a$ and $B \to aa$, 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

$$S \to aS|A,$$
$$A \to a,$$
$$B \to aa.$$

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

$$S \to aS|A|C,$$
$$A \to a,$$
$$B \to aa,$$
$$C \to aCb.$$

First, we identify the set of variables that can lead to a terminal word. Since $A \to a$ and $B \to aa$, 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

$$S \to aS|A,$$
$$A \to a,$$
$$B \to aa.$$

Example 6.3 (continuation)

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

$$C \to xDy.$$

A dependency graph for V_1 is shown in the Figure.

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, the Figure shows that B is useless.

Example 6.3 (continuation)

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

$$C \to xDy.$$

A dependency graph for V_1 is shown in the Figure.

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, the Figure shows that B is useless.

Example 6.3 (continuation)

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

$$C \to xDy.$$

A dependency graph for V_1 is shown in the Figure.

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, the Figure shows that B is useless.

Example 6.3 (continuation)

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

$$C \to xDy.$$

A dependency graph for V_1 is shown in the Figure.

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, the Figure shows that B is useless.

Example 6.3 (continuation)

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

$$C \to xDy.$$

A dependency graph for V_1 is shown in the Figure.

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, the Figure shows that B is useless.

Example 6.3 (continuation)

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

$$C \to xDy.$$

A dependency graph for V_1 is shown in the Figure.

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, the Figure shows that B is useless.

Example 6.3 (continuation)

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

$$C \to xDy.$$

A dependency graph for V_1 is shown in the Figure.

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, the Figure shows that B is useless.

Example 6.3 (continuation)

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

$$C \to xDy.$$

A dependency graph for V_1 is shown in the Figure.

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, the Figure shows that B is useless.

Example 6.3 (continuation)

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

$$C \to xDy.$$

A dependency graph for V_1 is shown in the Figure.

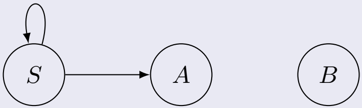
A variable is useful only if there is a path from the vertex labeled S to the vertex labeled with that variable. In our case, the Figure shows that B is useless.

Example 6.3 (continuation)

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

$$C \to xDy.$$

A dependency graph for V_1 is shown in the Figure.



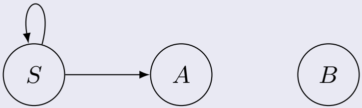
A variable is useful only if there is a path from the vertex labeled S to the vertex labeled with that variable. In our case, the Figure shows that B is useless.

Example 6.3 (continuation)

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

$$C \to xDy.$$

A dependency graph for V_1 is shown in the Figure.



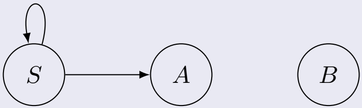
A variable is useful only if there is a path from the vertex labeled S to the vertex labeled with that variable. In our case, the Figure shows that B is useless.

Example 6.3 (continuation)

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

$$C \to xDy.$$

A dependency graph for V_1 is shown in the Figure.



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, the Figure shows that B is useless.

Example 6.3 (continuation)

Removing it and the affected productions and terminals, we are led to the final answer $\widehat{G} = (\widehat{V}, \widehat{T}, S, \widehat{P})$ with $\widehat{V} = \{S, A\}$, $\widehat{T} = \{a\}$, and productions

$$S \to aS|A,$$

$$A \to a.$$

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

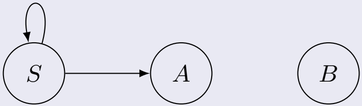
Example 6.3 (continuation)

Removing it and the affected productions and terminals, we are led to the final answer $\widehat{G} = (\widehat{V}, \widehat{T}, S, \widehat{P})$ with $\widehat{V} = \{S, A\}$, $\widehat{T} = \{a\}$, and productions

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

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

Example 6.3 (continuation)

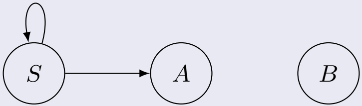


Removing it and the affected productions and terminals, we are led to the final answer $\widehat{G} = (\widehat{V}, \widehat{T}, S, \widehat{P})$ with $\widehat{V} = \{S, A\}$, $\widehat{T} = \{a\}$, and productions

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

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

Example 6.3 (continuation)

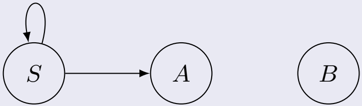


Removing it and the affected productions and terminals, we are led to the final answer $\widehat{G} = (\widehat{V}, \widehat{T}, S, \widehat{P})$ with $\widehat{V} = \{S, A\}$, $\widehat{T} = \{a\}$, and productions

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

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

Example 6.3 (continuation)

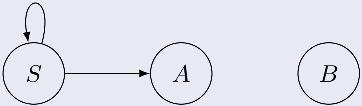


Removing it and the affected productions and terminals, we are led to the final answer $\widehat{G} = (\widehat{V}, \widehat{T}, S, \widehat{P})$ with $\widehat{V} = \{S, A\}$, $\widehat{T} = \{a\}$, and productions

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

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

Example 6.3 (continuation)



Removing it and the affected productions and terminals, we are led to the final answer $\widehat{G} = (\widehat{V}, \widehat{T}, S, \widehat{P})$ with $\widehat{V} = \{S, A\}$, $\widehat{T} = \{a\}$, and productions

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

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

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 useless 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

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

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
$$A \to x_1 x_2 \cdots x_n, \text{ with all } x_i \in (V_1 \cup T),$$
add A to V_1.
3. Take P_1 as all the productions in P whose symbols are all in $(V_1 \cup T)$.

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 useless 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

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

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
$$A \to x_1 x_2 \cdots x_n, \text{ with all } x_i \in (V_1 \cup T),$$
add A to V_1.
3. Take P_1 as all the productions in P whose symbols are all in $(V_1 \cup T)$.

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 useless 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
   $$A \to x_1 x_2 \cdots x_n, \text{ with all } x_i \in (V_1 \cup T),$$
   add A to V_1.
3. Take P_1 as all the productions in P whose symbols are all in $(V_1 \cup T)$.

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 useless 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

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

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
$$A \to x_1 x_2 \cdots x_n, \text{ with all } x_i \in V_1 \cup T,$$
add A to V_1.
3. Take P_1 as all the productions in P whose symbols are all in $(V_1 \cup T)$.

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 useless 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

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

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
$$A \to x_1 x_2 \cdots x_n, \text{ with all } x_i \in (V_1 \cup T),$$
add A to V_1.
3. Take P_1 as all the productions in P whose symbols are all in $(V_1 \cup T)$.

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 useless 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

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

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

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

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

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

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 useless 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
$$A \to x_1 x_2 \cdots x_n, \text{ with all } x_i \in V_1 \cup T,$$
add A to V_1.
3. Take P_1 as all the productions in P whose symbols are all in $(V_1 \cup T)$.

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 useless 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

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

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
$$A \to x_1 x_2 \cdots x_n, \text{ with all } x_i \in V_1 \cup T,$$
add A to V_1.
3. Take P_1 as all the productions in P whose symbols are all in $(V_1 \cup T)$.

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 useless 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

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

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 G has a production of the form
$$A \to x_1 x_2 \cdots x_n, \text{ with all } x_i \text{ in } (V_1 \cup T),$$
add A to V_1.
3. Take P_1 as all the productions in P whose symbols are all in $(V_1 \cup T)$.

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 useless 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

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

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
$$A \to x_1 x_2 \cdots x_n, \text{ with all } x_i \in V_1 \cup T,$$
add A to V_1.
3. Take P_1 as all the productions in P whose symbols are all in $(V_1 \cup T)$.

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 useless 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

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

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
$$A \to x_1 x_2 \cdots x_n, \text{ with all } x_i \in V_1 \cup T,$$
add A to V_1.
3. Take P_1 as all the productions in P whose symbols are all in $(V_1 \cup T)$.

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 useless 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
$$A \to x_1 x_2 \cdots x_n, \text{ with all } x_i \in V_1 \cup T,$$
add A to V_1.
3. Take P_1 as all the productions in P whose symbols are all in $(V_1 \cup T)$.

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 useless 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

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

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
$$A \to x_1 x_2 \cdots x_n, \text{ with all } x_i \in V_1 \cup T,$$
add A to V_1.
3. Take P_1 as all the productions in P whose symbols are all in $(V_1 \cup T)$.

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 useless 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

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

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
$$A \to x_1 x_2 \cdots x_n, \text{ with all } x_i \in V_1 \cup T,$$
add A to V_1.
3. Take P_1 as all the productions in P whose symbols are all in $(V_1 \cup T)$.

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 useless 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

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

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
$$A \to x_1 x_2 \cdots x_n, \text{ with all } x_i \in V_1 \cup T,$$
add A to V_1.
3. Take P_1 as all the productions in P whose symbols are all in $(V_1 \cup T)$.

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 useless 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

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

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
$$A \to x_1 x_2 \cdots x_n, \text{ with all } x_i \in V_1 \cup T,$$
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 $A \stackrel{*}{\Rightarrow} w \in T^*$ is a possible derivation with G_1. The remaining issue is whether every A for which $A \stackrel{*}{\Rightarrow} w = ab \cdots$ 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 (see the Figure).

