## Formal Languages, Automata and

 Codes
## Oleg Gutik



## Lecture 17

### 6.1 Methods for Transforming Grammars

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 $\lambda$. 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

$$
S_{0} \rightarrow S \mid \lambda
$$

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 $\lambda$ and those that do not. For the rest of this chapter, unless otherwise stated, we will restrict our discussion to $\lambda$-free languages.

### 6.1 Methods for Transforming Grammars

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 $\lambda$. 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
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 $\lambda$ and those that do not. For the rest of this chapter, unless otherwise stated, we will restrict our discussion to $\lambda$-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 $\lambda$. 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
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 $\lambda$ and those that do not. For the rest of this chapter, unless otherwise stated, we will restrict our discussion to $\lambda$-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,
give it special attention. We prefer to remove it from consideration altogether, looking only at languages that do not contain $\lambda$. 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
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 $\lambda$ and those that do not. For the rest of this chapter, unless otherwise stated, we will restrict our discussion to $\lambda$-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.
looking only at languages that do not contain $\lambda$. 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
> 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 $G$ such that $L(G)=L(G)-\{\lambda\}$. Consequently, for all practical purposes, there is no difference between context-free languages that include $\lambda$ and those that do not. For the rest of this chapter, unless otherwise stated, we will restrict our discussion to $\lambda$-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 $\lambda$. 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

almost certainly transfer to $L$. Also, given any context-free grammar $G$, there is a method for obtaining $G$ such that $L(G)=L(G)-\{\lambda\}$. Consequently, for all practical purposes, there is no difference between context-free languages that include $\lambda$ and those that do not. For the rest of this chapter, unless otherwise stated, we will restrict our discussion to $\lambda$-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 $\lambda$. 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
> 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 $\lambda$ and those that do not. For the rest of this chapter, unless otherwise stated, we will restrict our discussion to $\lambda$-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 $\lambda$. In doing so, we do not lose generality, as we see from the following considerations.


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 $\lambda$. In doing so, we do not lose generality, as we see from the following considerations. Let $L$ be any context-free language,
making $S_{0}$ the start variable, and adding to $P$ the productions
> 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 $\lambda$ and those that do not. For the rest of this chapter, unless otherwise stated, we will restrict our discussion to $\lambda$-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 $\lambda$. 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 $L-\{\lambda\}$.
making $S_{0}$ the start variable, and adding to $P$ the productions
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 $G$ such that $L(G)=L(G)-\{\lambda\}$. Consequently, for all practical purposes, there is no difference between context-free languages that include $\lambda$ and those that do not. For the rest of this chapter, unless otherwise stated, we will restrict our discussion to $\lambda$-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 $\lambda$. 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 $S_{0}$,

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 $\lambda$. 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,
> 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 $\lambda$ and those that do not. For the rest of this chapter, unless otherwise stated, we will restrict our discussion to $\lambda$-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 $\lambda$. 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
> 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 $\lambda$ and those that do not. For the rest of this chapter, unless otherwise stated, we will restrict our discussion to $\lambda$-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 $\lambda$. 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

$$
S_{0} \rightarrow S \mid \lambda
$$

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 $G$ such that $L(G)=L(G)-\{\lambda\}$. Consequently, for all practical purposes, there is no difference between context-free languages that include $\lambda$ and those that do not. For the rest of this chapter, unless otherwise stated, we will restrict our discussion to $\lambda$-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 $\lambda$. 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

$$
S_{0} \rightarrow S \mid \lambda
$$

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 $G$ such that $L(G)=L(G)-\{\lambda\}$. Consequently, for all practical purposes, there is no difference between context-free languages that include $\lambda$ and those that do not. For the rest of this chapter, unless otherwise stated, we will restrict our discussion to $\lambda$-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 $\lambda$. 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

$$
S_{0} \rightarrow S \mid \lambda
$$

generates $L$. Therefore, any nontrivial conclusion we can make for $L-\{\lambda\}$ will almost certainly transfer to $L$.
practical purposes, there is no difference between context-free languages that include $\lambda$ and those that do not. For the rest of this chapter, unless otherwise stated, we will restrict our discussion to $\lambda$-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 $\lambda$. 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

$$
S_{0} \rightarrow S \mid \lambda
$$

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

$$
S_{0} \rightarrow S \mid \lambda
$$

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 $L(\widehat{G})=L(G)-\{\lambda\}$.
include $\lambda$ and those that do not. For the rest of this chapter, unless otherwise stated, we will restrict our discussion to $\lambda$-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 $\lambda$. 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

$$
S_{0} \rightarrow S \mid \lambda
$$

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 $\lambda$ and those that do not.

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 $\lambda$. 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

$$
S_{0} \rightarrow S \mid \lambda
$$

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 $\lambda$ and those that do not. For the rest of this chapter, unless otherwise stated, we will restrict our discussion to $\lambda$-free languages.

### 6.1 Methods for Transforming Grammars

> 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

### 6.1 Methods for Transforming Grammars

## 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

### 6.1 Methods for Transforming Grammars

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.

### 6.1 Methods for Transforming Grammars

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.

### 6.1 Methods for Transforming Grammars

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,

```
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

### 6.1 Methods for Transforming Grammars

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.
productions; the process does not necessarily result in an actual reduction of the number of rules.

Theorem 6.1

### 6.1 Methods for Transforming Grammars

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 number of rules.

### 6.1 Methods for Transforming Grammars

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.

### 6.1 Methods for Transforming Grammars

## 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



### 6.1 Methods for Transforming Grammars

## 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 $A \rightarrow x_{1} B x_{2}$.
Assume that $A$ and $B$ are different variables and that

$$
B \rightarrow y_{1}\left|y_{2}\right| \cdots \mid y_{n}
$$

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 from $P$, and adding to it Then

### 6.1 Methods for Transforming Grammars

## 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


### 6.1 Methods for Transforming Grammars

## 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

$$
A \rightarrow x_{1} B x_{2} .
$$


from $P$, and adding to it

Then

### 6.1 Methods for Transforming Grammars

## 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

$$
A \rightarrow x_{1} B x_{2} .
$$

Assume that $A$ and $B$ are different variables and that
$\square$
from $P$, and adding to it

### 6.1 Methods for Transforming Grammars

## 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

$$
A \rightarrow x_{1} B x_{2} .
$$

Assume that $A$ and $B$ are different variables and that

$$
B \rightarrow y_{1}\left|y_{2}\right| \cdots \mid y_{n}
$$

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
from $P$, and adding to it

Then

### 6.1 Methods for Transforming Grammars

## 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

$$
A \rightarrow x_{1} B x_{2} .
$$

Assume that $A$ and $B$ are different variables and that

$$
B \rightarrow y_{1}\left|y_{2}\right| \cdots \mid y_{n}
$$

is the set of all productions in $P$ that have $B$ as the left side.
$G=(V, T, S, P)$ be the grammar in which $P$ is constructed by deleting
from $P$, and adding to it

Then

### 6.1 Methods for Transforming Grammars

## 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

$$
A \rightarrow x_{1} B x_{2} .
$$

Assume that $A$ and $B$ are different variables and that

$$
B \rightarrow y_{1}\left|y_{2}\right| \cdots \mid y_{n}
$$

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
from $P$, and adding to it

Then

### 6.1 Methods for Transforming Grammars

## 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

$$
A \rightarrow x_{1} B x_{2} .
$$

Assume that $A$ and $B$ are different variables and that

$$
B \rightarrow y_{1}\left|y_{2}\right| \cdots \mid y_{n}
$$

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

$$
\begin{equation*}
A \rightarrow x_{1} B x_{2} \tag{1}
\end{equation*}
$$

## from $P$, and adding to it

Then

### 6.1 Methods for Transforming Grammars

## 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

$$
A \rightarrow x_{1} B x_{2} .
$$

Assume that $A$ and $B$ are different variables and that

$$
B \rightarrow y_{1}\left|y_{2}\right| \cdots \mid y_{n}
$$

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

$$
\begin{equation*}
A \rightarrow x_{1} B x_{2} \tag{1}
\end{equation*}
$$

from $P$, and adding to it
Then

### 6.1 Methods for Transforming Grammars

## 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

$$
A \rightarrow x_{1} B x_{2} .
$$

Assume that $A$ and $B$ are different variables and that

$$
B \rightarrow y_{1}\left|y_{2}\right| \cdots \mid y_{n}
$$

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

$$
\begin{equation*}
A \rightarrow x_{1} B x_{2} \tag{1}
\end{equation*}
$$

from $P$, and adding to it

$$
A \rightarrow x_{1} y_{1} x_{2}\left|x_{1} y_{2} x_{2}\right| \cdots \mid x_{1} y_{n} x_{2}
$$

Then

### 6.1 Methods for Transforming Grammars

## 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

$$
A \rightarrow x_{1} B x_{2} .
$$

Assume that $A$ and $B$ are different variables and that

$$
B \rightarrow y_{1}\left|y_{2}\right| \cdots \mid y_{n}
$$

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

$$
\begin{equation*}
A \rightarrow x_{1} B x_{2} \tag{1}
\end{equation*}
$$

from $P$, and adding to it

$$
A \rightarrow x_{1} y_{1} x_{2}\left|x_{1} y_{2} x_{2}\right| \cdots \mid x_{1} y_{n} x_{2}
$$

Then

### 6.1 Methods for Transforming Grammars

## 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

$$
A \rightarrow x_{1} B x_{2} .
$$

Assume that $A$ and $B$ are different variables and that

$$
B \rightarrow y_{1}\left|y_{2}\right| \cdots \mid y_{n}
$$

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

$$
\begin{equation*}
A \rightarrow x_{1} B x_{2} \tag{1}
\end{equation*}
$$

from $P$, and adding to it

$$
A \rightarrow x_{1} y_{1} x_{2}\left|x_{1} y_{2} x_{2}\right| \cdots \mid x_{1} y_{n} x_{2}
$$

Then

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

### 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)

$$
\begin{equation*}
A \rightarrow x_{1} B x_{2}, \tag{1}
\end{equation*}
$$

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

$$
S \stackrel{\stackrel{1}{\Rightarrow}}{G}_{G} u_{1} A u_{2} \Rightarrow_{G} u_{1} x_{1} B x_{2} u_{2} \Rightarrow_{G} u_{1} x_{1} y_{j} x_{2} u_{2} .
$$

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

$$
S \stackrel{*}{\widehat{G}}^{{ }_{G}} u_{1} A u_{2} \Rightarrow_{\widehat{G}} u_{1} x_{1} y_{j} x_{2} u_{2} .
$$

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 $w \in L(\hat{G})$, 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)

$$
\begin{equation*}
A \rightarrow x_{1} B x_{2}, \tag{1}
\end{equation*}
$$

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

$$
S \stackrel{:}{*}_{G} u_{1} A u_{2} \Rightarrow_{G} u_{1} x_{1} B x_{2} u_{2} \Rightarrow_{G} u_{1} x_{1} y_{j} x_{2} u_{2} .
$$

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

$$
S \stackrel{*}{\widehat{G}}_{\widehat{G}} u_{1} A u_{2} \Rightarrow_{\widehat{G}} u_{1} x_{1} y_{j} x_{2} u_{2} .
$$

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

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

### 6.1 Methods for Transforming Grammars

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

$$
S \nRightarrow_{G} w .
$$

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)

$$
\begin{equation*}
A \rightarrow x_{1} B x_{2}, \tag{1}
\end{equation*}
$$

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

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

$$
\begin{equation*}
A \rightarrow x_{1} B x_{2}, \tag{1}
\end{equation*}
$$

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
$S \stackrel{*}{G}_{G} u_{1} A u_{2} \Rightarrow_{G} u_{1} x_{1} B x_{2} u_{2} \Rightarrow_{G} u_{1} x_{1} y_{j} x_{2} u_{2}$
But with grammar $\widehat{G}$ we can get

$$
S \Rightarrow{ }_{G} u_{1} A u_{2} \Rightarrow \sigma_{G} u_{1} x_{1} y_{j} x_{2} u_{2}
$$

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

Therefore, if $w \in L(G)$, then $w \in L(\widehat{G})$.
By similar reasoning, we can show that if $w \in L(G)$, 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.
production (1)

then obviously

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

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

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

Therefore, if $w \in L(G)$, then $w \in L(G)$.
By similar reasoning, we can show that if $w \in L(G)$, 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)
then obviously

$$
\begin{equation*}
A \rightarrow x_{1} B x_{2}, \tag{1}
\end{equation*}
$$

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

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

Thus we can reach the same sentential form with $G$ and $G$. 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

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

$$
\begin{equation*}
A \rightarrow x_{1} B x_{2}, \tag{1}
\end{equation*}
$$

then obviously
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

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

Thus we can reach the same sentential form with $G$ and $G$. 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

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

$$
\begin{equation*}
A \rightarrow x_{1} B x_{2}, \tag{1}
\end{equation*}
$$

then obviously
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
But with grammar $\widehat{G}$ we can get
Thus we can reach the same sentential form with $G$ and $G$. 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
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)

$$
\begin{equation*}
A \rightarrow x_{1} B x_{2}, \tag{1}
\end{equation*}
$$

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

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

Thus we can reach the same sentential form with $G$ and $G$. 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

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)
then obviously

$$
\begin{equation*}
A \rightarrow x_{1} B x_{2}, \tag{1}
\end{equation*}
$$

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

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

Thus we can reach the same sentential form with $G$ and $G$. 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

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

$$
\begin{equation*}
A \rightarrow x_{1} B x_{2}, \tag{1}
\end{equation*}
$$

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;
s done immediately. Thus

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

Thus we can reach the same sentential form with $G$ and $G$. 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

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)

$$
\begin{equation*}
A \rightarrow x_{1} B x_{2}, \tag{1}
\end{equation*}
$$

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.

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

Thus we can reach the same sentential form with $G$ and $G$. 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

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

$$
\begin{equation*}
A \rightarrow x_{1} B x_{2}, \tag{1}
\end{equation*}
$$

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

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

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

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

$$
\begin{equation*}
A \rightarrow x_{1} B x_{2}, \tag{1}
\end{equation*}
$$

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

Thus we can reach the same sentential form with $G$ and $G$. 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
$\square$
By similar reasoning, we can show that if $w \in L(G)$, 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)

$$
\begin{equation*}
A \rightarrow x_{1} B x_{2}, \tag{1}
\end{equation*}
$$

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

Thus we can reach the same sentential form with $G$ and $G$. 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
$\square$
By similar reasoning, we can show that if $w \in L(G)$, 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)

$$
\begin{equation*}
A \rightarrow x_{1} B x_{2}, \tag{1}
\end{equation*}
$$

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

$$
S \stackrel{*}{\widehat{G}}_{\widehat{G}_{1}} u_{1} u_{2} \Rightarrow_{\widehat{G}} u_{1} x_{1} y_{j} x_{2} u_{2} .
$$

Thus we can reach the same sentential
later, we can repeat the argument. It fo
of times the production is applied, that

Therefore, if $w \in L(G)$, then $w \in L(\widehat{G})$
By similar reasoning, we can show that if $w \in L(G)$, 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)

$$
\begin{equation*}
A \rightarrow x_{1} B x_{2} \tag{1}
\end{equation*}
$$

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

$$
S \stackrel{*}{\widehat{G}}_{\widehat{G}} u_{1} A u_{2} \Rightarrow_{\widehat{G}} u_{1} x_{1} y_{j} x_{2} u_{2} .
$$

Thus we can reach the same sentential form with $G$ and $\widehat{G}$.
of times the production is applied, that

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)

$$
\begin{equation*}
A \rightarrow x_{1} B x_{2} \tag{1}
\end{equation*}
$$

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

$$
S \stackrel{*}{\widehat{G}}_{\widehat{G}} u_{1} A u_{2} \Rightarrow_{\widehat{G}} u_{1} x_{1} y_{j} x_{2} u_{2} .
$$

Thus we can reach the same sentential form with $G$ and $\widehat{G}$. If (1) is used again later, we can repeat the argument. $\qquad$
$\square$
Therefore, if $w \in L(G)$, then $w \in L(G)$.
By similar reasoning, we can show that if $w \in L(G)$, 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)

$$
\begin{equation*}
A \rightarrow x_{1} B x_{2}, \tag{1}
\end{equation*}
$$

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

$$
S \stackrel{*}{\widehat{G}}_{\widehat{G}} u_{1} A u_{2} \Rightarrow_{\widehat{G}} u_{1} x_{1} y_{j} x_{2} u_{2} .
$$

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

[^0] By similar $r$ easoning, we can show that if $w \in L(G)$, 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)

$$
\begin{equation*}
A \rightarrow x_{1} B x_{2}, \tag{1}
\end{equation*}
$$

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

$$
S \stackrel{*}{\widehat{G}}_{\widehat{G}} u_{1} A u_{2} \Rightarrow_{\widehat{G}} u_{1} x_{1} y_{j} x_{2} u_{2} .
$$

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 $w \in L(G)$.
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)

$$
\begin{equation*}
A \rightarrow x_{1} B x_{2}, \tag{1}
\end{equation*}
$$

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

$$
S \stackrel{*}{G}_{\widehat{G}} u_{1} A u_{2} \Rightarrow_{\widehat{G}} u_{1} x_{1} y_{j} x_{2} u_{2} .
$$

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 $w \in L(\widehat{G})$.

### 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)

$$
\begin{equation*}
A \rightarrow x_{1} B x_{2}, \tag{1}
\end{equation*}
$$

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

$$
S \stackrel{*}{\widehat{G}}_{\widehat{G}} u_{1} A u_{2} \Rightarrow_{\widehat{G}} u_{1} x_{1} y_{j} x_{2} u_{2} .
$$

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 $w \in L(G)$,

### 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)

$$
\begin{equation*}
A \rightarrow x_{1} B x_{2}, \tag{1}
\end{equation*}
$$

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

$$
S \stackrel{*}{\widehat{G}}_{\widehat{G}} u_{1} A u_{2} \Rightarrow_{\widehat{G}} u_{1} x_{1} y_{j} x_{2} u_{2} .
$$

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)

$$
\begin{equation*}
A \rightarrow x_{1} B x_{2}, \tag{1}
\end{equation*}
$$

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

$$
S \stackrel{*}{\widehat{G}}_{\widehat{G}} u_{1} A u_{2} \Rightarrow_{\widehat{G}} u_{1} x_{1} y_{j} x_{2} u_{2} .
$$

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