Clearly this procedure terminates. It is equally clear that if $A \in V_1$, then $A \stackrel{*}{\Rightarrow} w \in T^*$ is a possible derivation with G_1. The remaining issue is whether every A for which $A \stackrel{*}{\Rightarrow} w = ab\cdots$ 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 (see the Figure).

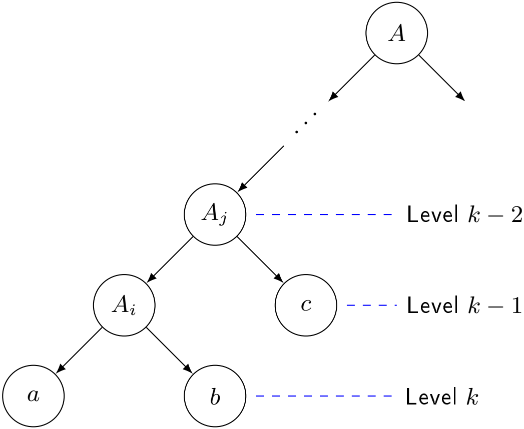
Clearly this procedure terminates. It is equally clear that if $A \in V_1$, then $A \overset{*}{\Rightarrow} w \in T^*$ is a possible derivation with G_1. The remaining issue is whether every A for which $A \overset{*}{\Rightarrow} w = ab\cdots$ is added to V_1 before the procedure terminates. To see this, consider any such A and look at the partial derivation tree corresponding to that derivation (see the Figure).

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 is 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 (see the Figure).

Clearly this procedure terminates. It is equally clear that if $A \in V_1$, then $A \stackrel{*}{\Rightarrow} w \in T^*$ is a possible derivation with G_1. The remaining issue is whether every A for which $A \stackrel{*}{\Rightarrow} w = ab \cdots$ 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 (see the Figure).

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 is 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 (see the Figure).

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 is 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 (see the Figure).



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.

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 $\widehat{G} = (\widehat{V}, \widehat{T}, S, \widehat{P})$.

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

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

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

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

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

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.

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 $\widehat{G} = (\widehat{V}, \widehat{T}, S, \widehat{P})$.

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 $\widehat{P} \subset P$. Consequently $L(\widehat{G}) \subseteq L(G)$. Putting the two results together, we see that G and $\widehat{G}$ are equivalent. ■

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.

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 $\widehat{G} = (\widehat{V}, \widehat{T}, S, \widehat{P})$.

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

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

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

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

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

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.

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 $\widehat{G} = (\widehat{V}, \widehat{T}, S, \widehat{P})$.

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 $\widehat{P} \subset P$. Consequently $L(\widehat{G}) \subseteq L(G)$. Putting the two results together, we see that G and $\widehat{G}$ are equivalent. ∎

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.

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 $\widehat{G} = (\widehat{V}, \widehat{T}, S, \widehat{P})$.

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 $\widehat{P} \subset P$. Consequently $L(\widehat{G}) \subseteq L(G)$. Putting the two results together, we see that G and $\widehat{G}$ are equivalent. ■

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.

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 $\widehat{G} = (\widehat{V}, \widehat{T}, S, \widehat{P})$.

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 $\widehat{P} \subset P$. Consequently $L(\widehat{G}) \subseteq L(G)$. Putting the two results together, we see that G and $\widehat{G}$ are equivalent. ■

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.

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 $\widehat{G} = (\widehat{V}, \widehat{T}, S, \widehat{P})$.

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 $\widehat{P} \subset P$. Consequently $L(\widehat{G}) \subseteq L(G)$. Putting the two results together, we see that G and $\widehat{G}$ are equivalent. ∎

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.

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 $\widehat{G} = (\widehat{V}, \widehat{T}, S, \widehat{P})$.

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 $\widehat{P} \subset P$. Consequently $L(\widehat{G}) \subseteq L(G)$. Putting the two results together, we see that G and $\widehat{G}$ are equivalent. ■

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.

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 $\widehat{G} = (\widehat{V}, \widehat{T}, S, \widehat{P})$.

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 $\widehat{P} \subset P$. Consequently $L(\widehat{G}) \subseteq L(G)$. Putting the two results together, we see that G and $\widehat{G}$ are equivalent. ■

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.

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 $\widehat{G} = (\widehat{V}, \widehat{T}, S, \widehat{P})$.

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 $\widehat{P} \subset P$. Consequently $L(\widehat{G}) \subseteq L(G)$. Putting the two results together, we see that G and $\widehat{G}$ are equivalent. ■

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.

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 $\widehat{G} = (\widehat{V}, \widehat{T}, S, \widehat{P})$.

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 $\widehat{P} \subset P$. Consequently $L(\widehat{G}) \subseteq L(G)$. Putting the two results together, we see that G and $\widehat{G}$ are equivalent. ■

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.

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 $\widehat{G} = (\widehat{V}, \widehat{T}, S, \widehat{P})$.

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 $\widehat{P} \subset P$. Consequently $L(\widehat{G}) \subseteq L(G)$. Putting the two results together, we see that G and $\widehat{G}$ are equivalent. ■

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.

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 $\widehat{G} = (\widehat{V}, \widehat{T}, S, \widehat{P})$.

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 $\widehat{P} \subset P$. Consequently $L(\widehat{G}) \subseteq L(G)$. Putting the two results together, we see that G and $\widehat{G}$ are equivalent. ■

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.

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 $\widehat{G} = (\widehat{V}, \widehat{T}, S, \widehat{P})$.

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 $\widehat{P} \subset P$. Consequently $L(\widehat{G}) \subseteq L(G)$. Putting the two results together, we see that G and $\widehat{G}$ are equivalent. ■

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.

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 $\widehat{G} = (\widehat{V}, \widehat{T}, S, \widehat{P})$.

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

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

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

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

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

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.

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 $\widehat{G} = (\widehat{V}, \widehat{T}, S, \widehat{P})$.

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 $\widehat{P} \subset P$. Consequently $L(\widehat{G}) \subseteq L(G)$. Putting the two results together, we see that G and $\widehat{G}$ are equivalent. ■

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.

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 $\widehat{G} = (\widehat{V}, \widehat{T}, S, \widehat{P})$.

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 $\widehat{P} \subset P$. Consequently $L(\widehat{G}) \subseteq L(G)$. Putting the two results together, we see that G and $\widehat{G}$ are equivalent. ■

6.1 Methods for Transforming Grammars

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.

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 $\widehat{G} = (\widehat{V}, \widehat{T}, S, \widehat{P})$.

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 $\widehat{P} \subset P$. Consequently $L(\widehat{G}) \subseteq L(G)$. Putting the two results together, we see that G and $\widehat{G}$ are equivalent. ■

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.

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 $\widehat{G} = (\widehat{V}, \widehat{T}, S, \widehat{P})$.

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

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

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

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

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

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.

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 $\widehat{G} = (\widehat{V}, \widehat{T}, S, \widehat{P})$.

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 $\widehat{P} \subset P$. Consequently $L(\widehat{G}) \subseteq L(G)$. Putting the two results together, we see that G and $\widehat{G}$ are equivalent. ■

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.

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 $\widehat{G} = (\widehat{V}, \widehat{T}, S, \widehat{P})$.

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 $\widehat{P} \subset P$. Consequently $L(\widehat{G}) \subseteq L(G)$. Putting the two results together, we see that G and $\widehat{G}$ are equivalent. ■

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.

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 $\widehat{G} = (\widehat{V}, \widehat{T}, S, \widehat{P})$.

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 $\widehat{P} \subset P$. Consequently $L(\widehat{G}) \subseteq L(G)$. Putting the two results together, we see that G and $\widehat{G}$ are equivalent. ∎

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.

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 $\widehat{G} = (\widehat{V}, \widehat{T}, S, \widehat{P})$.

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 $\widehat{P} \subset P$. Consequently $L(\widehat{G}) \subseteq L(G)$. Putting the two results together, we see that G and $\widehat{G}$ are equivalent. ■

Removing λ-Productions

One kind of production that is sometimes undesirable is one in which the right side is the empty string.

Definition 6.2

Any production of a context-free grammar of the form

$$A \to \lambda$$

is called a *λ-production*. Any variable A for which the derivation

$$A \stackrel{*}{\Rightarrow} \lambda \tag{3}$$

is possible is called *nullable*.

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.

Removing λ-Productions

One kind of production that is sometimes undesirable is one in which the right side is the empty string.

Definition 6.2

Any production of a context-free grammar of the form

$$A \to \lambda$$

is called a *λ-production*. Any variable A for which the derivation

$$A \stackrel{*}{\Rightarrow} \lambda \tag{3}$$

is possible is called *nullable*.

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.

Removing λ-Productions

One kind of production that is sometimes undesirable is one in which the right side is the empty string.

Definition 6.2

Any production of a context-free grammar of the form

$$A \to \lambda$$

is called a *λ-production*. Any variable A for which the derivation

$$A \stackrel{*}{\Rightarrow} \lambda \qquad (3)$$

is possible is called *nullable*.

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.

Removing λ-Productions

One kind of production that is sometimes undesirable is one in which the right side is the empty string.

Definition 6.2

Any production of a context-free grammar of the form

$$A \to \lambda$$

is called a *λ-production*. Any variable A for which the derivation

$$A \stackrel{*}{\Rightarrow} \lambda \tag{3}$$

is possible is called *nullable*.

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.

Removing λ-Productions

One kind of production that is sometimes undesirable is one in which the right side is the empty string.

Definition 6.2

Any production of a context-free grammar of the form

$$A \to \lambda$$

is called a *λ-production*. Any variable A for which the derivation

$$A \stackrel{*}{\Rightarrow} \lambda \tag{3}$$

is possible is called *nullable*.

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.

Removing λ-Productions
One kind of production that is sometimes undesirable is one in which the right side is the empty string.

Definition 6.2

Any production of a context-free grammar of the form

$$A \to \lambda$$

is called a *λ-production*. Any variable A for which the derivation

$$A \overset{*}{\Rightarrow} \lambda \tag{3}$$

is possible is called *nullable*.

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.

Removing λ-Productions
One kind of production that is sometimes undesirable is one in which the right side is the empty string.

Definition 6.2

Any production of a context-free grammar of the form

$$A \to \lambda$$

is called a *λ-production*. Any variable A for which the derivation

$$A \overset{*}{\Rightarrow} \lambda \tag{3}$$

is possible is called *nullable*.

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.

Removing λ-Productions

One kind of production that is sometimes undesirable is one in which the right side is the empty string.

Definition 6.2

Any production of a context-free grammar of the form

$$A \to \lambda$$

is called a *λ-production*. Any variable A for which the derivation

$$A \overset{*}{\Rightarrow} \lambda \tag{3}$$

is possible is called *nullable*.

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.

Removing λ-Productions

One kind of production that is sometimes undesirable is one in which the right side is the empty string.

Definition 6.2

Any production of a context-free grammar of the form

$$A \to \lambda$$

is called a *λ-production*. Any variable A for which the derivation

$$A \stackrel{*}{\Rightarrow} \lambda \quad (3)$$

is possible is called *nullable*.

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.

Removing λ-Productions

One kind of production that is sometimes undesirable is one in which the right side is the empty string.

Definition 6.2

Any production of a context-free grammar of the form

$$A \to \lambda$$

is called a *λ-production*. Any variable A for which the derivation

$$A \stackrel{*}{\Rightarrow} \lambda \tag{3}$$

is possible is called *nullable*.

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.

Removing λ-Productions

One kind of production that is sometimes undesirable is one in which the right side is the empty string.

Definition 6.2

Any production of a context-free grammar of the form

$$A \to \lambda$$

is called a *λ-production*. Any variable A for which the derivation

$$A \overset{*}{\Rightarrow} \lambda \qquad (3)$$

is possible is called *nullable*.

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.

Removing λ-Productions

One kind of production that is sometimes undesirable is one in which the right side is the empty string.

Definition 6.2

Any production of a context-free grammar of the form

$$A \to \lambda$$

is called a *λ-production*. Any variable A for which the derivation

$$A \stackrel{*}{\Rightarrow} \lambda \tag{3}$$

is possible is called *nullable*.

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

$$S \to aS_1b,$$

$$S_1 \to aS_1b|\lambda,$$

with start variable S. This grammar generates the λ-free language $\{a^nb^n : n \geqslant 1\}$. The λ-production $S_1 \to \lambda$ 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

$$S \to aS_1b|ab,$$

$$S_1 \to aS_1b|ab.$$

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.

Example 6.4

Consider the grammar

$$S \to aS_1b,$$

$$S_1 \to aS_1b|\lambda,$$

with start variable S. This grammar generates the λ-free language $\{a^nb^n : n \geqslant 1\}$. The λ-production $S_1 \to \lambda$ 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

$$S \to aS_1b|ab,$$

$$S_1 \to aS_1b|ab.$$

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.

Example 6.4

Consider the grammar

$$S \to aS_1b,$$

$$S_1 \to aS_1b|\lambda,$$

with start variable S. This grammar generates the λ-free language $\{a^nb^n : n \geqslant 1\}$. The λ-production $S_1 \to \lambda$ 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

$$S \to aS_1b|ab,$$

$$S_1 \to aS_1b|ab.$$

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.

Example 6.4

Consider the grammar

$$S \to aS_1b,$$

$$S_1 \to aS_1b|\lambda,$$

with start variable S. This grammar generates the λ-free language $\{a^nb^n : n \geqslant 1\}$. The λ-production $S_1 \to \lambda$ 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

$$S \to aS_1b|ab,$$

$$S_1 \to aS_1b|ab.$$

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.

Example 6.4

Consider the grammar

$$S \to aS_1b,$$

$$S_1 \to aS_1b|\lambda,$$

with start variable S. This grammar generates the λ-free language $\{a^nb^n : n \geqslant 1\}$. The λ-production $S_1 \to \lambda$ 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

$$S \to aS_1b|ab,$$

$$S_1 \to aS_1b|ab.$$

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.

Example 6.4

Consider the grammar

$$S \to aS_1b,$$

$$S_1 \to aS_1b|\lambda,$$

with start variable S. This grammar generates the λ-free language $\{a^nb^n : n \geqslant 1\}$. The λ-production $S_1 \to \lambda$ 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

$$S \to aS_1b|ab,$$

$$S_1 \to aS_1b|ab.$$

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.

Example 6.4

Consider the grammar

$$S \to aS_1b,$$

$$S_1 \to aS_1b|\lambda,$$

with start variable S. This grammar generates the λ-free language $\{a^nb^n : n \geqslant 1\}$. The λ-production $S_1 \to \lambda$ 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

$$S \to aS_1b|ab,$$

$$S_1 \to aS_1b|ab.$$

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.

Example 6.4

Consider the grammar

$$S \to aS_1b,$$

$$S_1 \to aS_1b|\lambda,$$

with start variable S. This grammar generates the λ-free language $\{a^n b^n : n \geqslant 1\}$. The λ-production $S_1 \to \lambda$ 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

$$S \to aS_1b|ab,$$

$$S_1 \to aS_1b|ab.$$

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.

Example 6.4

Consider the grammar

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

with start variable S. This grammar generates the λ-free language $\{a^nb^n : n \geqslant 1\}$. The λ-production $S_1 \to \lambda$ 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

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

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.

Example 6.4

Consider the grammar

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

with start variable S. This grammar generates the λ-free language $\{a^nb^n : n \geqslant 1\}$. The λ-production $S_1 \to \lambda$ 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

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

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.

Example 6.4

Consider the grammar

$$S \to aS_1b,$$

$$S_1 \to aS_1b|\lambda,$$

with start variable S. This grammar generates the λ-free language $\{a^n b^n : n \geqslant 1\}$. The λ-production $S_1 \to \lambda$ 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

$$S \to aS_1b|ab,$$

$$S_1 \to aS_1b|ab.$$

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.

Example 6.4

Consider the grammar

$$S \to aS_1b,$$

$$S_1 \to aS_1b|\lambda,$$

with start variable S. This grammar generates the λ-free language $\{a^nb^n : n \geqslant 1\}$. The λ-production $S_1 \to \lambda$ 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

$$S \to aS_1b|ab,$$

$$S_1 \to aS_1b|ab.$$

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.

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

1. For all productions $A \to \lambda$, put A into V_N.
2. Repeat the following step until no further variables are added to V_N. For all productions
$$B \to A_1 A_2 \cdots A_n,$$
where $A_1, A_2, \cdots, A_n$ 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
$$A \to x_1 x_2 \cdots x_m, \qquad m \geqslant 1,$$
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, then the production $A \to \lambda$ 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. ∎

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.

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

1. For all productions $A \to \lambda$, put A into V_N.
2. Repeat the following step until no further variables are added to V_N. For all productions
$$B \to A_1 A_2 \cdots A_n,$$
where $A_1, A_2, \cdots, A_n$ 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
$$A \to x_1 x_2 \cdots x_m, \qquad m \geqslant 1,$$
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, then the production $A \to \lambda$ 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. ∎

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.

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

1. For all productions $A \to \lambda$, put A into V_N.
2. Repeat the following step until no further variables are added to V_N. For all productions
$$B \to A_1 A_2 \cdots A_n,$$
where $A_1, A_2, \cdots, A_n$ 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
$$A \to x_1 x_2 \cdots x_m, \qquad m \geqslant 1,$$
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, then the production $A \to \lambda$ 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. ∎

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.

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

1. For all productions $A \to \lambda$, put A into V_N.
2. Repeat the following step until no further variables are added to V_N. For all productions
$$B \to A_1 A_2 \cdots A_n,$$
where $A_1, A_2, \cdots, A_n$ 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
$$A \to x_1 x_2 \cdots x_m, \qquad m \geqslant 1,$$
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, then the production $A \to \lambda$ 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. ∎

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.