```
Theorem 6.1 is a simple and quite intuitive substitution rule: A production
A-> x }B\mp@subsup{x}{2}{}\mathrm{ can be eliminated from a grammar if we put in its place the set of
productions in which B}\mathrm{ is replaced by all strings it derives in one step. In this
result, it is necessary that }A\mathrm{ and }B\mathrm{ be different variables.
```

```
Example 6.1
```


### 6.1 Methods for Transforming Grammars

Theorem 6.1 is a simple and quite intuitive substitution rule: A production
$A \rightarrow x_{1} B x_{2}$ 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.

[^1]
### 6.1 Methods for Transforming Grammars

Theorem 6.1 is a simple and quite intuitive substitution rule: A production $A \rightarrow x_{1} B x_{2}$ 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.

### 6.1 Methods for Transforming Grammars

Theorem 6.1 is a simple and quite intuitive substitution rule: A production $A \rightarrow x_{1} B x_{2}$ 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.

```
result, it is necessary that }A\mathrm{ and B be different variables.
```


### 6.1 Methods for Transforming Grammars

Theorem 6.1 is a simple and quite intuitive substitution rule: A production $A \rightarrow x_{1} B x_{2}$ 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.

### 6.1 Methods for Transforming Grammars

Theorem 6.1 is a simple and quite intuitive substitution rule: A production $A \rightarrow x_{1} B x_{2}$ 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

\[\)| $A \rightarrow a\|a a A\| a b B c,$ |
| :--- |
| $B \rightarrow a b b A \mid b .$ |

\]

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

with productions | $A \rightarrow a\|a a A\| a b a b b A c \mid a b b c$, |
| :--- |
| $B \rightarrow a b b A \mid b$. |

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

$$
A \Rightarrow a a A \Rightarrow a a a b B c \Rightarrow a a a b b c
$$

in $G$, and the corresponding derivation
in $A \Rightarrow a a A \Rightarrow$ aaabbc
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 $A \rightarrow x_{1} B x_{2}$ 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

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

The new grammar $\widehat{G}$ is equivalent to $G$. The string $a a a b b c$ has the derivation
in $G$, and the corresponding derivation

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

### 6.1 Methods for Transforming Grammars

Theorem 6.1 is a simple and quite intuitive substitution rule: A production $A \rightarrow x_{1} B x_{2}$ 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

$$
\begin{aligned}
& A \rightarrow a|a a A| a b B c \\
& B \rightarrow a b b A \mid b
\end{aligned}
$$

Using the suggested substitution for the variable $B$, we get the grammar $G$ with productions

The new grammar $\widehat{G}$ is equivalent to $G$. The string $a a a b b c$ has the derivation
in $G$, and the corresponding derivation

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

### 6.1 Methods for Transforming Grammars

Theorem 6.1 is a simple and quite intuitive substitution rule: A production $A \rightarrow x_{1} B x_{2}$ 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

$$
\begin{aligned}
& A \rightarrow a|a a A| a b B c, \\
& B \rightarrow a b b A \mid b
\end{aligned}
$$

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

The new grammar $\widehat{G}$ is equivalent to $G$. The string $a a a b b c$ has the derivation
in $G$, and the corresponding derivation

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

### 6.1 Methods for Transforming Grammars

Theorem 6.1 is a simple and quite intuitive substitution rule: A production $A \rightarrow x_{1} B x_{2}$ 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

$$
\begin{aligned}
& A \rightarrow a|a a A| a b B c, \\
& B \rightarrow a b b A \mid b
\end{aligned}
$$

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

$$
\begin{aligned}
& A \rightarrow a|a a A| a b a b b A c \mid a b b c, \\
& B \rightarrow a b b A \mid b .
\end{aligned}
$$

The new grammar $G$ is equivalent to $G$. The string aaabbc has the derivation
in $G$, and the corresponding derivation
$\qquad$

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

### 6.1 Methods for Transforming Grammars

Theorem 6.1 is a simple and quite intuitive substitution rule: A production $A \rightarrow x_{1} B x_{2}$ 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

$$
\begin{aligned}
& A \rightarrow a|a a A| a b B c, \\
& B \rightarrow a b b A \mid b
\end{aligned}
$$

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

$$
\begin{aligned}
& A \rightarrow a|a a A| a b a b b A c \mid a b b c, \\
& B \rightarrow a b b A \mid b .
\end{aligned}
$$

The new grammar $\widehat{G}$ is equivalent to $G$. The string aaabbc has the derivation
in $G$, and the corresponding derivation
$\qquad$

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

### 6.1 Methods for Transforming Grammars

Theorem 6.1 is a simple and quite intuitive substitution rule: A production $A \rightarrow x_{1} B x_{2}$ 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

$$
\begin{aligned}
& A \rightarrow a|a a A| a b B c, \\
& B \rightarrow a b b A \mid b
\end{aligned}
$$

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

$$
\begin{aligned}
& A \rightarrow a|a a A| a b a b b A c \mid a b b c, \\
& B \rightarrow a b b A \mid b .
\end{aligned}
$$

The new grammar $\widehat{G}$ is equivalent to $G$. The string $a a a b b c$ has the derivation
in $G$, and the corresponding derivation
$\qquad$

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

### 6.1 Methods for Transforming Grammars

Theorem 6.1 is a simple and quite intuitive substitution rule: A production $A \rightarrow x_{1} B x_{2}$ 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

$$
\begin{aligned}
& A \rightarrow a|a a A| a b B c, \\
& B \rightarrow a b b A \mid b
\end{aligned}
$$

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

$$
\begin{aligned}
& A \rightarrow a|a a A| a b a b b A c \mid a b b c, \\
& B \rightarrow a b b A \mid b .
\end{aligned}
$$

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

$$
A \Rightarrow a a A \Rightarrow a a a b B c \Rightarrow a a a b b c
$$

in $G$, and the corresponding derivation

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

### 6.1 Methods for Transforming Grammars

Theorem 6.1 is a simple and quite intuitive substitution rule: A production $A \rightarrow x_{1} B x_{2}$ 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

$$
\begin{aligned}
& A \rightarrow a|a a A| a b B c, \\
& B \rightarrow a b b A \mid b
\end{aligned}
$$

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

$$
\begin{aligned}
& A \rightarrow a|a a A| a b a b b A c \mid a b b c, \\
& B \rightarrow a b b A \mid b .
\end{aligned}
$$

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

$$
A \Rightarrow a a A \Rightarrow a a a b B c \Rightarrow a a a b b c
$$

in $G$, and the corresponding derivation

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

### 6.1 Methods for Transforming Grammars

Theorem 6.1 is a simple and quite intuitive substitution rule: A production $A \rightarrow x_{1} B x_{2}$ 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

$$
\begin{aligned}
& A \rightarrow a|a a A| a b B c, \\
& B \rightarrow a b b A \mid b
\end{aligned}
$$

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

$$
\begin{aligned}
& A \rightarrow a|a a A| a b a b b A c \mid a b b c, \\
& B \rightarrow a b b A \mid b .
\end{aligned}
$$

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

$$
A \Rightarrow a a A \Rightarrow a a a b B c \Rightarrow a a a b b c
$$

in $G$, and the corresponding derivation

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

### 6.1 Methods for Transforming Grammars

Theorem 6.1 is a simple and quite intuitive substitution rule: A production $A \rightarrow x_{1} B x_{2}$ 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

$$
\begin{aligned}
& A \rightarrow a|a a A| a b B c, \\
& B \rightarrow a b b A \mid b
\end{aligned}
$$

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

$$
\begin{aligned}
& A \rightarrow a|a a A| a b a b b A c \mid a b b c, \\
& B \rightarrow a b b A \mid b .
\end{aligned}
$$

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

$$
A \Rightarrow a a A \Rightarrow a a a b B c \Rightarrow a a a b b c
$$

in $G$, and the corresponding derivation

$$
A \Rightarrow a a A \Rightarrow a a a b b c
$$

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

### 6.1 Methods for Transforming Grammars

Theorem 6.1 is a simple and quite intuitive substitution rule: A production $A \rightarrow x_{1} B x_{2}$ 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

$$
\begin{aligned}
& A \rightarrow a|a a A| a b B c, \\
& B \rightarrow a b b A \mid b
\end{aligned}
$$

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

$$
\begin{aligned}
& A \rightarrow a|a a A| a b a b b A c \mid a b b c, \\
& B \rightarrow a b b A \mid b .
\end{aligned}
$$

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

$$
A \Rightarrow a a A \Rightarrow a a a b B c \Rightarrow a a a b b c
$$

in $G$, and the corresponding derivation

$$
\text { in } \widehat{G} .
$$

$$
A \Rightarrow a a A \Rightarrow a a a b b c
$$

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

### 6.1 Methods for Transforming Grammars

Theorem 6.1 is a simple and quite intuitive substitution rule: A production $A \rightarrow x_{1} B x_{2}$ 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

$$
\begin{aligned}
& A \rightarrow a|a a A| a b B c \\
& B \rightarrow a b b A \mid b
\end{aligned}
$$

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

$$
\begin{aligned}
& A \rightarrow a|a a A| a b a b b A c \mid a b b c, \\
& B \rightarrow a b b A \mid b
\end{aligned}
$$

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

$$
A \Rightarrow a a A \Rightarrow a a a b B c \Rightarrow a a a b b c
$$

in $G$, and the corresponding derivation

$$
A \Rightarrow a a A \Rightarrow a a a b b c
$$

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.

[^2]
### 6.1 Methods for Transforming Grammars

Theorem 6.1 is a simple and quite intuitive substitution rule: A production $A \rightarrow x_{1} B x_{2}$ 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

$$
\begin{aligned}
& A \rightarrow a|a a A| a b B c, \\
& B \rightarrow a b b A \mid b
\end{aligned}
$$

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

$$
\begin{aligned}
& A \rightarrow a|a a A| a b a b b A c \mid a b b c, \\
& B \rightarrow a b b A \mid b .
\end{aligned}
$$

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

$$
A \Rightarrow a a A \Rightarrow a a a b B c \Rightarrow a a a b b c
$$

in $G$, and the corresponding derivation

$$
A \Rightarrow a a A \Rightarrow a a a b b c
$$

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