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

1. For all productions $A \to \lambda$, put A into V_N.
2. Repeat the following step until no further variables are added to V_N. For all productions
$$B \to A_1 A_2 \cdots A_n,$$
where $A_1, A_2, \cdots, A_n$ 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
$$A \to x_1 x_2 \cdots x_m, \qquad m \geqslant 1,$$
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, then the production $A \to \lambda$ 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. ∎

6.1 Methods for Transforming Grammars

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.

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

1. For all productions $A \to \lambda$, put A into V_N.
2. Repeat the following step until no further variables are added to V_N. For all productions
$$B \to A_1 A_2 \cdots A_n,$$
where $A_1, A_2, \cdots, A_n$ 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
$$A \to x_1 x_2 \cdots x_m, \qquad m \geqslant 1,$$
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, then the production $A \to \lambda$ 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. ∎

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.

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

1. For all productions $A \to \lambda$, put A into V_N.
2. Repeat the following step until no further variables are added to V_N. For all productions
$$B \to A_1 A_2 \cdots A_n,$$
where $A_1, A_2, \cdots, A_n$ 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
$$A \to x_1 x_2 \cdots x_m, \qquad m \geqslant 1,$$
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, then the production $A \to \lambda$ 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. ∎

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.

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

1. For all productions $A \to \lambda$, put A into V_N.
2. Repeat the following step until no further variables are added to V_N. For all productions
$$B \to A_1 A_2 \cdots A_n,$$
where $A_1, A_2, \cdots, A_n$ 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

$$A \to x_1 x_2 \cdots x_m, \qquad m \geqslant 1,$$

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, then the production $A \to \lambda$ 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. ∎

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.

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

1. For all productions $A \to \lambda$, put A into V_N.
2. Repeat the following step until no further variables are added to V_N. For all productions
$$B \to A_1 A_2 \cdots A_n,$$
where $A_1, A_2, \cdots, A_n$ 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
$$A \to x_1 x_2 \cdots x_m, \qquad m \geqslant 1,$$
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, then the production $A \to \lambda$ 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. ∎

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.

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

1. For all productions $A \to \lambda$, put A into V_N.
2. Repeat the following step until no further variables are added to V_N. For all productions
$$B \to A_1 A_2 \cdots A_n,$$
where $A_1, A_2, \cdots, A_n$ 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

$$A \to x_1 x_2 \cdots x_m, \qquad m \geqslant 1,$$

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, then the production $A \to \lambda$ 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. ∎

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.

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

1. For all productions $A \to \lambda$, put A into V_N.
2. Repeat the following step until no further variables are added to V_N. For all productions
$$B \to A_1 A_2 \cdots A_n,$$
where $A_1, A_2, \cdots, A_n$ 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

$$A \to x_1 x_2 \cdots x_m, \qquad m \geqslant 1,$$

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, then the production $A \to \lambda$ 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. ∎

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.

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

1. For all productions $A \to \lambda$, put A into V_N.
2. Repeat the following step until no further variables are added to V_N. For all productions
$$B \to A_1 A_2 \cdots A_n,$$
where $A_1, A_2, \cdots, A_n$ 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

$$A \to x_1 x_2 \cdots x_m, \qquad m \geqslant 1,$$

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, then the production $A \to \lambda$ 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. ∎

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.

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

1. For all productions $A \to \lambda$, put A into V_N.
2. Repeat the following step until no further variables are added to V_N. For all productions
$$B \to A_1 A_2 \cdots A_n,$$
where $A_1, A_2, \cdots, A_n$ 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

$$A \to x_1 x_2 \cdots x_m, \qquad m \geqslant 1,$$

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, then the production $A \to \lambda$ 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. ■

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.

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

1. For all productions $A \to \lambda$, put A into V_N.
2. Repeat the following step until no further variables are added to V_N. For all productions
$$B \to A_1 A_2 \cdots A_n,$$
where $A_1, A_2, \cdots, A_n$ 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

$$A \to x_1 x_2 \cdots x_m, \qquad m \geqslant 1,$$

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, then the production $A \to \lambda$ 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. ∎

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.

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

1. For all productions $A \to \lambda$, put A into V_N.
2. Repeat the following step until no further variables are added to V_N. For all productions
$$B \to A_1 A_2 \cdots A_n,$$
where $A_1, A_2, \cdots, A_n$ 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

$$A \to x_1 x_2 \cdots x_m, \qquad m \geqslant 1,$$

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, then the production $A \to \lambda$ 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. ∎

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.

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

1. For all productions $A \to \lambda$, put A into V_N.
2. Repeat the following step until no further variables are added to V_N. For all productions
$$B \to A_1 A_2 \cdots A_n,$$
where $A_1, A_2, \cdots, A_n$ 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
$$A \to x_1 x_2 \cdots x_m, \qquad m \geqslant 1,$$
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, then the production $A \to \lambda$ 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. ∎

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.

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

1. For all productions $A \to \lambda$, put A into V_N.
2. Repeat the following step until no further variables are added to V_N. For all productions
$$B \to A_1 A_2 \cdots A_n,$$
where $A_1, A_2, \cdots, A_n$ 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
$$A \to x_1 x_2 \cdots x_m, \qquad m \geqslant 1,$$
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, then the production $A \to \lambda$ 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. ∎

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.

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

1. For all productions $A \to \lambda$, put A into V_N.
2. Repeat the following step until no further variables are added to V_N. For all productions
$$B \to A_1 A_2 \cdots A_n,$$
where $A_1, A_2, \cdots, A_n$ 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
$$A \to x_1 x_2 \cdots x_m, \qquad m \geqslant 1,$$
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, then the production $A \to \lambda$ 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. ∎

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.

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

1. For all productions $A \to \lambda$, put A into V_N.
2. Repeat the following step until no further variables are added to V_N. For all productions
$$B \to A_1 A_2 \cdots A_n,$$
where $A_1, A_2, \cdots, A_n$ 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
$$A \to x_1 x_2 \cdots x_m, \qquad m \geqslant 1,$$
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, then the production $A \to \lambda$ 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. ■

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.

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

1. For all productions $A \to \lambda$, put A into V_N.
2. Repeat the following step until no further variables are added to V_N. For all productions
$$B \to A_1 A_2 \cdots A_n,$$
where $A_1, A_2, \cdots, A_n$ 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
$$A \to x_1 x_2 \cdots x_m, \qquad m \geqslant 1,$$
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, then the production $A \to \lambda$ 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. ∎

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.

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

1. For all productions $A \to \lambda$, put A into V_N.
2. Repeat the following step until no further variables are added to V_N. For all productions
$$B \to A_1 A_2 \cdots A_n,$$
where $A_1, A_2, \cdots, A_n$ 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
$$A \to x_1 x_2 \cdots x_m, \qquad m \geqslant 1,$$
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, then the production $A \to \lambda$ 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. ■

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.

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

1. For all productions $A \to \lambda$, put A into V_N.
2. Repeat the following step until no further variables are added to V_N. For all productions
$$B \to A_1 A_2 \cdots A_n,$$
where $A_1, A_2, \cdots, A_n$ 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
$$A \to x_1 x_2 \cdots x_m, \qquad m \geqslant 1,$$
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, then the production $A \to \lambda$ 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. ■

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.

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

1. For all productions $A \to \lambda$, put A into V_N.
2. Repeat the following step until no further variables are added to V_N. For all productions
$$B \to A_1 A_2 \cdots A_n,$$
where $A_1, A_2, \cdots, A_n$ 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
$$A \to x_1 x_2 \cdots x_m, \qquad m \geqslant 1,$$
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, then the production $A \to \lambda$ 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. ∎

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.

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

1. For all productions $A \to \lambda$, put A into V_N.
2. Repeat the following step until no further variables are added to V_N. For all productions
$$B \to A_1 A_2 \cdots A_n,$$
where $A_1, A_2, \cdots, A_n$ 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
$$A \to x_1 x_2 \cdots x_m, \qquad m \geqslant 1,$$
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, then the production $A \to \lambda$ 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. ∎

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.

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

1. For all productions $A \to \lambda$, put A into V_N.
2. Repeat the following step until no further variables are added to V_N. For all productions
$$B \to A_1 A_2 \cdots A_n,$$
where $A_1, A_2, \cdots, A_n$ 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
$$A \to x_1 x_2 \cdots x_m, \qquad m \geqslant 1,$$
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, then the production $A \to \lambda$ 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

$$S \to ABaC,$$
$$A \to BC,$$
$$B \to b|\lambda,$$
$$C \to D|\lambda,$$
$$D \to d.$$

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

$$S \to ABaC|BaC|AaC|ABa|aC|Aa|Ba|a,$$
$$A \to BC,$$
$$B \to b|\lambda,$$
$$C \to D|\lambda,$$
$$D \to d.$$

Example 6.5

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

$$S \to ABaC,$$
$$A \to BC,$$
$$B \to b|\lambda,$$
$$C \to D|\lambda,$$
$$D \to d.$$

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

$$S \to ABaC|BaC|AaC|ABa|aC|Aa|Ba|a,$$
$$A \to BC,$$
$$B \to b|\lambda,$$
$$C \to D|\lambda,$$
$$D \to d.$$

Example 6.5

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

$$S \to ABaC,$$
$$A \to BC,$$
$$B \to b|\lambda,$$
$$C \to D|\lambda,$$
$$D \to d.$$

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

$$S \to ABaC|BaC|AaC|ABa|aC|Aa|Ba|a,$$
$$A \to BC,$$
$$B \to b|\lambda,$$
$$C \to D|\lambda,$$
$$D \to d.$$

Example 6.5

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

$$S \to ABaC,$$
$$A \to BC,$$
$$B \to b|\lambda,$$
$$C \to D|\lambda,$$
$$D \to d.$$

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

$$S \to ABaC|BaC|AaC|ABa|aC|Aa|Ba|a,$$
$$A \to BC,$$
$$B \to b|\lambda,$$
$$C \to D|\lambda,$$
$$D \to d.$$

Example 6.5

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

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

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

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

Example 6.5

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

$$S \to ABaC,$$
$$A \to BC,$$
$$B \to b|\lambda,$$
$$C \to D|\lambda,$$
$$D \to d.$$

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

$$S \to ABaC|BaC|AaC|ABa|aC|Aa|Ba|a,$$
$$A \to BC,$$
$$B \to b|\lambda,$$
$$C \to D|\lambda,$$
$$D \to d.$$

Example 6.5

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

$$S \to ABaC,$$
$$A \to BC,$$
$$B \to b|\lambda,$$
$$C \to D|\lambda,$$
$$D \to d.$$

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

$$S \to ABaC|BaC|AaC|ABa|aC|Aa|Ba|a,$$
$$A \to BC,$$
$$B \to b|\lambda,$$
$$C \to D|\lambda,$$
$$D \to d.$$

Example 6.5

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

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

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

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

Removing Unit-Productions

As we have seen in Theorem 5.2, productions in which both sides are a single variable are at times undesirable.

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*.

To remove unit-productions, we use the substitution rule discussed in Theorem 6.1. As the construction in the next theorem shows, this can be done if we proceed with some care.

Theorem 6.4

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

Removing Unit-Productions

As we have seen in Theorem 5.2, productions in which both sides are a single variable are at times undesirable.

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*.

To remove unit-productions, we use the substitution rule discussed in Theorem 6.1. As the construction in the next theorem shows, this can be done if we proceed with some care.

Theorem 6.4

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

Removing Unit-Productions

As we have seen in Theorem 5.2, productions in which both sides are a single variable are at times undesirable.

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*.

To remove unit-productions, we use the substitution rule discussed in Theorem 6.1. As the construction in the next theorem shows, this can be done if we proceed with some care.

Theorem 6.4

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

Removing Unit-Productions

As we have seen in Theorem 5.2, productions in which both sides are a single variable are at times undesirable.

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*.

To remove unit-productions, we use the substitution rule discussed in Theorem 6.1. As the construction in the next theorem shows, this can be done if we proceed with some care.

Theorem 6.4

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

Removing Unit-Productions

As we have seen in Theorem 5.2, productions in which both sides are a single variable are at times undesirable.

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*.

To remove unit-productions, we use the substitution rule discussed in Theorem 6.1. As the construction in the next theorem shows, this can be done if we proceed with some care.

Theorem 6.4

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

Removing Unit-Productions

As we have seen in Theorem 5.2, productions in which both sides are a single variable are at times undesirable.

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*.

To remove unit-productions, we use the substitution rule discussed in Theorem 6.1. As the construction in the next theorem shows, this can be done if we proceed with some care.

Theorem 6.4

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

Removing Unit-Productions

As we have seen in Theorem 5.2, productions in which both sides are a single variable are at times undesirable.

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*.

To remove unit-productions, we use the substitution rule discussed in Theorem 6.1. As the construction in the next theorem shows, this can be done if we proceed with some care.

Theorem 6.4

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

Removing Unit-Productions

As we have seen in Theorem 5.2, productions in which both sides are a single variable are at times undesirable.

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*.

To remove unit-productions, we use the substitution rule discussed in Theorem 6.1. As the construction in the next theorem shows, this can be done if we proceed with some care.

Theorem 6.4

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

Removing Unit-Productions

As we have seen in Theorem 5.2, productions in which both sides are a single variable are at times undesirable.

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*.

To remove unit-productions, we use the substitution rule discussed in Theorem 6.1. As the construction in the next theorem shows, this can be done if we proceed with some care.

Theorem 6.4

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

Removing Unit-Productions

As we have seen in Theorem 5.2, productions in which both sides are a single variable are at times undesirable.

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*.

To remove unit-productions, we use the substitution rule discussed in Theorem 6.1. As the construction in the next theorem shows, this can be done if we proceed with some care.

Theorem 6.4

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

Removing Unit-Productions

As we have seen in Theorem 5.2, productions in which both sides are a single variable are at times undesirable.

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*.

To remove unit-productions, we use the substitution rule discussed in Theorem 6.1. As the construction in the next theorem shows, this can be done if we proceed with some care.

Theorem 6.4

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

Proof. Obviously, any unit-production of the form $A \to A$ can be removed from the grammar without effect, and we need only consider $A \to B$, 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

$$A \to B$$

with

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

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

$$A \overset{*}{\Rightarrow} B. \tag{4}$$

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 (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 (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.

Proof. Obviously, any unit-production of the form $A \to A$ can be removed from the grammar without effect, and we need only consider $A \to B$, 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

$$A \to B$$

with

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

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

$$A \overset{*}{\Rightarrow} B. \tag{4}$$

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 (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 (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.

Proof. Obviously, any unit-production of the form $A \to A$ can be removed from the grammar without effect, and we need only consider $A \to B$, 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

$$A \to B$$

with

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

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

$$A \stackrel{*}{\Rightarrow} B. \tag{4}$$

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 (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 (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.

Proof. Obviously, any unit-production of the form $A \to A$ can be removed from the grammar without effect, and we need only consider $A \to B$, 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

$$A \to B$$

with

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

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

$$A \overset{*}{\Rightarrow} B. \tag{4}$$

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 (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 (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.

Proof. Obviously, any unit-production of the form $A \to A$ can be removed from the grammar without effect, and we need only consider $A \to B$, 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

$$A \to B$$

with

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

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

$$A \stackrel{*}{\Rightarrow} B. \tag{4}$$

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 (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 (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.

Proof. Obviously, any unit-production of the form $A \to A$ can be removed from the grammar without effect, and we need only consider $A \to B$, 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

$$A \to B$$

with

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

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

$$A \stackrel{*}{\Rightarrow} B. \tag{4}$$

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 (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 (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.

Proof. Obviously, any unit-production of the form $A \to A$ can be removed from the grammar without effect, and we need only consider $A \to B$, 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

$$A \to B$$

with

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

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

$$A \overset{*}{\Rightarrow} B. \tag{4}$$

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 (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 (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.

Proof. Obviously, any unit-production of the form $A \to A$ can be removed from the grammar without effect, and we need only consider $A \to B$, 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

$$A \to B$$

with

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

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

$$A \overset{*}{\Rightarrow} B. \tag{4}$$

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 (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 (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.

Proof. Obviously, any unit-production of the form $A \to A$ can be removed from the grammar without effect, and we need only consider $A \to B$, 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

$$A \to B$$

with

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

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

$$A \overset{*}{\Rightarrow} B. \tag{4}$$

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 (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 (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.

Proof. Obviously, any unit-production of the form $A \to A$ can be removed from the grammar without effect, and we need only consider $A \to B$, 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

$$A \to B$$

with

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

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

$$A \overset{*}{\Rightarrow} B. \tag{4}$$

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 (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 (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.

Proof. Obviously, any unit-production of the form $A \to A$ can be removed from the grammar without effect, and we need only consider $A \to B$, 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

$$A \to B$$

with

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

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

$$A \overset{*}{\Rightarrow} B. \tag{4}$$

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 (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 (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.

Proof. Obviously, any unit-production of the form $A \to A$ can be removed from the grammar without effect, and we need only consider $A \to B$, 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

$$A \to B$$

with

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

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

$$A \stackrel{*}{\Rightarrow} B. \tag{4}$$

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 (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 (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.

Proof. Obviously, any unit-production of the form $A \to A$ can be removed from the grammar without effect, and we need only consider $A \to B$, 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

$$A \to B$$

with

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

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

$$A \overset{*}{\Rightarrow} B. \tag{4}$$

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 (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 (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.

Proof. Obviously, any unit-production of the form $A \to A$ can be removed from the grammar without effect, and we need only consider $A \to B$, 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

$$A \to B$$

with

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

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

$$A \stackrel{*}{\Rightarrow} B. \tag{4}$$

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 (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 (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.

Proof. Obviously, any unit-production of the form $A \to A$ can be removed from the grammar without effect, and we need only consider $A \to B$, 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

$$A \to B$$

with

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

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

$$A \stackrel{*}{\Rightarrow} B. \tag{4}$$

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 (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 (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.