```
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
\[
\begin{aligned}
& S \rightarrow a S b|\lambda| A \\
& A \rightarrow a A
\end{aligned}
\]
the production S C clearly plays no role, as }A\mathrm{ cannot be transformed into a
terminal string. While }A\mathrm{ 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

### 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->aSb|||A,
the production S C A clearly plays no role, as A cannot be transformed into a
terminal string. While }A\mathrm{ 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

## 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
the production $S \rightarrow A$ 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

## 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
the production $S \rightarrow A$ 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

## 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

$$
\begin{aligned}
& S \rightarrow a S b|\lambda| A, \\
& A \rightarrow a A,
\end{aligned}
$$

the production $S \rightarrow A$ 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

## 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

$$
\begin{aligned}
& S \rightarrow a S b|\lambda| A, \\
& A \rightarrow a A,
\end{aligned}
$$

the production $S \rightarrow A$ 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

## 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

$$
\begin{aligned}
& S \rightarrow a S b|\lambda| A, \\
& A \rightarrow a A,
\end{aligned}
$$

the production $S \rightarrow A$ 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

## 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

$$
\begin{aligned}
& S \rightarrow a S b|\lambda| A \\
& A \rightarrow a A
\end{aligned}
$$

the production $S \rightarrow A$ clearly plays no role, as $A$ cannot be transformed into a terminal string. While $A$ can occur in a string derived from $S$, lead to a sentence Removing this production leaves the language unaffected and is a simplification by any definition.

[^3]
### 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

$$
\begin{aligned}
& S \rightarrow a S b|\lambda| A, \\
& A \rightarrow a A,
\end{aligned}
$$

the production $S \rightarrow A$ 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.

[^4]
## 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

$$
\begin{aligned}
& S \rightarrow a S b|\lambda| A \\
& A \rightarrow a A
\end{aligned}
$$

the production $S \rightarrow A$ 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.

## 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

$$
\begin{aligned}
& S \rightarrow a S b|\lambda| A \\
& A \rightarrow a A
\end{aligned}
$$

the production $S \rightarrow A$ 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

$$
S \Rightarrow x A y \Rightarrow w \text {. }
$$

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

$$
\begin{aligned}
& S \rightarrow a S b|\lambda| A \\
& A \rightarrow a A
\end{aligned}
$$

the production $S \rightarrow A$ 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.

## 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

$$
\begin{aligned}
& S \rightarrow a S b|\lambda| A \\
& A \rightarrow a A
\end{aligned}
$$

the production $S \rightarrow A$ 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
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

$$
\begin{aligned}
& S \rightarrow a S b|\lambda| A \\
& A \rightarrow a A
\end{aligned}
$$

the production $S \rightarrow A$ 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

$$
\begin{equation*}
S \stackrel{*}{\Rightarrow} x A y \stackrel{*}{\Rightarrow} w . \tag{2}
\end{equation*}
$$

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

$$
\begin{aligned}
& S \rightarrow a S b|\lambda| A \\
& A \rightarrow a A
\end{aligned}
$$

the production $S \rightarrow A$ 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

$$
\begin{equation*}
S \stackrel{*}{\Rightarrow} x A y \stackrel{*}{\Rightarrow} w . \tag{2}
\end{equation*}
$$

with $x, y \in(V \cup T)^{*}$.
In words, a variable
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

$$
\begin{aligned}
& S \rightarrow a S b|\lambda| A \\
& A \rightarrow a A
\end{aligned}
$$

the production $S \rightarrow A$ 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

$$
\begin{equation*}
S \stackrel{*}{\Rightarrow} x A y \stackrel{*}{\Rightarrow} w . \tag{2}
\end{equation*}
$$

with $x, y \in(V \cup T)^{*}$. In words, a variable is useful if and only if it occurs in at least one derivation.
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

$$
\begin{aligned}
& S \rightarrow a S b|\lambda| A, \\
& A \rightarrow a A
\end{aligned}
$$

the production $S \rightarrow A$ 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

$$
\begin{equation*}
S \stackrel{*}{\Rightarrow} x A y \stackrel{*}{\Rightarrow} w . \tag{2}
\end{equation*}
$$

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.

## 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

$$
\begin{aligned}
& S \rightarrow a S b|\lambda| A, \\
& A \rightarrow a A
\end{aligned}
$$

the production $S \rightarrow A$ 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

$$
\begin{equation*}
S \stackrel{*}{\Rightarrow} x A y \stackrel{*}{\Rightarrow} w . \tag{2}
\end{equation*}
$$

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
```

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.

### 6.1 Methods for Transforming Grammars

## 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->A,
    A->a\Delta|\lambda
    B->bA,
```

the variable $B$ is useless and so is the production $B \rightarrow b A$. Although $B$ can
derive a terminal string, there is no way we can achieve $S \stackrel{*}{\Rightarrow} x B y$.

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
the variable $B$ is useless and so is the production $B \rightarrow b A$. Although $B$ can derive a terminal string, there is no way we can achieve $S \stackrel{*}{\Rightarrow} x B y$.

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.
be useless is shown in the next grammar. In a grammar with start symbol $S$ and productions
the variable $B$ is useless and so is the production $B \rightarrow b A$. Although $B$ can derive a terminal string, there is no way we can achieve $S \stackrel{*}{\Rightarrow} x B y$.

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
the variable $B$ is useless and so is the production $B \rightarrow b A$. Although $B$ can derive a terminal string, there is no way we can achieve $S \stackrel{*}{\Rightarrow} x B y$.

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
the variable $B$ is useless and so is the production $B \rightarrow b A$. Although $B$ can derive a terminal string, there is no way we can achieve $S \stackrel{*}{\Rightarrow} x B y$.

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

$$
\begin{aligned}
& S \rightarrow A \\
& A \rightarrow a A \mid \lambda \\
& B \rightarrow b A
\end{aligned}
$$

the variable $B$ is useless and so is the production $B \rightarrow b A$. Although $B$ can derive a terminal string, there is no way we can achieve $S \stackrel{*}{\Rightarrow} x B y$.

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

$$
\begin{aligned}
& S \rightarrow A \\
& A \rightarrow a A \mid \lambda \\
& B \rightarrow b A
\end{aligned}
$$

the variable $B$ is useless and so is the production $B \rightarrow b A$. Although $B$ can
derive a terminal string, there is no way we can achieve $S \stackrel{*}{\Rightarrow} x B y$.
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

$$
\begin{aligned}
& S \rightarrow A \\
& A \rightarrow a A \mid \lambda \\
& B \rightarrow b A
\end{aligned}
$$

the variable $B$ is useless and so is the production $B \rightarrow b A$. Although $B$ can derive a terminal string, there is no way we can achieve $S \stackrel{*}{\Rightarrow} x B y$.

> 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

$$
\begin{aligned}
& S \rightarrow A \\
& A \rightarrow a A \mid \lambda \\
& B \rightarrow b A
\end{aligned}
$$

the variable $B$ is useless and so is the production $B \rightarrow b A$. Although $B$ can derive a terminal string, there is no way we can achieve $S \stackrel{*}{\Rightarrow} x B y$.

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

$$
\begin{aligned}
& S \rightarrow A \\
& A \rightarrow a A \mid \lambda \\
& B \rightarrow b A
\end{aligned}
$$

the variable $B$ is useless and so is the production $B \rightarrow b A$. Although $B$ can derive a terminal string, there is no way we can achieve $S \stackrel{*}{\Rightarrow} x B y$.

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.
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

$$
\begin{aligned}
& S \rightarrow A \\
& A \rightarrow a A \mid \lambda \\
& B \rightarrow b A
\end{aligned}
$$

the variable $B$ is useless and so is the production $B \rightarrow b A$. Although $B$ can derive a terminal string, there is no way we can achieve $S \stackrel{*}{\Rightarrow} x B y$.

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

$$
\begin{aligned}
& S \rightarrow A \\
& A \rightarrow a A \mid \lambda \\
& B \rightarrow b A
\end{aligned}
$$

the variable $B$ is useless and so is the production $B \rightarrow b A$. Although $B$ can derive a terminal string, there is no way we can achieve $S \stackrel{*}{\Rightarrow} x B y$.