Proof. Obviously, any unit-production of the form $A \to A$ can be removed from the grammar without effect, and we need only consider $A \to B$, 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

$$A \to B$$

with

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

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

$$A \overset{*}{\Rightarrow} B. \tag{4}$$

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 (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 (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.

Proof. Obviously, any unit-production of the form $A \to A$ can be removed from the grammar without effect, and we need only consider $A \to B$, 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

$$A \to B$$

with

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

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

$$A \overset{*}{\Rightarrow} B. \tag{4}$$

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 (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 (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.

Proof. Obviously, any unit-production of the form $A \to A$ can be removed from the grammar without effect, and we need only consider $A \to B$, 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

$$A \to B$$

with

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

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

$$A \stackrel{*}{\Rightarrow} B. \tag{4}$$

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 (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 (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.

Proof. Obviously, any unit-production of the form $A \to A$ can be removed from the grammar without effect, and we need only consider $A \to B$, 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

$$A \to B$$

with

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

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

$$A \overset{*}{\Rightarrow} B. \qquad (4)$$

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 (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 (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.

Proof. Obviously, any unit-production of the form $A \to A$ can be removed from the grammar without effect, and we need only consider $A \to B$, 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

$$A \to B$$

with

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

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

$$A \overset{*}{\Rightarrow} B. \tag{4}$$

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 (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 (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.

Proof. Obviously, any unit-production of the form $A \to A$ can be removed from the grammar without effect, and we need only consider $A \to B$, 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

$$A \to B$$

with

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

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

$$A \overset{*}{\Rightarrow} B. \tag{4}$$

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 (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 (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.

Proof. Obviously, any unit-production of the form $A \to A$ can be removed from the grammar without effect, and we need only consider $A \to B$, 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

$$A \to B$$

with

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

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

$$A \stackrel{*}{\Rightarrow} B. \tag{4}$$

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 (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 (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.

Proof. Obviously, any unit-production of the form $A \to A$ can be removed from the grammar without effect, and we need only consider $A \to B$, 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

$$A \to B$$

with

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

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

$$A \overset{*}{\Rightarrow} B. \tag{4}$$

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 (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 (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.

Proof. Obviously, any unit-production of the form $A \to A$ can be removed from the grammar without effect, and we need only consider $A \to B$, 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

$$A \to B$$

with

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

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

$$A \stackrel{*}{\Rightarrow} B. \tag{4}$$

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 (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 (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.

6.1 Methods for Transforming Grammars

To show that the resulting grammar is equivalent to the original one, we can follow the same line of reasoning as in Theorem 6.1. ∎

Example 6.6

Remove all unit-productions from

$$S \to Aa|B,$$
$$B \to A|bb,$$
$$A \to a|bc|B.$$

The dependency graph for the unit-productions is given in the Figure.

We see from it that $S \stackrel{*}{\Rightarrow} A$, $S \stackrel{*}{\Rightarrow} B$, $B \stackrel{*}{\Rightarrow} A$, and $A \stackrel{*}{\Rightarrow} B$. Hence, we add to the original non-unit productions

$$S \to Aa,$$
$$A \to a|bc,$$
$$B \to bb,$$

To show that the resulting grammar is equivalent to the original one, we can follow the same line of reasoning as in Theorem 6.1. ∎

Example 6.6

Remove all unit-productions from

$$S \to Aa|B,$$
$$B \to A|bb,$$
$$A \to a|bc|B.$$

The dependency graph for the unit-productions is given in the Figure.

We see from it that $S \stackrel{*}{\Rightarrow} A$, $S \stackrel{*}{\Rightarrow} B$, $B \stackrel{*}{\Rightarrow} A$, and $A \stackrel{*}{\Rightarrow} B$. Hence, we add to the original non-unit productions

$$S \to Aa,$$
$$A \to a|bc,$$
$$B \to bb,$$

6.1 Methods for Transforming Grammars

To show that the resulting grammar is equivalent to the original one, we can follow the same line of reasoning as in Theorem 6.1.

Example 6.6

Remove all unit-productions from

$$S \to Aa|B,$$
$$B \to A|bb,$$
$$A \to a|bc|B.$$

The dependency graph for the unit-productions is given in the Figure.

We see from it that $S \stackrel{*}{\Rightarrow} A$, $S \stackrel{*}{\Rightarrow} B$, $B \stackrel{*}{\Rightarrow} A$, and $A \stackrel{*}{\Rightarrow} B$. Hence, we add to the original non-unit productions

$$S \to Aa,$$
$$A \to a|bc,$$
$$B \to bb,$$

6.1 Methods for Transforming Grammars

To show that the resulting grammar is equivalent to the original one, we can follow the same line of reasoning as in Theorem 6.1. ■

Example 6.6

Remove all unit-productions from

$$S \to Aa|B,$$
$$B \to A|bb,$$
$$A \to a|bc|B.$$

The dependency graph for the unit-productions is given in the Figure.

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

$$S \to Aa,$$
$$A \to a|bc,$$
$$B \to bb,$$

To show that the resulting grammar is equivalent to the original one, we can follow the same line of reasoning as in Theorem 6.1. ■

Example 6.6

Remove all unit-productions from

$$S \to Aa|B,$$

$$B \to A|bb,$$

$$A \to a|bc|B.$$

The dependency graph for the unit-productions is given in the Figure.

We see from it that $S \stackrel{*}{\Rightarrow} A$, $S \stackrel{*}{\Rightarrow} B$, $B \stackrel{*}{\Rightarrow} A$, and $A \stackrel{*}{\Rightarrow} B$. Hence, we add to the original non-unit productions

$$S \to Aa,$$

$$A \to a|bc,$$

$$B \to bb,$$

To show that the resulting grammar is equivalent to the original one, we can follow the same line of reasoning as in Theorem 6.1. ■

Example 6.6

Remove all unit-productions from

$$S \to Aa|B,$$
$$B \to A|bb,$$
$$A \to a|bc|B.$$

The dependency graph for the unit-productions is given in the Figure.

We see from it that $S \stackrel{*}{\Rightarrow} A$, $S \stackrel{*}{\Rightarrow} B$, $B \stackrel{*}{\Rightarrow} A$, and $A \stackrel{*}{\Rightarrow} B$. Hence, we add to the original non-unit productions

$$S \to Aa,$$
$$A \to a|bc,$$
$$B \to bb,$$

To show that the resulting grammar is equivalent to the original one, we can follow the same line of reasoning as in Theorem 6.1. ■

Example 6.6

Remove all unit-productions from

$$S \to Aa|B,$$
$$B \to A|bb,$$
$$A \to a|bc|B.$$

The dependency graph for the unit-productions is given in the Figure.

We see from it that $S \stackrel{*}{\Rightarrow} A$, $S \stackrel{*}{\Rightarrow} B$, $B \stackrel{*}{\Rightarrow} A$, and $A \stackrel{*}{\Rightarrow} B$. Hence, we add to the original non-unit productions

$$S \to Aa,$$
$$A \to a|bc,$$
$$B \to bb,$$

6.1 Methods for Transforming Grammars

To show that the resulting grammar is equivalent to the original one, we can follow the same line of reasoning as in Theorem 6.1. ■

Example 6.6

Remove all unit-productions from

$$S \to Aa|B,$$

$$B \to A|bb,$$

$$A \to a|bc|B.$$

The dependency graph for the unit-productions is given in the Figure.

We see from it that $S \stackrel{*}{\Rightarrow} A$, $S \stackrel{*}{\Rightarrow} B$, $B \stackrel{*}{\Rightarrow} A$, and $A \stackrel{*}{\Rightarrow} B$. Hence, we add to the original non-unit productions

$$S \to Aa,$$

$$A \to a|bc,$$

$$B \to bb,$$

To show that the resulting grammar is equivalent to the original one, we can follow the same line of reasoning as in Theorem 6.1. ■

Example 6.6

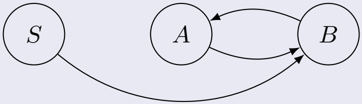
Remove all unit-productions from

$$S \to Aa|B,$$

$$B \to A|bb,$$

$$A \to a|bc|B.$$

The dependency graph for the unit-productions is given in the Figure.



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

$$S \to Aa,$$

$$A \to a|bc,$$

$$B \to bb,$$

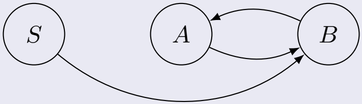
To show that the resulting grammar is equivalent to the original one, we can follow the same line of reasoning as in Theorem 6.1. ■

Example 6.6

Remove all unit-productions from

$$S \to Aa|B,$$
$$B \to A|bb,$$
$$A \to a|bc|B.$$

The dependency graph for the unit-productions is given in the Figure.



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

$$S \to Aa,$$
$$A \to a|bc,$$
$$B \to bb,$$

6.1 Methods for Transforming Grammars

To show that the resulting grammar is equivalent to the original one, we can follow the same line of reasoning as in Theorem 6.1. ■

Example 6.6

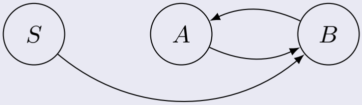
Remove all unit-productions from

$$S \to Aa|B,$$

$$B \to A|bb,$$

$$A \to a|bc|B.$$

The dependency graph for the unit-productions is given in the Figure.



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

$$S \to Aa,$$

$$A \to a|bc,$$

$$B \to bb,$$

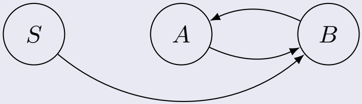
To show that the resulting grammar is equivalent to the original one, we can follow the same line of reasoning as in Theorem 6.1. ■

Example 6.6

Remove all unit-productions from

$$S \to Aa|B,$$
$$B \to A|bb,$$
$$A \to a|bc|B.$$

The dependency graph for the unit-productions is given in the Figure.



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

$$S \to Aa,$$
$$A \to a|bc,$$
$$B \to bb,$$

Example 6.6 (continuation)

the new rules

$$S \to a|bc|bb,$$

$$A \to bb,$$

$$B \to a|bc,$$

to obtain the equivalent grammar

$$S \to a|bc|bb|Aa,$$

$$A \to a|bb|bc,$$

$$B \to a|bb|bc.$$

Note that the removal of the unit-productions has made B and the associated productions useless.

We can put all these results together to show that grammars for context-free languages can be made free of useless productions, λ-productions, and unit-productions.

Example 6.6 (continuation)

the new rules

$$S \to a|bc|bb,$$

$$A \to bb,$$

$$B \to a|bc,$$

to obtain the equivalent grammar

$$S \to a|bc|bb|Aa,$$

$$A \to a|bb|bc,$$

$$B \to a|bb|bc.$$

Note that the removal of the unit-productions has made B and the associated productions useless.

We can put all these results together to show that grammars for context-free languages can be made free of useless productions, λ-productions, and unit-productions.

Example 6.6 (continuation)

the new rules

$$S \to a|bc|bb,$$

$$A \to bb,$$

$$B \to a|bc,$$

to obtain the equivalent grammar

$$S \to a|bc|bb|Aa,$$

$$A \to a|bb|bc,$$

$$B \to a|bb|bc.$$

Note that the removal of the unit-productions has made B and the associated productions useless.

We can put all these results together to show that grammars for context-free languages can be made free of useless productions, λ-productions, and unit-productions.

Example 6.6 (continuation)

the new rules

$$S \to a|bc|bb,$$

$$A \to bb,$$

$$B \to a|bc,$$

to obtain the equivalent grammar

$$S \to a|bc|bb|Aa,$$

$$A \to a|bb|bc,$$

$$B \to a|bb|bc.$$

Note that the removal of the unit-productions has made B and the associated productions useless.

We can put all these results together to show that grammars for context-free languages can be made free of useless productions, λ-productions, and unit-productions.

Example 6.6 (continuation)

the new rules

$$S \to a|bc|bb,$$

$$A \to bb,$$

$$B \to a|bc,$$

to obtain the equivalent grammar

$$S \to a|bc|bb|Aa,$$

$$A \to a|bb|bc,$$

$$B \to a|bb|bc.$$

Note that the removal of the unit-productions has made B and the associated productions useless.

We can put all these results together to show that grammars for context-free languages can be made free of useless productions, λ-productions, and unit-productions.

Example 6.6 (continuation)

the new rules

$$S \to a|bc|bb,$$

$$A \to bb,$$

$$B \to a|bc,$$

to obtain the equivalent grammar

$$S \to a|bc|bb|Aa,$$

$$A \to a|bb|bc,$$

$$B \to a|bb|bc.$$

Note that the removal of the unit-productions has made B and the associated productions useless.

We can put all these results together to show that grammars for context-free languages can be made free of useless productions, λ-productions, and unit-productions.

Example 6.6 (continuation)

the new rules

$$S \to a|bc|bb,$$
$$A \to bb,$$
$$B \to a|bc,$$

to obtain the equivalent grammar

$$S \to a|bc|bb|Aa,$$
$$A \to a|bb|bc,$$
$$B \to a|bb|bc.$$

Note that the removal of the unit-productions has made B and the associated productions useless.

We can put all these results together to show that grammars for context-free languages can be made free of useless productions, λ-productions, and unit-productions.

Example 6.6 (continuation)

the new rules

$$S \to a|bc|bb,$$

$$A \to bb,$$

$$B \to a|bc,$$

to obtain the equivalent grammar

$$S \to a|bc|bb|Aa,$$

$$A \to a|bb|bc,$$

$$B \to a|bb|bc.$$

Note that the removal of the unit-productions has made B and the associated productions useless.

We can put all these results together to show that grammars for context-free languages can be made free of useless productions, λ-productions, and unit-productions.

Theorem 6.5

Let L be a context-free language that does not contain λ. Then there exists a context-free grammar that generates L and that does not have any useless productions, λ-productions, or unit-productions.

Proof. The procedures given in Theorems 6.2, 6.3, and 6.4 remove these kinds of productions in turn. The only point that needs consideration is that the removal of one type of production may introduce productions of another type; for example, the procedure for removing λ-productions can create new unit-productions. Also, Theorem 6.4 requires that the grammar have no λ-productions. But note that the removal of unit-productions does not create λ-productions, and the removal of useless productions does not create λ-productions or unit-productions. Therefore, we can remove all undesirable productions using the following sequence of steps:

1. Remove λ-productions.
2. Remove unit-productions.
3. Remove useless productions.

The result will then have none of these productions, and the theorem is proved.

Theorem 6.5

Let L be a context-free language that does not contain λ. Then there exists a context-free grammar that generates L and that does not have any useless productions, λ-productions, or unit-productions.

Proof. The procedures given in Theorems 6.2, 6.3, and 6.4 remove these kinds of productions in turn. The only point that needs consideration is that the removal of one type of production may introduce productions of another type; for example, the procedure for removing λ-productions can create new unit-productions. Also, Theorem 6.4 requires that the grammar have no λ-productions. But note that the removal of unit-productions does not create λ-productions, and the removal of useless productions does not create λ-productions or unit-productions. Therefore, we can remove all undesirable productions using the following sequence of steps:

1. Remove λ-productions.
2. Remove unit-productions.
3. Remove useless productions.

The result will then have none of these productions, and the theorem is proved.

Theorem 6.5

Let L be a context-free language that does not contain λ. Then there exists a context-free grammar that generates L and that does not have any useless productions, λ-productions, or unit-productions.

Proof. The procedures given in Theorems 6.2, 6.3, and 6.4 remove these kinds of productions in turn. The only point that needs consideration is that the removal of one type of production may introduce productions of another type; for example, the procedure for removing λ-productions can create new unit-productions. Also, Theorem 6.4 requires that the grammar have no λ-productions. But note that the removal of unit-productions does not create λ-productions, and the removal of useless productions does not create λ-productions or unit-productions. Therefore, we can remove all undesirable productions using the following sequence of steps:

1. Remove λ-productions.
2. Remove unit-productions.
3. Remove useless productions.

The result will then have none of these productions, and the theorem is proved.

Theorem 6.5

Let L be a context-free language that does not contain λ. Then there exists a context-free grammar that generates L and that does not have any useless productions, λ-productions, or unit-productions.

Proof. The procedures given in Theorems 6.2, 6.3, and 6.4 remove these kinds of productions in turn. The only point that needs consideration is that the removal of one type of production may introduce productions of another type; for example, the procedure for removing λ-productions can create new unit-productions. Also, Theorem 6.4 requires that the grammar have no λ-productions. But note that the removal of unit-productions does not create λ-productions, and the removal of useless productions does not create λ-productions or unit-productions. Therefore, we can remove all undesirable productions using the following sequence of steps:

1. Remove λ-productions.
2. Remove unit-productions.
3. Remove useless productions.

The result will then have none of these productions, and the theorem is proved.

Theorem 6.5

Let L be a context-free language that does not contain λ. Then there exists a context-free grammar that generates L and that does not have any useless productions, λ-productions, or unit-productions.

Proof. The procedures given in Theorems 6.2, 6.3, and 6.4 remove these kinds of productions in turn. The only point that needs consideration is that the removal of one type of production may introduce productions of another type; for example, the procedure for removing λ-productions can create new unit-productions. Also, Theorem 6.4 requires that the grammar have no λ-productions. But note that the removal of unit-productions does not create λ-productions, and the removal of useless productions does not create λ-productions or unit-productions. Therefore, we can remove all undesirable productions using the following sequence of steps:

1. Remove λ-productions.
2. Remove unit-productions.
3. Remove useless productions.

The result will then have none of these productions, and the theorem is proved.

Theorem 6.5

Let L be a context-free language that does not contain λ. Then there exists a context-free grammar that generates L and that does not have any useless productions, λ-productions, or unit-productions.