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

### 6.1 Methods for Transforming Grammars

## 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 $S \rightarrow a S|A| C$, $A \rightarrow a$, $B \rightarrow a a$, $C \rightarrow a C b$

First, we identify the set of variables that can lead to a terminal word. Since $A \rightarrow a$ and $B \rightarrow a a$, the variables $A$ and $B$ belong to this set. So does $S$, because $S \Rightarrow A \Rightarrow a$. However, this argument cannot be made for $C$, thus identifying it as useless. Removing $C$ and its corresponding productions, we are led to the grammar $G_{1}$ with variables $V_{1}=\{S, A, B\}$, terminals $T=\{a\}$, and productions

$$
\begin{aligned}
& S \rightarrow a S \mid A \\
& A \rightarrow a \\
& B \rightarrow a a
\end{aligned}
$$

## 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

$$
S \rightarrow a S|A| C,
$$

$A \rightarrow a$,
$B \rightarrow a n$

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


## 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


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


## Example 6.3

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

$$
\begin{aligned}
& S \rightarrow a S|A| C \\
& A \rightarrow a \\
& B \rightarrow a a \\
& C \rightarrow a C b
\end{aligned}
$$

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


## Example 6.3

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

$$
\begin{aligned}
& S \rightarrow a S|A| C, \\
& A \rightarrow a \\
& B \rightarrow a a \\
& C \rightarrow a C b
\end{aligned}
$$

First, we identify the set of variables that can lead to a terminal word. Since 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


## Example 6.3

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

$$
\begin{aligned}
& S \rightarrow a S|A| C, \\
& A \rightarrow a \\
& B \rightarrow a a \\
& C \rightarrow a C b
\end{aligned}
$$

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


## Example 6.3

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

$$
\begin{aligned}
& S \rightarrow a S|A| C, \\
& A \rightarrow a \\
& B \rightarrow a a \\
& C \rightarrow a C b
\end{aligned}
$$

First, we identify the set of variables that can lead to a terminal word. Since $A \rightarrow a$ and $B \rightarrow a a$, the variables $A$ and $B$ belong to this set. So does $S$, because $S \Rightarrow A \Rightarrow a$.
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

## Example 6.3

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

$$
\begin{aligned}
& S \rightarrow a S|A| C, \\
& A \rightarrow a \\
& B \rightarrow a a \\
& C \rightarrow a C b
\end{aligned}
$$

First, we identify the set of variables that can lead to a terminal word. Since $A \rightarrow a$ and $B \rightarrow a a$, 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.
led to the grammar $G_{1}$ with variables $V_{1}=\{S, A, B\}$, terminals $T=\{a\}$, and productions

## Example 6.3

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

$$
\begin{aligned}
& S \rightarrow a S|A| C, \\
& A \rightarrow a \\
& B \rightarrow a a \\
& C \rightarrow a C b
\end{aligned}
$$

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

## Example 6.3

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

$$
\begin{aligned}
& S \rightarrow a S|A| C, \\
& A \rightarrow a \\
& B \rightarrow a a \\
& C \rightarrow a C b
\end{aligned}
$$

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

## Example 6.3

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

$$
\begin{aligned}
& S \rightarrow a S|A| C, \\
& A \rightarrow a \\
& B \rightarrow a a \\
& C \rightarrow a C b
\end{aligned}
$$

First, we identify the set of variables that can lead to a terminal word. Since $A \rightarrow a$ and $B \rightarrow a a$, the variables $A$ and $B$ belong to this set. So does $S$, because $S \Rightarrow A \Rightarrow a$. However, this argument cannot be made for $C$, thus identifying it as useless. Removing $C$ and its corresponding productions, we are led to the grammar $G_{1}$ with variables $V_{1}=\{S, A, B\}$, terminals $T=\{a\}$, and productions

$$
\begin{aligned}
& S \rightarrow a S \mid A \\
& A \rightarrow a \\
& B \rightarrow a a
\end{aligned}
$$

## Example 6.3 (continuation)

### 6.1 Methods for Transforming Grammars

## 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 $C \rightarrow x D y$.
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

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

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

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,
if and only if there is a production of the form
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

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

$$
C \rightarrow x D y
$$

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

$$
C \rightarrow x D y
$$

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

$$
C \rightarrow x D y
$$

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

$$
C \rightarrow x D y
$$

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.
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

$$
C \rightarrow x D y
$$

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)

### 6.1 Methods for Transforming Grammars

## 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


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

### 6.1 Methods for Transforming Grammars

## 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

$A \rightarrow a$.
The formalization of this process leads to a general construction and the corresponding theorem.

### 6.1 Methods for Transforming Grammars

## 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

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

### 6.1 Methods for Transforming Grammars

## 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

$$
\begin{aligned}
& S \rightarrow a S \mid A \\
& A \rightarrow a
\end{aligned}
$$

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

$$
\begin{aligned}
& S \rightarrow a S \mid A \\
& A \rightarrow a
\end{aligned}
$$

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

## Theorem 6.2

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 $G_{1}=\left(V_{1}, T_{2}, S, P_{1}\right)$ such that $V_{1}$ contains only variables $A$ for which $A \stackrel{*}{\Rightarrow} w \in T^{*}$
is possible. The steps in the algorithm are

## 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 $G_{1}=\left(V_{1}, T_{2}, S, P_{1}\right)$ such that $V_{1}$ contains only variables $A$ for which $A \stackrel{*}{\Rightarrow} w \in T^{*}$
is possible. The steps in the algorithm are

## 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 $G_{1}=\left(V_{1}, T_{2}, S, P_{1}\right)$ such that $V_{1}$ contains only variables $A$ for which $A \stackrel{*}{\Rightarrow} w \in T^{*}$
is possible. The steps in the algorithm are

## 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 $G_{1}=\left(V_{1}, T_{2}, S, P_{1}\right)$ such that $V_{1}$ contains only variables $A$ for which is possible. The steps in the algorithm are

## 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 $G$ can be generated from $G$ by an algorithm consisting of two parts. In the first part we construct an intermediate grammar $G_{1}=\left(V_{1}, T_{2}, S, P_{1}\right)$ such that $V_{1}$ contains only variables $A$ for which is possible. The steps in the algorithm are

## 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 $G_{1}=\left(V_{1}, T_{2}, S, P_{1}\right)$ such that $V_{1}$ contains only variables $A$ for which

[^5]
## 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 $G_{1}=\left(V_{1}, T_{2}, S, P_{1}\right)$ such that $V_{1}$ contains only variables $A$ for which is possible. The steps in the algorithm are

## 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 $G_{1}=\left(V_{1}, T_{2}, S, P_{1}\right)$ such that $V_{1}$ contains only variables $A$ for which

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

is possible. The steps in the algorithm are

## 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 $G_{1}=\left(V_{1}, T_{2}, S, P_{1}\right)$ such that $V_{1}$ contains only variables $A$ for which

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

is possible. The steps in the algorithm are

## 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 $G_{1}=\left(V_{1}, T_{2}, S, P_{1}\right)$ such that $V_{1}$ contains only variables $A$ for which

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

is possible. The steps in the algorithm are

## 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 $G_{1}=\left(V_{1}, T_{2}, S, P_{1}\right)$ such that $V_{1}$ contains only variables $A$ for which

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

is possible. The steps in the algorithm are
(2) Repeat the following step until no more variables are added to $V_{1}$
(3) Take $P_{1}$ as all the productions in $P$ whose symbols are all in $\left(V_{1} \cup T\right)$.

## 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 $G_{1}=\left(V_{1}, T_{2}, S, P_{1}\right)$ 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 $\varnothing$.
(3) Repeat the following step until no more variables are added to $V_{1}$
(3) Take $P_{1}$ as all the productions in $P$ whose symbols are all in $\left(V_{1} \cup T\right)$.

## 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 $G_{1}=\left(V_{1}, T_{2}, S, P_{1}\right)$ 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 $V_{1}$.
every $A \in V$ for which $P$ has a production of the form
add $A$ to $V_{1}$

- Take $P_{1}$ as all the productions in $P$ whose symbols are all in $\left(V_{1} \cup T\right)$.


## 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 $G_{1}=\left(V_{1}, T_{2}, S, P_{1}\right)$ 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

## 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 $G_{1}=\left(V_{1}, T_{2}, S, P_{1}\right)$ 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

$$
A \rightarrow x_{1} x_{2} \cdots x_{n}, \text { with all } x_{i} \in V_{1} \cup T
$$

add $A$ to $V_{1}$
(3) Take $P_{1}$ as all the productions in $P$ whose symbols are all in $\left(V_{1} \cup T\right)$

## 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 $G_{1}=\left(V_{1}, T_{2}, S, P_{1}\right)$ 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

$$
A \rightarrow x_{1} x_{2} \cdots x_{n}, \text { with all } x_{i} \in V_{1} \cup T
$$

add $A$ to $V_{1}$.

## Theorem 6.2

Let $G=(V, T, S, P)$ be a context-free grammar. Then there exists an equivalent grammar $\widehat{G}=(\widehat{V}, \widehat{T}, S, \widehat{P})$ that does not contain any 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 $G_{1}=\left(V_{1}, T_{2}, S, P_{1}\right)$ 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

$$
A \rightarrow x_{1} x_{2} \cdots x_{n}, \text { with all } x_{i} \in V_{1} \cup T
$$

add $A$ to $V_{1}$.
(3) Take $P_{1}$ as all the productions in $P$ whose symbols are all in $\left(V_{1} \cup T\right)$.

### 6.1 Methods for Transforming Grammars

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 $A \stackrel{*}{\Rightarrow} w=a b \cdots$ 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).


### 6.1 Methods for Transforming Grammars

Clearly this procedure terminates. It is equally clear that if $A \in V_{1}$, then $A \Rightarrow w \in T^{*}$ is a possible derivation with $G_{1}$. The remaining issue is whether every $A$ for which $A \stackrel{*}{\Rightarrow} w=a b \cdots$ 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).


### 6.1 Methods for Transforming Grammars

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 $A \stackrel{*}{\Rightarrow} w=a b \cdots$ 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).


### 6.1 Methods for Transforming Grammars

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 $A \stackrel{*}{\Rightarrow} w=a b \cdots$ is added to $V_{1}$ before the procedure terminates. tree corresponding to that derivation (see the Figure).


### 6.1 Methods for Transforming Grammars

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 $A \stackrel{*}{\Rightarrow} w=a b \cdots$ 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)


### 6.1 Methods for Transforming Grammars

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 $A \stackrel{*}{\Rightarrow} w=a b \cdots$ 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).


### 6.1 Methods for Transforming Grammars

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 $A \stackrel{*}{\Rightarrow} w=a b \cdots$ 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).


### 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

$$
S \stackrel{*}{\Rightarrow} x A y \stackrel{*}{\Rightarrow} w .
$$

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

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

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. 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, $G$ does not contain any useless symbols or productions. Also, for each $w \in L(G)$ we have a derivation

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

The grammar $G$ 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 $G$ 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 algorithm cannot terminate while there are variables in the tree that are
not yet in }\mp@subsup{V}{1}{}\mathrm{ . Hence }A\mathrm{ will eventually be added to }\mp@subsup{V}{1}{
In the second part of the construction, we get the final answer G from G}\mp@subsup{G}{1}{}\mathrm{ . We
draw the variable dependency graph for G}\mp@subsup{G}{1}{}\mathrm{ 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 G}=(\widehat{V},\widehat{T},S,\widehat{P})\mathrm{ .
Because of the construction, }G\mathrm{ does not contain any useless symbols or
productions. Also, for each w\inL(G) we have a derivation
```

Since the construction of $\widehat{G}$ retains $A$ and all associated productions, we have
everything needed to make the derivation
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, 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 $G$ 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 $G=(V, T, S, P)$ 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

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

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.
$\square$
not yet in $V_{1}$. Hence $A$ will eventually be added to $V$
In the second part of the construction, we get the final answer $G$ 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 $G=(V, T, S, P)$.
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

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

### 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 $V_{1}$.
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 $G=(V, T, S, P)$. 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

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

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 $G$ 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 $V_{1}$.
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 $G=(V, T, S, P)$. 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

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

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 $G$ 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 $G_{1}$.
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 $G=(V, T, S, P)$. 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

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

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 $S$.
> 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

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

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 $G$ 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.

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

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

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

> 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

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

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

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

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

$$
\text { that } G \text { and } \widehat{G} \text { are equivalent. }
$$

### 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.

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

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

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

> 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

$$
S \stackrel{*}{\Rightarrow} x A y \stackrel{*}{\Rightarrow} w .
$$

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

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

$$
S \stackrel{*}{\Rightarrow} x A y \stackrel{*}{\Rightarrow} w .
$$

Since the construction of $\widehat{G}$ retains $A$ and all associated productions, we have

> 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

$$
S \stackrel{*}{\Rightarrow} x A y \stackrel{*}{\Rightarrow} w .
$$

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

> 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

$$
S \stackrel{*}{\Rightarrow} x A y \stackrel{*}{\Rightarrow} w .
$$

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

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

The grammar $G$ 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

$$
S \stackrel{*}{\Rightarrow} x A y \stackrel{*}{\Rightarrow} w .
$$

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

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

The grammar $\widehat{G}$ is constructed from $G$ by the removal of productions, 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

$$
S \stackrel{*}{\Rightarrow} x A y \stackrel{*}{\Rightarrow} w .
$$

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

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

The grammar $\widehat{G}$ is constructed from $G$ by the removal of productions, so that $\widehat{P} \subset P$. 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

$$
S \stackrel{*}{\Rightarrow} x A y \stackrel{*}{\Rightarrow} w .
$$

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

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

The grammar $\widehat{G}$ is constructed from $G$ by the removal of productions, so that $\widehat{P} \subset P$. Consequently $L(\widehat{G}) \subseteq L(G)$.

### 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

$$
S \stackrel{*}{\Rightarrow} x A y \stackrel{*}{\Rightarrow} w .
$$

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

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

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

$$
S \stackrel{*}{\Rightarrow} x A y \stackrel{*}{\Rightarrow} w .
$$

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

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

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 }\lambda\mathrm{ -Productions
One kind of production that is sometimes undesirable is one in which the right
side is the empty string.
```


## Definition 6.2

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

## Removing $\lambda$-Productions

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

## Definition 6.2

[^6]
### 6.1 Methods for Transforming Grammars

## Removing $\lambda$-Productions

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

## Definition 6.2

[^7]
## Removing $\lambda$-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
is called a $\lambda$-production. Any variable $A$ for which the derivation $A \stackrel{*}{\Rightarrow} \lambda$
is possible is called nullable.
A grammar may generate a language not containing $\lambda$, yet have some $\lambda$-productions or nullable variables. In such cases, the $\lambda$-productions can be removed.

### 6.1 Methods for Transforming Grammars

Removing $\lambda$-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
is called a $\lambda$-production. Any variable $A$ for which the derivation
is possible is called nullable.

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

## Removing $\lambda$-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

$$
A \rightarrow \lambda
$$

is called a $\lambda$-production. Any variable $A$ for which the derivation
is possible is called nullable.

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

### 6.1 Methods for Transforming Grammars

Removing $\lambda$-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

$$
A \rightarrow \lambda
$$

is called a $\lambda$-production. Any variable $A$ for which the derivation
is possible is called nullable.

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

Removing $\lambda$-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

$$
A \rightarrow \lambda
$$

is called a $\lambda$-production. Any variable $A$ for which the derivation
is possible is called nullable.

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

Removing $\lambda$-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

$$
A \rightarrow \lambda
$$

is called a $\lambda$-production. Any variable $A$ for which the derivation

$$
\begin{equation*}
A \stackrel{*}{\Rightarrow} \lambda \tag{3}
\end{equation*}
$$

is possible is called nullable.
A grammar may generate a language not containing $\lambda$, yet have some $\lambda$-productions or nullable variables. In such cases, the $\lambda$-productions can be removed.

Removing $\lambda$-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

$$
A \rightarrow \lambda
$$

is called a $\lambda$-production. Any variable $A$ for which the derivation

$$
\begin{equation*}
A \stackrel{*}{\Rightarrow} \lambda \tag{3}
\end{equation*}
$$

is possible is called nullable.
A grammar may generate a language not containing $\lambda$, yet have some $\lambda$-productions or nullable variables. In such cases, the $\lambda$-productions can be removed.

Removing $\lambda$-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

$$
A \rightarrow \lambda
$$

is called a $\lambda$-production. Any variable $A$ for which the derivation

$$
\begin{equation*}
A \stackrel{*}{\Rightarrow} \lambda \tag{3}
\end{equation*}
$$

is possible is called nullable.
A grammar may generate a language not containing $\lambda$,
$\lambda$-productions or nullable variables. In such cases, the $\lambda$-productions can be
removed.

Removing $\lambda$-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

$$
A \rightarrow \lambda
$$

is called a $\lambda$-production. Any variable $A$ for which the derivation

$$
\begin{equation*}
A \stackrel{*}{\Rightarrow} \lambda \tag{3}
\end{equation*}
$$

is possible is called nullable.
A grammar may generate a language not containing $\lambda$, yet have some $\lambda$-productions or nullable variables. In such cases, the $\lambda$-productions can be removed.

## Example 6.4

### 6.1 Methods for Transforming Grammars

## Example 6.4

Consider the grammar

$$
\begin{aligned}
S & \rightarrow a S_{1} b \\
S_{1} & \rightarrow a S_{1} b \mid \lambda
\end{aligned}
$$

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

$$
\begin{aligned}
& S \rightarrow a S_{1} b \mid a b \\
& S_{1} \rightarrow a S_{1} b \mid a b
\end{aligned}
$$

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

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

## Example 6.4

Consider the grammar

$$
\begin{aligned}
S & \rightarrow a S_{1} b \\
S_{1} & \rightarrow a S_{1} b \mid \lambda
\end{aligned}
$$

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

$$
\begin{aligned}
& S \rightarrow a S_{1} b \mid a b \\
& S_{1} \rightarrow a S_{1} b \mid a b
\end{aligned}
$$

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

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

## Example 6.4

Consider the grammar

$$
\begin{aligned}
S & \rightarrow a S_{1} b \\
S_{1} & \rightarrow a S_{1} b \mid \lambda
\end{aligned}
$$

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

$$
\begin{aligned}
S & \rightarrow a S_{1} b \mid a b \\
S_{1} & \rightarrow a S_{1} b \mid a b .
\end{aligned}
$$

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

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

## Example 6.4

Consider the grammar

$$
\begin{aligned}
S & \rightarrow a S_{1} b \\
S_{1} & \rightarrow a S_{1} b \mid \lambda
\end{aligned}
$$

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

$$
\begin{aligned}
S & \rightarrow a S_{1} b \mid a b \\
S_{1} & \rightarrow a S_{1} b \mid a b
\end{aligned}
$$

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

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

## Example 6.4

Consider the grammar

$$
\begin{aligned}
S & \rightarrow a S_{1} b \\
S_{1} & \rightarrow a S_{1} b \mid \lambda
\end{aligned}
$$

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

## Example 6.4

Consider the grammar

$$
\begin{aligned}
S & \rightarrow a S_{1} b, \\
S_{1} & \rightarrow a S_{1} b \mid \lambda
\end{aligned}
$$

with start variable $S$. This grammar generates the $\lambda$-free language $\left\{a^{n} b^{n}: n \geqslant 1\right\}$. The $\lambda$-production $S_{1} \rightarrow \lambda$ can be removed after adding new productions obtained by substituting $\lambda$ for $S_{1}$ where it occurs on the right. Doing this we get the grammar
$\square$
We can easily show that this new grammar generates the same language as the original one.

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

## Example 6.4

Consider the grammar

$$
\begin{aligned}
S & \rightarrow a S_{1} b, \\
S_{1} & \rightarrow a S_{1} b \mid \lambda
\end{aligned}
$$

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

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

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

## Example 6.4

Consider the grammar

$$
\begin{aligned}
S & \rightarrow a S_{1} b, \\
S_{1} & \rightarrow a S_{1} b \mid \lambda
\end{aligned}
$$

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

$$
\begin{aligned}
S & \rightarrow a S_{1} b \mid a b \\
S_{1} & \rightarrow a S_{1} b \mid a b
\end{aligned}
$$

We can easily show that this new grammar generates the same language as the original one.
In more general situations, substitutions for $\lambda$-productions can be made in a similar, although more complicated, manner.

## Example 6.4

Consider the grammar

$$
\begin{aligned}
S & \rightarrow a S_{1} b, \\
S_{1} & \rightarrow a S_{1} b \mid \lambda
\end{aligned}
$$

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

$$
\begin{aligned}
S & \rightarrow a S_{1} b \mid a b \\
S_{1} & \rightarrow a S_{1} b \mid a b
\end{aligned}
$$

We can easily show that this new grammar generates the same language as the original one.
In more general situations, substitutions for $\lambda$-productions can be made in a
similar, although more complicated, manner.

## Example 6.4

Consider the grammar

$$
\begin{aligned}
S & \rightarrow a S_{1} b \\
S_{1} & \rightarrow a S_{1} b \mid \lambda
\end{aligned}
$$

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

$$
\begin{aligned}
S & \rightarrow a S_{1} b \mid a b \\
S_{1} & \rightarrow a S_{1} b \mid a b
\end{aligned}
$$

We can easily show that this new grammar generates the same language as the original one.
In more general situations, substitutions for $\lambda$-productions can be made in a similar,

## Example 6.4

Consider the grammar

$$
\begin{aligned}
S & \rightarrow a S_{1} b \\
S_{1} & \rightarrow a S_{1} b \mid \lambda
\end{aligned}
$$

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

$$
\begin{aligned}
S & \rightarrow a S_{1} b \mid a b \\
S_{1} & \rightarrow a S_{1} b \mid a b
\end{aligned}
$$

We can easily show that this new grammar generates the same language as the original one.
In more general situations, substitutions for $\lambda$-productions can be made in a similar, although more complicated, manner.

### 6.1 Methods for Transforming Grammars

Theorem 6.3

Proof. We first find the set $V_{N}$ of all nullable variables of $G$, using the
following steps.

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

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 $\lambda$ 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 $\lambda$, one in which $x_{j}$ is replaced with $\lambda$, and one in which both $x_{i}$ and $x_{i}$ are replaced with $\lambda$. There is one exception: If all $x_{i}$ are nullable, then the production $A \rightarrow \lambda$ is not put into $\widehat{P}$.
The argument that this grammar $\widehat{G}$ is equivalent to $G$ is straightforward and will be left to the reader.

### 6.1 Methods for Transforming Grammars

Theorem 6.3
Let $G$ be any context-free grammar with $\lambda$ not in $L(G)$. Then there exists an equivalent grammar $\widehat{G}$ having no $\lambda$-productions.

Proot. We first find the set $V_{N}$ of all nullable variables of $G$, using the following steps.

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
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 $\lambda$ 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 $\lambda$, one in which $x_{j}$ is replaced with $\lambda$, and one in which both $x_{i}$ and $x_{j}$ are replaced with $\lambda$. There is one exception: If all $x_{i}$ are nullable, then the production $A \rightarrow \lambda$ is not put into $\widehat{P}$.
The argument that this grammar $\widehat{G}$ is equivalent to $G$ is straightforward and will be left to the reader.

### 6.1 Methods for Transforming Grammars

Theorem 6.3
Let $G$ be any context-free grammar with $\lambda$ not in $L(G)$. Then there exists an
equivalent grammar $\widehat{G}$ having no $\lambda$-productions.
Proof. We first find the set $V_{N}$ of all nullable variables of $G$, using the
following steps.

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
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 $\lambda$ 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 $\lambda$, one in which $x_{j}$ is replaced with $\lambda$, and one in which both $x_{i}$ and $x_{i}$ are replaced with $\lambda$. There is one exception: If all $x_{i}$ are nullable, then the production $A \rightarrow \lambda$ is not put into $\widehat{P}$. The argument that this grammar $\widehat{G}$ is equivalent to $G$ is straightforward and will be left to the reader.

### 6.1 Methods for Transforming Grammars

Theorem 6.3
Let $G$ be any context-free grammar with $\lambda$ not in $L(G)$. Then there exists an equivalent grammar $\widehat{G}$ having no $\lambda$-productions.

```
Proot. We tirst find the set }\mp@subsup{V}{N}{}\mathrm{ of all nullable variables of }G\mathrm{ , using the
following steps.
```

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
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 $\lambda$
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 $\lambda$, one in which $x_{j}$ is replaced
with $\lambda$, and one in which both $x_{i}$ and $x_{j}$ are replaced with $\lambda$. There is one
exception: If all $x_{i}$ are nullable, then the production $A \rightarrow \lambda$ is not put into $\widehat{P}$.
The argument that this grammar $\bar{G}$ 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 $\lambda$ not in $L(G)$. Then there exists an equivalent grammar $\widehat{G}$ having no $\lambda$-productions.
Proof. We first find the set $V_{N}$ of all nullable variables of $G$, using the
following steps.

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
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 $\lambda$ 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 $\lambda$, one in which $x_{j}$ is replaced with $\lambda$, and one in which both $x_{i}$ and $x_{j}$ are replaced with $\lambda$. There is one exception: If all $x_{i}$ are nullable, then the production $A \rightarrow \lambda$ is not put into $\widehat{P}$
The argument that this grammar $\widehat{G}$ is equivalent to $G$ is straightforward and will be left to the reader.

### 6.1 Methods for Transforming Grammars

Theorem 6.3
Let $G$ be any context-free grammar with $\lambda$ not in $L(G)$. Then there exists an equivalent grammar $\widehat{G}$ having no $\lambda$-productions.

Proof. We first find the set $V_{N}$ of all nullable variables of $G$, using the following steps.

```
Once the set }\mp@subsup{V}{N}{}\mathrm{ has been found, we are ready to construct }P\mathrm{ . To do so, we
look at all productions in P}\mathrm{ of the form
where each }\mp@subsup{x}{i}{}\inV\cupT\mathrm{ . For each such production of P, we put into P}\mathrm{ that
production as well as all those generated by replacing nullable variables with }
in all possible combinations. For example, if }\mp@subsup{x}{i}{}\mathrm{ and }\mp@subsup{x}{j}{}\mathrm{ are both nullable, there
will be one production in }\widehat{P}\mathrm{ with }\mp@subsup{x}{i}{}\mathrm{ replaced with }\lambda\mathrm{ , one in which }\mp@subsup{x}{j}{}\mathrm{ is replaced
with }\lambda\mathrm{ , and one in which both }\mp@subsup{x}{i}{}\mathrm{ and }\mp@subsup{x}{j}{}\mathrm{ are replaced with }\lambda\mathrm{ . There is one
exception: If all }\mp@subsup{x}{i}{}\mathrm{ are nullable, then the production A}->\lambda\mathrm{ is not put into }\hat{P}\mathrm{ .
The argument that this grammar }\widehat{G}\mathrm{ 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 $\lambda$ not in $L(G)$. Then there exists an equivalent grammar $\widehat{G}$ having no $\lambda$-productions.

Proof. We first find the set $V_{N}$ of all nullable variables of $G$, using the following steps.
(1) For all productions $A \rightarrow \lambda$, put $A$ into $V_{A}$
(3) Repeat the following step until no further variables are added to $V_{N}$ For all productions

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

```
where each }\mp@subsup{x}{i}{}\inV\cupT\mathrm{ . For each such production of P, we put into P}\mathrm{ that
production as well as all those generated by replacing nullable variables with }
in all possible combinations. For example, if }\mp@subsup{x}{i}{}\mathrm{ and }\mp@subsup{x}{j}{}\mathrm{ are both nullable, there
will be one production in \widehat{P}\mathrm{ with }\mp@subsup{x}{i}{}\mathrm{ replaced with }\lambda\mathrm{ , one in which }\mp@subsup{x}{j}{}\mathrm{ is replaced}
with }\lambda\mathrm{ , and one in which both }\mp@subsup{x}{i}{}\mathrm{ and }\mp@subsup{x}{j}{}\mathrm{ are replaced with }\lambda\mathrm{ . There is one
exception: If all }\mp@subsup{x}{i}{}\mathrm{ are nullable, then the production }A->\lambda\mathrm{ is not put into }\widehat{P}\mathrm{ .
The argument that this grammar G}\mathrm{ is equivalent to }G\mathrm{ 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 $\lambda$ not in $L(G)$. Then there exists an equivalent grammar $\widehat{G}$ having no $\lambda$-productions.