Proof. The procedures given in Theorems 6.2, 6.3, and 6.4 remove these kinds of productions in turn. The only point that needs consideration is that the removal of one type of production may introduce productions of another type; for example, the procedure for removing λ-productions can create new unit-productions. Also, Theorem 6.4 requires that the grammar have no λ-productions. But note that the removal of unit-productions does not create λ-productions, and the removal of useless productions does not create λ-productions or unit-productions. Therefore, we can remove all undesirable productions using the following sequence of steps:

1. Remove λ-productions.
2. Remove unit-productions.
3. Remove useless productions.

The result will then have none of these productions, and the theorem is proved.

Theorem 6.5

Let L be a context-free language that does not contain λ. Then there exists a context-free grammar that generates L and that does not have any useless productions, λ-productions, or unit-productions.

Proof. The procedures given in Theorems 6.2, 6.3, and 6.4 remove these kinds of productions in turn. The only point that needs consideration is that the removal of one type of production may introduce productions of another type; for example, the procedure for removing λ-productions can create new unit-productions. Also, Theorem 6.4 requires that the grammar have no λ-productions. But note that the removal of unit-productions does not create λ-productions, and the removal of useless productions does not create λ-productions or unit-productions. Therefore, we can remove all undesirable productions using the following sequence of steps:

1. Remove λ-productions.
2. Remove unit-productions.
3. Remove useless productions.

The result will then have none of these productions, and the theorem is proved.

6.1 Methods for Transforming Grammars

Theorem 6.5

Let L be a context-free language that does not contain λ. Then there exists a context-free grammar that generates L and that does not have any useless productions, λ-productions, or unit-productions.

Proof. The procedures given in Theorems 6.2, 6.3, and 6.4 remove these kinds of productions in turn. The only point that needs consideration is that the removal of one type of production may introduce productions of another type; for example, the procedure for removing λ-productions can create new unit-productions. Also, Theorem 6.4 requires that the grammar have no λ-productions. But note that the removal of unit-productions does not create λ-productions, and the removal of useless productions does not create λ-productions or unit-productions. Therefore, we can remove all undesirable productions using the following sequence of steps:

1. Remove λ-productions.
2. Remove unit-productions.
3. Remove useless productions.

The result will then have none of these productions, and the theorem is proved.

Theorem 6.5

Let L be a context-free language that does not contain λ. Then there exists a context-free grammar that generates L and that does not have any useless productions, λ-productions, or unit-productions.

Proof. The procedures given in Theorems 6.2, 6.3, and 6.4 remove these kinds of productions in turn. The only point that needs consideration is that the removal of one type of production may introduce productions of another type; for example, the procedure for removing λ-productions can create new unit-productions. Also, Theorem 6.4 requires that the grammar have no λ-productions. But note that the removal of unit-productions does not create λ-productions, and the removal of useless productions does not create λ-productions or unit-productions. Therefore, we can remove all undesirable productions using the following sequence of steps:

1. Remove λ-productions.
2. Remove unit-productions.
3. Remove useless productions.

The result will then have none of these productions, and the theorem is proved.

Theorem 6.5

Let L be a context-free language that does not contain λ. Then there exists a context-free grammar that generates L and that does not have any useless productions, λ-productions, or unit-productions.

Proof. The procedures given in Theorems 6.2, 6.3, and 6.4 remove these kinds of productions in turn. The only point that needs consideration is that the removal of one type of production may introduce productions of another type; for example, the procedure for removing λ-productions can create new unit-productions. Also, Theorem 6.4 requires that the grammar have no λ-productions. But note that the removal of unit-productions does not create λ-productions, and the removal of useless productions does not create λ-productions or unit-productions. Therefore, we can remove all undesirable productions using the following sequence of steps:

1. Remove λ-productions.
2. Remove unit-productions.
3. Remove useless productions.

The result will then have none of these productions, and the theorem is proved.

Theorem 6.5

Let L be a context-free language that does not contain λ. Then there exists a context-free grammar that generates L and that does not have any useless productions, λ-productions, or unit-productions.

Proof. The procedures given in Theorems 6.2, 6.3, and 6.4 remove these kinds of productions in turn. The only point that needs consideration is that the removal of one type of production may introduce productions of another type; for example, the procedure for removing λ-productions can create new unit-productions. Also, Theorem 6.4 requires that the grammar have no λ-productions. But note that the removal of unit-productions does not create λ-productions, and the removal of useless productions does not create λ-productions or unit-productions. Therefore, we can remove all undesirable productions using the following sequence of steps:

1. Remove λ-productions.
2. Remove unit-productions.
3. Remove useless productions.

The result will then have none of these productions, and the theorem is proved.

Theorem 6.5

Let L be a context-free language that does not contain λ. Then there exists a context-free grammar that generates L and that does not have any useless productions, λ-productions, or unit-productions.

Proof. The procedures given in Theorems 6.2, 6.3, and 6.4 remove these kinds of productions in turn. The only point that needs consideration is that the removal of one type of production may introduce productions of another type; for example, the procedure for removing λ-productions can create new unit-productions. Also, Theorem 6.4 requires that the grammar have no λ-productions. But note that the removal of unit-productions does not create λ-productions, and the removal of useless productions does not create λ-productions or unit-productions. Therefore, we can remove all undesirable productions using the following sequence of steps:

1. Remove λ-productions.
2. Remove unit-productions.
3. Remove useless productions.

The result will then have none of these productions, and the theorem is proved.

Theorem 6.5

Let L be a context-free language that does not contain λ. Then there exists a context-free grammar that generates L and that does not have any useless productions, λ-productions, or unit-productions.

Proof. The procedures given in Theorems 6.2, 6.3, and 6.4 remove these kinds of productions in turn. The only point that needs consideration is that the removal of one type of production may introduce productions of another type; for example, the procedure for removing λ-productions can create new unit-productions. Also, Theorem 6.4 requires that the grammar have no λ-productions. But note that the removal of unit-productions does not create λ-productions, and the removal of useless productions does not create λ-productions or unit-productions. Therefore, we can remove all undesirable productions using the following sequence of steps:

1. Remove λ-productions.
2. Remove unit-productions.
3. Remove useless productions.

The result will then have none of these productions, and the theorem is proved.

Theorem 6.5

Let L be a context-free language that does not contain λ. Then there exists a context-free grammar that generates L and that does not have any useless productions, λ-productions, or unit-productions.

Proof. The procedures given in Theorems 6.2, 6.3, and 6.4 remove these kinds of productions in turn. The only point that needs consideration is that the removal of one type of production may introduce productions of another type; for example, the procedure for removing λ-productions can create new unit-productions. Also, Theorem 6.4 requires that the grammar have no λ-productions. But note that the removal of unit-productions does not create λ-productions, and the removal of useless productions does not create λ-productions or unit-productions. Therefore, we can remove all undesirable productions using the following sequence of steps:

1. Remove λ-productions.
2. Remove unit-productions.
3. Remove useless productions.

The result will then have none of these productions, and the theorem is proved.

Theorem 6.5

Let L be a context-free language that does not contain λ. Then there exists a context-free grammar that generates L and that does not have any useless productions, λ-productions, or unit-productions.

Proof. The procedures given in Theorems 6.2, 6.3, and 6.4 remove these kinds of productions in turn. The only point that needs consideration is that the removal of one type of production may introduce productions of another type; for example, the procedure for removing λ-productions can create new unit-productions. Also, Theorem 6.4 requires that the grammar have no λ-productions. But note that the removal of unit-productions does not create λ-productions, and the removal of useless productions does not create λ-productions or unit-productions. Therefore, we can remove all undesirable productions using the following sequence of steps:

1. Remove λ-productions.
2. Remove unit-productions.
3. Remove useless productions.

The result will then have none of these productions, and the theorem is proved.

Theorem 6.5

Let L be a context-free language that does not contain λ. Then there exists a context-free grammar that generates L and that does not have any useless productions, λ-productions, or unit-productions.

Proof. The procedures given in Theorems 6.2, 6.3, and 6.4 remove these kinds of productions in turn. The only point that needs consideration is that the removal of one type of production may introduce productions of another type; for example, the procedure for removing λ-productions can create new unit-productions. Also, Theorem 6.4 requires that the grammar have no λ-productions. But note that the removal of unit-productions does not create λ-productions, and the removal of useless productions does not create λ-productions or unit-productions. Therefore, we can remove all undesirable productions using the following sequence of steps:

1. Remove λ-productions.
2. Remove unit-productions.
3. Remove useless productions.

The result will then have none of these productions, and the theorem is proved.

Theorem 6.5

Let L be a context-free language that does not contain λ. Then there exists a context-free grammar that generates L and that does not have any useless productions, λ-productions, or unit-productions.

Proof. The procedures given in Theorems 6.2, 6.3, and 6.4 remove these kinds of productions in turn. The only point that needs consideration is that the removal of one type of production may introduce productions of another type; for example, the procedure for removing λ-productions can create new unit-productions. Also, Theorem 6.4 requires that the grammar have no λ-productions. But note that the removal of unit-productions does not create λ-productions, and the removal of useless productions does not create λ-productions or unit-productions. Therefore, we can remove all undesirable productions using the following sequence of steps:

1. Remove λ-productions.
2. Remove unit-productions.
3. Remove useless productions.

The result will then have none of these productions, and the theorem is proved.

Thank You for attention!