Proof. We first find the set $V_{N}$ of all nullable variables of $G$, using the following steps.
(1) For all productions $A \rightarrow \lambda$, put $A$ into $V_{N}$.
(3) Repeat the following step until no further variables are added to $V_{1}$ For all productions where $A_{1}, A_{2}$

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

```
where each }\mp@subsup{x}{i}{}\inV\cupT\mathrm{ . For each such production of P, we put into P that
production as well as all those generated by replacing nullable variables with }
in all possible combinations. For example, if }\mp@subsup{x}{i}{}\mathrm{ and }\mp@subsup{x}{j}{}\mathrm{ are both nullable, there
will be one production in }\widehat{P}\mathrm{ with }\mp@subsup{x}{i}{}\mathrm{ replaced with }\lambda\mathrm{ , one in which }\mp@subsup{x}{j}{}\mathrm{ is replaced
with }\lambda\mathrm{ , and one in which both }\mp@subsup{x}{i}{}\mathrm{ and }\mp@subsup{x}{j}{}\mathrm{ are replaced with }\lambda\mathrm{ . There is one
exception: If all }\mp@subsup{x}{i}{}\mathrm{ are nullable, then the production }A->\lambda\mathrm{ is not put into }\widehat{P}\mathrm{ .
The argument that this grammar }G\mathrm{ 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 $\lambda$ not in $L(G)$. Then there exists an equivalent grammar $\widehat{G}$ having no $\lambda$-productions.

Proof. We first find the set $V_{N}$ of all nullable variables of $G$, using the following steps.
(1) For all productions $A \rightarrow \lambda$, put $A$ into $V_{N}$.
(2) Repeat the following step until no further variables are added to $V_{N}$.

For all productions

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

### 6.1 Methods for Transforming Grammars

Theorem 6.3
Let $G$ be any context-free grammar with $\lambda$ not in $L(G)$. Then there exists an equivalent grammar $\widehat{G}$ having no $\lambda$-productions.

Proof. We first find the set $V_{N}$ of all nullable variables of $G$, using the following steps.
(1) For all productions $A \rightarrow \lambda$, put $A$ into $V_{N}$.
(2) Repeat the following step until no further variables are added to $V_{N}$. For all productions

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

### 6.1 Methods for Transforming Grammars

Theorem 6.3
Let $G$ be any context-free grammar with $\lambda$ not in $L(G)$. Then there exists an equivalent grammar $\widehat{G}$ having no $\lambda$-productions.

Proof. We first find the set $V_{N}$ of all nullable variables of $G$, using the following steps.
(1) For all productions $A \rightarrow \lambda$, put $A$ into $V_{N}$.
(2) Repeat the following step until no further variables are added to $V_{N}$. For all productions

$$
B \rightarrow A_{1} A_{2} \cdots A_{n},
$$

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

### 6.1 Methods for Transforming Grammars

## Theorem 6.3

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

Proof. We first find the set $V_{N}$ of all nullable variables of $G$, using the following steps.
(1) For all productions $A \rightarrow \lambda$, put $A$ into $V_{N}$.
(2) Repeat the following step until no further variables are added to $V_{N}$. For all productions

$$
B \rightarrow A_{1} A_{2} \cdots A_{n}
$$ where $A_{1}, A_{2}, \cdots, A_{n}$ are in $V_{N}$, put $B$ into $V_{N}$.

```
Once the set VN has been found, we are ready to construct P. To do so, we
```

look at all productions in $P$ of the form
where each $x_{i} \in V \cup T$. For each such production of $P$, we put into $P$ that
production as well as all those generated by replacing nullable variables with $\lambda$
in all possible combinations. For example, if $x_{i}$ and $x_{j}$ are both nullable, there
will be one production in $P$ with $x_{i}$ replaced with $\lambda$, one in which $x_{j}$ is replaced
with $\lambda$, and one in which both $x_{i}$ and $x_{j}$ are replaced with $\lambda$. There is one
exception: If all $x_{i}$ are nullable, then the production $A \rightarrow \lambda$ is not put into $P$
The argument that this grammar $G$ 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 $\lambda$ not in $L(G)$. Then there exists an equivalent grammar $\widehat{G}$ having no $\lambda$-productions.

Proof. We first find the set $V_{N}$ of all nullable variables of $G$, using the following steps.
(1) For all productions $A \rightarrow \lambda$, put $A$ into $V_{N}$.
(2) Repeat the following step until no further variables are added to $V_{N}$. For all productions

$$
B \rightarrow A_{1} A_{2} \cdots A_{n}
$$ where $A_{1}, A_{2}, \cdots, A_{n}$ 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
$\square$
where each $x_{i} \in V \cup T$. For each such production of $P$, we put into $P$ that production as well as all those generated by replacing nullable variables with $\lambda$ in all possible combinations. For example, if $x_{i}$ and $x_{j}$ are both nullable, there will be one production in $P$ with $x_{i}$ replaced with $\lambda$, one in which $x_{j}$ is replaced with $\lambda$, and one in which both $x_{i}$ and $x_{i}$ are replaced with $\lambda$. There is one exception: If all $x_{i}$ are nullable, then the production $A \rightarrow \lambda$ is not put into $P$

The argument that this grammar $G$ is equivalent to $G$ is straightforward and will be left to the reader.

## Theorem 6.3

Let $G$ be any context-free grammar with $\lambda$ not in $L(G)$. Then there exists an equivalent grammar $\widehat{G}$ having no $\lambda$-productions.
Proof. We first find the set $V_{N}$ of all nullable variables of $G$, using the following steps.
(1) For all productions $A \rightarrow \lambda$, put $A$ into $V_{N}$.
(2) Repeat the following step until no further variables are added to $V_{N}$. For all productions

$$
B \rightarrow A_{1} A_{2} \cdots A_{n}
$$ where $A_{1}, A_{2}, \cdots, A_{n}$ 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

### 6.1 Methods for Transforming Grammars

## Theorem 6.3

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

Proof. We first find the set $V_{N}$ of all nullable variables of $G$, using the following steps.
(1) For all productions $A \rightarrow \lambda$, put $A$ into $V_{N}$.
(2) Repeat the following step until no further variables are added to $V_{N}$. For all productions

$$
B \rightarrow A_{1} A_{2} \cdots A_{n}
$$ where $A_{1}, A_{2}, \cdots, A_{n}$ 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

$$
A \rightarrow x_{1} x_{2} \cdots x_{m}, \quad m \geqslant 1
$$

where each $x_{i} \in V \cup T$. For each such production of 1
production as well as all those generated by replacing nullable variables with in all possible combinations. For example, if $x_{i}$ and $x_{j}$ are both nullable, there will be one production in $P$ with $x_{i}$ replaced with $\lambda$, one in which $x_{j}$ is replaced with $\lambda$, and one in which both $x_{i}$ and $x_{j}$ are replaced with $\lambda$. There is one exception: If all $x_{i}$ are nullable, then the production $A \rightarrow \lambda$ is not put into $P$

The argument that this grammar $G$ is equivalent to $G$ is straightforward and will be left to the reader.

## Theorem 6.3

Let $G$ be any context-free grammar with $\lambda$ not in $L(G)$. Then there exists an equivalent grammar $\widehat{G}$ having no $\lambda$-productions.
Proof. We first find the set $V_{N}$ of all nullable variables of $G$, using the following steps.
(1) For all productions $A \rightarrow \lambda$, put $A$ into $V_{N}$.
(2) Repeat the following step until no further variables are added to $V_{N}$. For all productions

$$
B \rightarrow A_{1} A_{2} \cdots A_{n},
$$ where $A_{1}, A_{2}, \cdots, A_{n}$ 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

$$
A \rightarrow x_{1} x_{2} \cdots x_{m}, \quad m \geqslant 1
$$

where each $x_{i} \in V \cup T$.
production as well as all those generated by replacing nullable variables with 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 $\lambda$, one in which $x_{j}$ is replaced with $\lambda$, and one in which both $x_{i}$ and $x_{j}$ are replaced with $\lambda$. There is one exception: If all $x_{i}$ are nullable, then the production $A \rightarrow \lambda$ is not put into $\widehat{P}$

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

## Theorem 6.3

Let $G$ be any context-free grammar with $\lambda$ not in $L(G)$. Then there exists an equivalent grammar $\widehat{G}$ having no $\lambda$-productions.
Proof. We first find the set $V_{N}$ of all nullable variables of $G$, using the following steps.
(1) For all productions $A \rightarrow \lambda$, put $A$ into $V_{N}$.
(2) Repeat the following step until no further variables are added to $V_{N}$. For all productions

$$
B \rightarrow A_{1} A_{2} \cdots A_{n},
$$ where $A_{1}, A_{2}, \cdots, A_{n}$ 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

$$
A \rightarrow x_{1} x_{2} \cdots x_{m}, \quad m \geqslant 1
$$

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 $\lambda$ in all possible combinations. $\qquad$ will be one production in $P$ with $x_{i}$ replaced with $\lambda$, one in which $x_{j}$ is replaced with $\lambda$, and one in which both $x_{i}$ and $x_{j}$ are replaced with $\lambda$. There is one exception: If all $x_{i}$ are nullable, then the production $A \rightarrow \lambda$ is not put into $\widehat{P}$ The argument that this grammar $G$ is equivalent to $G$ is straightforward and will be left to the reader.

## Theorem 6.3

Let $G$ be any context-free grammar with $\lambda$ not in $L(G)$. Then there exists an equivalent grammar $\widehat{G}$ having no $\lambda$-productions.
Proof. We first find the set $V_{N}$ of all nullable variables of $G$, using the following steps.
(1) For all productions $A \rightarrow \lambda$, put $A$ into $V_{N}$.
(2) Repeat the following step until no further variables are added to $V_{N}$. For all productions

$$
B \rightarrow A_{1} A_{2} \cdots A_{n},
$$ where $A_{1}, A_{2}, \cdots, A_{n}$ 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

$$
A \rightarrow x_{1} x_{2} \cdots x_{m}, \quad m \geqslant 1
$$

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 $\lambda$ in all possible combinations. For example, if $x_{i}$ and $x_{j}$ are both nullable, will be one production in $P$ with $x_{i}$ replaced with $\lambda$, one in which $x_{j}$ is replaced with $\lambda$, and one in which both $x_{i}$ and $x_{j}$ are replaced with $\lambda$. There is one exception: If all $x_{i}$ are nullable, then the production $A \rightarrow \lambda$ is not put into $\widehat{P}$ The argument that this grammar $G$ is equivalent to $G$ is straightforward and will be left to the reader.

## Theorem 6.3

Let $G$ be any context-free grammar with $\lambda$ not in $L(G)$. Then there exists an equivalent grammar $\widehat{G}$ having no $\lambda$-productions.
Proof. We first find the set $V_{N}$ of all nullable variables of $G$, using the following steps.
(1) For all productions $A \rightarrow \lambda$, put $A$ into $V_{N}$.
(2) Repeat the following step until no further variables are added to $V_{N}$.

For all productions

$$
B \rightarrow A_{1} A_{2} \cdots A_{n},
$$ where $A_{1}, A_{2}, \cdots, A_{n}$ 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

$$
A \rightarrow x_{1} x_{2} \cdots x_{m}, \quad m \geqslant 1
$$

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 $\lambda$ 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 $\lambda$,

[^8] will be left to the reader.

## Theorem 6.3

Let $G$ be any context-free grammar with $\lambda$ not in $L(G)$. Then there exists an equivalent grammar $\widehat{G}$ having no $\lambda$-productions.
Proof. We first find the set $V_{N}$ of all nullable variables of $G$, using the following steps.
(1) For all productions $A \rightarrow \lambda$, put $A$ into $V_{N}$.
(2) Repeat the following step until no further variables are added to $V_{N}$.

For all productions

$$
B \rightarrow A_{1} A_{2} \cdots A_{n},
$$ where $A_{1}, A_{2}, \cdots, A_{n}$ 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

$$
A \rightarrow x_{1} x_{2} \cdots x_{m}, \quad m \geqslant 1
$$

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 $\lambda$ 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 $\lambda$, one in which $x_{j}$ is replaced with $\lambda$,
exception: If all $x_{i}$ are nullable, then the production $A \rightarrow \lambda$ is not put into $\widehat{P}$
The argument that this grammar $\widehat{G}$ is equivalent to $G$ is straightforward and will be left to the reader.

## Theorem 6.3

Let $G$ be any context-free grammar with $\lambda$ not in $L(G)$. Then there exists an equivalent grammar $\widehat{G}$ having no $\lambda$-productions.
Proof. We first find the set $V_{N}$ of all nullable variables of $G$, using the following steps.
(1) For all productions $A \rightarrow \lambda$, put $A$ into $V_{N}$.
(2) Repeat the following step until no further variables are added to $V_{N}$.

For all productions

$$
B \rightarrow A_{1} A_{2} \cdots A_{n},
$$ where $A_{1}, A_{2}, \cdots, A_{n}$ 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

$$
A \rightarrow x_{1} x_{2} \cdots x_{m}, \quad m \geqslant 1
$$

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 $\lambda$ 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 $\lambda$, one in which $x_{j}$ is replaced with $\lambda$, and one in which both $x_{i}$ and $x_{j}$ are replaced with $\lambda$.

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 $\lambda$ not in $L(G)$. Then there exists an equivalent grammar $\widehat{G}$ having no $\lambda$-productions.
Proof. We first find the set $V_{N}$ of all nullable variables of $G$, using the following steps.
(1) For all productions $A \rightarrow \lambda$, put $A$ into $V_{N}$.
(2) Repeat the following step until no further variables are added to $V_{N}$.

For all productions

$$
B \rightarrow A_{1} A_{2} \cdots A_{n},
$$ where $A_{1}, A_{2}, \cdots, A_{n}$ 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

$$
A \rightarrow x_{1} x_{2} \cdots x_{m}, \quad m \geqslant 1
$$

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 $\lambda$ 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 $\lambda$, one in which $x_{j}$ is replaced with $\lambda$, and one in which both $x_{i}$ and $x_{j}$ are replaced with $\lambda$. There is one exception:
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 $\lambda$ not in $L(G)$. Then there exists an equivalent grammar $\widehat{G}$ having no $\lambda$-productions.
Proof. We first find the set $V_{N}$ of all nullable variables of $G$, using the following steps.
(1) For all productions $A \rightarrow \lambda$, put $A$ into $V_{N}$.
(2) Repeat the following step until no further variables are added to $V_{N}$.

For all productions

$$
B \rightarrow A_{1} A_{2} \cdots A_{n},
$$

where $A_{1}, A_{2}, \cdots, A_{n}$ 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

$$
A \rightarrow x_{1} x_{2} \cdots x_{m}, \quad m \geqslant 1
$$

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 $\lambda$ 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 $\lambda$, one in which $x_{j}$ is replaced with $\lambda$, and one in which both $x_{i}$ and $x_{j}$ are replaced with $\lambda$. There is one exception: If all $x_{i}$ are nullable, then the production $A \rightarrow \lambda$ is not put into $\widehat{P}$.

The argument that this grammar $G$ is equivalent to $G$ is straightforward and will be left to the reader.

## Theorem 6.3

Let $G$ be any context-free grammar with $\lambda$ not in $L(G)$. Then there exists an equivalent grammar $\widehat{G}$ having no $\lambda$-productions.
Proof. We first find the set $V_{N}$ of all nullable variables of $G$, using the following steps.
(1) For all productions $A \rightarrow \lambda$, put $A$ into $V_{N}$.
(3) Repeat the following step until no further variables are added to $V_{N}$.

For all productions

$$
B \rightarrow A_{1} A_{2} \cdots A_{n},
$$

where $A_{1}, A_{2}, \cdots, A_{n}$ 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

$$
A \rightarrow x_{1} x_{2} \cdots x_{m}, \quad m \geqslant 1,
$$

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 $\lambda$ 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 $\lambda$, one in which $x_{j}$ is replaced with $\lambda$, and one in which both $x_{i}$ and $x_{j}$ are replaced with $\lambda$. There is one exception: If all $x_{i}$ are nullable, then the production $A \rightarrow \lambda$ is not put into $\widehat{P}$. The argument that this grammar $\widehat{G}$ is equivalent to $G$ is straightforward and will be left to the reader.

## Theorem 6.3

Let $G$ be any context-free grammar with $\lambda$ not in $L(G)$. Then there exists an equivalent grammar $\widehat{G}$ having no $\lambda$-productions.
Proof. We first find the set $V_{N}$ of all nullable variables of $G$, using the following steps.
(1) For all productions $A \rightarrow \lambda$, put $A$ into $V_{N}$.
(3) Repeat the following step until no further variables are added to $V_{N}$.

For all productions

$$
B \rightarrow A_{1} A_{2} \cdots A_{n},
$$

where $A_{1}, A_{2}, \cdots, A_{n}$ 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

$$
A \rightarrow x_{1} x_{2} \cdots x_{m}, \quad m \geqslant 1,
$$

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 $\lambda$ 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 $\lambda$, one in which $x_{j}$ is replaced with $\lambda$, and one in which both $x_{i}$ and $x_{j}$ are replaced with $\lambda$. There is one exception: If all $x_{i}$ are nullable, then the production $A \rightarrow \lambda$ is not put into $\widehat{P}$. The argument that this grammar $\widehat{G}$ is equivalent to $G$ is straightforward and will be left to the reader.

## Example 6.5

### 6.1 Methods for Transforming Grammars

## Example 6.5

Find a context-free grammar without $\lambda$-productions equivalent to the grammar defined by

$$
\begin{aligned}
& S \rightarrow A B a C, \\
& A \rightarrow B C, \\
& B \rightarrow b \mid \lambda, \\
& C \rightarrow D \mid \lambda, \\
& D \rightarrow d .
\end{aligned}
$$

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

```
    \(S \rightarrow A B a C|B a C| A a C|A B a| a C|A a| B a \mid a\),
    \(A \rightarrow B C\),
    \(B \rightarrow b \mid \lambda\),
    \(C \rightarrow D \mid \lambda\),
    \(D \rightarrow d\).
```


### 6.1 Methods for Transforming Grammars

## Example 6.5

Find a context-free grammar without $\lambda$-productions equivalent to the grammar defined by

### 6.1 Methods for Transforming Grammars

## Example 6.5

Find a context-free grammar without $\lambda$-productions equivalent to the grammar defined by

$$
\begin{aligned}
& S \rightarrow A B a C, \\
& A \rightarrow B C, \\
& B \rightarrow b \mid \lambda, \\
& C \rightarrow D \mid \lambda, \\
& D \rightarrow d .
\end{aligned}
$$

From the first step of the construction in Theorem 6.3, we find that the nullable variables are $A, B, C$. Then, following the second step of the construction, we get
$S \rightarrow A B a C|B a C| A a C|A B a| a C|A a| B a \mid a$,
$A \rightarrow B C$,
$B \rightarrow b \mid \lambda$.
$C \rightarrow D \mid \lambda$,
$D \rightarrow d$.

## Example 6.5

Find a context-free grammar without $\lambda$-productions equivalent to the grammar defined by

$$
\begin{aligned}
& S \rightarrow A B a C, \\
& A \rightarrow B C, \\
& B \rightarrow b \mid \lambda, \\
& C \rightarrow D \mid \lambda, \\
& D \rightarrow d .
\end{aligned}
$$

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


## Example 6.5

Find a context-free grammar without $\lambda$-productions equivalent to the grammar defined by

$$
\begin{aligned}
& S \rightarrow A B a C, \\
& A \rightarrow B C, \\
& B \rightarrow b \mid \lambda, \\
& C \rightarrow D \mid \lambda, \\
& D \rightarrow d .
\end{aligned}
$$

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

## construction, we get



## Example 6.5

Find a context-free grammar without $\lambda$-productions equivalent to the grammar defined by

$$
\begin{aligned}
& S \rightarrow A B a C, \\
& A \rightarrow B C, \\
& B \rightarrow b \mid \lambda, \\
& C \rightarrow D \mid \lambda, \\
& D \rightarrow d .
\end{aligned}
$$

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


## Example 6.5

Find a context-free grammar without $\lambda$-productions equivalent to the grammar defined by

$$
\begin{aligned}
& S \rightarrow A B a C, \\
& A \rightarrow B C, \\
& B \rightarrow b \mid \lambda, \\
& C \rightarrow D \mid \lambda, \\
& D \rightarrow d .
\end{aligned}
$$

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

$$
\begin{aligned}
& S \rightarrow A B a C|B a C| A a C|A B a| a C|A a| B a \mid a \\
& A \rightarrow B C \\
& B \rightarrow b \mid \lambda \\
& C \rightarrow D \mid \lambda \\
& D \rightarrow d
\end{aligned}
$$

### 6.1 Methods for Transforming Grammars

```
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

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

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

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

### 6.1 Methods for Transforming Grammars

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

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

### 6.1 Methods for Transforming Grammars

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
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

### 6.1 Methods for Transforming Grammars

## 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
$\square$
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

### 6.1 Methods for Transforming Grammars

## 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

$$
A \rightarrow B,
$$

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

### 6.1 Methods for Transforming Grammars

## 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

$$
A \rightarrow B,
$$

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

## 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

$$
A \rightarrow B,
$$

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

## 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

$$
A \rightarrow B,
$$

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 $\lambda$-productions. Then there exists a context-free grammar $G=(V, T, S, P)$ 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

$$
A \rightarrow B,
$$

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 $\lambda$-productions. 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

$$
A \rightarrow B,
$$

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 $\lambda$-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 $G$.

### 6.1 Methods for Transforming Grammars

Proof. Obviously, any unit-production of the form $A \rightarrow A$ can be removed from the grammar without effect, and we need only consider $A \rightarrow B$, where $A$ and $B$ are different variables. At first sight, it may seem that we can use Theorem 6.1 directly with $x_{1}=x_{2}=\lambda$ to replace

$$
A \rightarrow B
$$

with

$$
A \rightarrow y_{1}\left|y_{2}\right| \cdots \mid y_{n} .
$$

But this will not always work; in the special case

$$
\begin{aligned}
& A \rightarrow B \\
& B \rightarrow A
\end{aligned}
$$

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

$$
\begin{equation*}
A \stackrel{*}{\Rightarrow} B \tag{4}
\end{equation*}
$$

We can do this by drawing a dependency graph with an edge $(C, D)$ whenever the grammar has a unit-production $C \rightarrow D$; 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 $\widehat{P}$

$$
A \rightarrow y_{1}\left|y_{2}\right| \cdots \mid y_{n}
$$

where $B \rightarrow y_{1}\left|y_{2}\right| \cdots \mid y_{n}$ is the set of all rules in $\widehat{P}$ with $B$ on the left. Note that since $B \rightarrow y_{1}\left|y_{2}\right| \cdots \mid y_{n}$ is taken from $P$, 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

Proof. Obviously, any unit-production of the form $A \rightarrow A$ can be removed from the grammar without effect, and we need only consider $A \rightarrow B$, where $A$ and $B$ are different variables. At first sight, it may seem that we can use Theorem 6.1 directly with $x_{1}=x_{2}=\lambda$ to replace

$$
A \rightarrow B
$$

with

$$
A \rightarrow y_{1}\left|y_{2}\right| \cdots \mid y_{n} .
$$

But this will not always work; in the special case

$$
\begin{aligned}
& A \rightarrow B, \\
& B \rightarrow A,
\end{aligned}
$$

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

$$
\begin{equation*}
A \stackrel{*}{\Rightarrow} B . \tag{4}
\end{equation*}
$$

We can do this by drawing a dependency graph with an edge ( $C, D$ ) whenever the grammar has a unit-production $C \rightarrow D$; 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 $\widehat{P}$

$$
A \rightarrow y_{1}\left|y_{2}\right| \cdots \mid y_{n},
$$

where $B \rightarrow y_{1}\left|y_{2}\right| \cdots \mid y_{n}$ is the set of all rules in $\widehat{P}$ with $B$ on the left. Note that since $B \rightarrow y_{1}\left|y_{2}\right| \cdots \mid y_{n}$ 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

Proof. Obviously, any unit-production of the form $A \rightarrow A$ can be removed from the grammar without effect, $B$ 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

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

$$
\begin{equation*}
A \stackrel{*}{\Rightarrow} B . \tag{4}
\end{equation*}
$$

We can do this by drawing a dependency graph with an edge $(C, D)$ whenever the grammar has a unit-production $C \rightarrow D$; 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 $\widehat{P}$
where $B \rightarrow y_{1}\left|y_{2}\right| \cdots \mid y_{n}$ is the set of all rules in $\widehat{P}$ with $B$ on the left. Note that since $B \rightarrow y_{1}\left|y_{2}\right| \cdots \mid y_{n}$ is taken from $P$, 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

Proof. Obviously, any unit-production of the form $A \rightarrow A$ can be removed from the grammar without effect, and we need only consider $A \rightarrow B$, $B$ 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

```
with
But this will not always work; in the special case
```

                                    \(A \rightarrow B\),
    the unit-productions are not removed. To get around this, we first find, for each
$A$, all variables $B$ such that
We can do this by drawing a dependency graph with an edge $(C, D)$ whenever
the grammar has a unit-production $C \rightarrow D$; then (4) holds whenever there is a
walk between $A$ and $B$. The new grammar $\widehat{G}$ is generated by first putting into
$P$ all non-unit productions of $P$. Next, for all $A$ and $B$ satisfying (4), we add
to $\widehat{P}$
where $B \rightarrow y_{1}\left|y_{2}\right| \cdots \mid y_{n}$ is the set of all rules in $\widehat{P}$ with $B$ on the left. Note
that since $B \rightarrow y_{1}\left|y_{2}\right| \cdots \mid y_{n}$ is taken from $P$, 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

Proof. Obviously, any unit-production of the form $A \rightarrow A$ can be removed from the grammar without effect, and we need only consider $A \rightarrow B$, where $A$ and $B$ are different variables.

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


### 6.1 Methods for Transforming Grammars

Proof. Obviously, any unit-production of the form $A \rightarrow A$ can be removed from the grammar without effect, and we need only consider $A \rightarrow B$, where $A$ and $B$ are different variables. At first sight, it may seem that we can use Theorem 6.1 directly with $x_{1}=x_{2}=\lambda$ to replace
with
But this will not always work; in the special case
the unit-productions are not removed. To get around this, we first find, for each $A$, all variables $B$ such that


### 6.1 Methods for Transforming Grammars

Proof. Obviously, any unit-production of the form $A \rightarrow A$ can be removed from the grammar without effect, and we need only consider $A \rightarrow B$, where $A$ and $B$ are different variables. At first sight, it may seem that we can use Theorem 6.1 directly with $x_{1}=x_{2}=\lambda$ to replace

$$
A \rightarrow B
$$

## with

But this will not always work; in the special case
the unit-productions are not removed. To get around this, we first find, for each $A$, all variables $B$ such that


### 6.1 Methods for Transforming Grammars

Proof. Obviously, any unit-production of the form $A \rightarrow A$ can be removed from the grammar without effect, and we need only consider $A \rightarrow B$, where $A$ and $B$ are different variables. At first sight, it may seem that we can use Theorem 6.1 directly with $x_{1}=x_{2}=\lambda$ to replace

$$
A \rightarrow B
$$

with
But this will not always work; in the special case
the unit-productions are not removed. To get around this, we first find, for each $A$, all variables $B$ such that

$\square$ $y_{n}$ is the set of all rules in $P$ with $B$ on the left. Note that since $B \rightarrow y_{1}\left|\boldsymbol{y}_{2}\right| \cdots \mid y_{n}$ is taken from $P$, 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

Proof. Obviously, any unit-production of the form $A \rightarrow A$ can be removed from the grammar without effect, and we need only consider $A \rightarrow B$, where $A$ and $B$ are different variables. At first sight, it may seem that we can use Theorem 6.1 directly with $x_{1}=x_{2}=\lambda$ to replace

$$
A \rightarrow B
$$

with

$$
A \rightarrow y_{1}\left|y_{2}\right| \cdots \mid y_{n}
$$

## But this will not always work; in the special case

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


### 6.1 Methods for Transforming Grammars

Proof. Obviously, any unit-production of the form $A \rightarrow A$ can be removed from the grammar without effect, and we need only consider $A \rightarrow B$, where $A$ and $B$ are different variables. At first sight, it may seem that we can use Theorem 6.1 directly with $x_{1}=x_{2}=\lambda$ to replace

$$
A \rightarrow B
$$

with

$$
A \rightarrow y_{1}\left|y_{2}\right| \cdots \mid y_{n}
$$

But this will not always work; in the special case
the unit-productions are not removed. To get around this, we first find, for each $A$, all variables $B$ such that

$\square$ $y_{n}$ is the set of all rules in $P$ with $B$ on the left. Note that since $B \rightarrow y_{1}\left|y_{2}\right| \cdots \mid y_{n}$ is taken from $P$, 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

Proof. Obviously, any unit-production of the form $A \rightarrow A$ can be removed from the grammar without effect, and we need only consider $A \rightarrow B$, where $A$ and $B$ are different variables. At first sight, it may seem that we can use Theorem 6.1 directly with $x_{1}=x_{2}=\lambda$ to replace

$$
A \rightarrow B
$$

with

$$
A \rightarrow y_{1}\left|y_{2}\right| \cdots \mid y_{n} .
$$

But this will not always work; in the special case
the unit-productions are not removed. To get around this, we first find, for each $A$, all variables $B$ such that

$\square$ $y_{n}$ is the set of all rules in $P$ with $B$ on the left. Note that since $B \rightarrow y_{1}\left|y_{2}\right| \cdots \mid y_{n}$ is taken from $P$, 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

Proof. Obviously, any unit-production of the form $A \rightarrow A$ can be removed from the grammar without effect, and we need only consider $A \rightarrow B$, where $A$ and $B$ are different variables. At first sight, it may seem that we can use Theorem 6.1 directly with $x_{1}=x_{2}=\lambda$ to replace

$$
A \rightarrow B
$$

with

$$
A \rightarrow y_{1}\left|y_{2}\right| \cdots \mid y_{n} .
$$

But this will not always work; in the special case

$$
\begin{aligned}
& A \rightarrow B, \\
& B \rightarrow A,
\end{aligned}
$$

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

$\square$ that since $B \rightarrow y_{1}\left|y_{2}\right| \cdots \mid y_{n}$ is taken from $P$. 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

Proof. Obviously, any unit-production of the form $A \rightarrow A$ can be removed from the grammar without effect, and we need only consider $A \rightarrow B$, where $A$ and $B$ are different variables. At first sight, it may seem that we can use Theorem 6.1 directly with $x_{1}=x_{2}=\lambda$ to replace

$$
A \rightarrow B
$$

with

$$
A \rightarrow y_{1}\left|y_{2}\right| \cdots \mid y_{n} .
$$

But this will not always work; in the special case

$$
\begin{aligned}
& A \rightarrow B \\
& B \rightarrow A
\end{aligned}
$$

the unit-productions are not removed.

$\square$ that since $B \rightarrow y_{1}\left|y_{2}\right| \cdots \mid y_{n}$ is taken from $P$ 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

Proof. Obviously, any unit-production of the form $A \rightarrow A$ can be removed from the grammar without effect, and we need only consider $A \rightarrow B$, where $A$ and $B$ are different variables. At first sight, it may seem that we can use Theorem 6.1 directly with $x_{1}=x_{2}=\lambda$ to replace

$$
A \rightarrow B
$$

with

$$
A \rightarrow y_{1}\left|y_{2}\right| \cdots \mid y_{n} .
$$

But this will not always work; in the special case

$$
\begin{aligned}
& A \rightarrow B \\
& B \rightarrow A
\end{aligned}
$$

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

> We can do this by drawing a dependency graph with an edge $(C, D)$ whenever the grammar has a unit-production $C \rightarrow D$; 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 $\widehat{P}$
$\qquad$ that since $B \rightarrow y_{1}\left|y_{2}\right| \cdots \mid y_{n}$ is taken from $P$, 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

Proof. Obviously, any unit-production of the form $A \rightarrow A$ can be removed from the grammar without effect, and we need only consider $A \rightarrow B$, where $A$ and $B$ are different variables. At first sight, it may seem that we can use Theorem 6.1 directly with $x_{1}=x_{2}=\lambda$ to replace

$$
A \rightarrow B
$$

with

$$
A \rightarrow y_{1}\left|y_{2}\right| \cdots \mid y_{n} .
$$

But this will not always work; in the special case

$$
\begin{aligned}
& A \rightarrow B \\
& B \rightarrow A
\end{aligned}
$$

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

$$
\begin{equation*}
A \stackrel{*}{\Rightarrow} B . \tag{4}
\end{equation*}
$$

We can do this by drawing a dependency graph with an edge $(C, D)$ whenever the grammar has a unit-production $C \rightarrow D$; 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 $\widehat{P}$
$\qquad$ that since $B \rightarrow y_{1}\left|y_{2}\right| \cdots \mid y_{n}$ is taken from $P$, 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

Proof. Obviously, any unit-production of the form $A \rightarrow A$ can be removed from the grammar without effect, and we need only consider $A \rightarrow B$, where $A$ and $B$ are different variables. At first sight, it may seem that we can use Theorem 6.1 directly with $x_{1}=x_{2}=\lambda$ to replace

$$
A \rightarrow B
$$

with

$$
A \rightarrow y_{1}\left|y_{2}\right| \cdots \mid y_{n} .
$$

But this will not always work; in the special case

$$
\begin{aligned}
& A \rightarrow B, \\
& B \rightarrow A,
\end{aligned}
$$

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

$$
\begin{equation*}
A \stackrel{*}{\Rightarrow} B . \tag{4}
\end{equation*}
$$

We can do this by drawing a dependency graph with an edge ( $C, D$ ) whenever the grammar has a unit-production $C \rightarrow D$;

$\square$ where $B \rightarrow y_{1} y_{2}$ $y_{n}$ is the set of that since $B \rightarrow u_{1}\left|y_{2}\right|$ variable, so that no unit-productions are created by the last step

### 6.1 Methods for Transforming Grammars

Proof. Obviously, any unit-production of the form $A \rightarrow A$ can be removed from the grammar without effect, and we need only consider $A \rightarrow B$, where $A$ and $B$ are different variables. At first sight, it may seem that we can use Theorem 6.1 directly with $x_{1}=x_{2}=\lambda$ to replace

$$
A \rightarrow B
$$

with

$$
A \rightarrow y_{1}\left|y_{2}\right| \cdots \mid y_{n} .
$$

But this will not always work; in the special case

$$
\begin{aligned}
& A \rightarrow B \\
& B \rightarrow A
\end{aligned}
$$

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

$$
\begin{equation*}
A \stackrel{*}{\Rightarrow} B . \tag{4}
\end{equation*}
$$

We can do this by drawing a dependency graph with an edge ( $C, D$ ) whenever the grammar has a unit-production $C \rightarrow D$; then (4) holds whenever there is a walk between $A$ and $B$.
$P$ all non-unit productions of $P$. Next, for all $A$ and $B$ satisfying (4), we add to $\widehat{P}$

[^9]$y_{n}$ is the set of all rules i
that since $B \rightarrow u_{n}\left|\varkappa_{n}\right|$
variable, so that no unit-productions are created by the last step.

### 6.1 Methods for Transforming Grammars

Proof. Obviously, any unit-production of the form $A \rightarrow A$ can be removed from the grammar without effect, and we need only consider $A \rightarrow B$, where $A$ and $B$ are different variables. At first sight, it may seem that we can use Theorem 6.1 directly with $x_{1}=x_{2}=\lambda$ to replace

$$
A \rightarrow B
$$

with

$$
A \rightarrow y_{1}\left|y_{2}\right| \cdots \mid y_{n} .
$$

But this will not always work; in the special case

$$
\begin{aligned}
& A \rightarrow B \\
& B \rightarrow A
\end{aligned}
$$

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

$$
\begin{equation*}
A \stackrel{*}{\Rightarrow} B . \tag{4}
\end{equation*}
$$

We can do this by drawing a dependency graph with an edge ( $C, D$ ) whenever the grammar has a unit-production $C \rightarrow D$; 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 $P$.

### 6.1 Methods for Transforming Grammars

Proof. Obviously, any unit-production of the form $A \rightarrow A$ can be removed from the grammar without effect, and we need only consider $A \rightarrow B$, where $A$ and $B$ are different variables. At first sight, it may seem that we can use Theorem 6.1 directly with $x_{1}=x_{2}=\lambda$ to replace

$$
A \rightarrow B
$$

with

$$
A \rightarrow y_{1}\left|y_{2}\right| \cdots \mid y_{n} .
$$

But this will not always work; in the special case

$$
\begin{aligned}
& A \rightarrow B \\
& B \rightarrow A
\end{aligned}
$$

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

$$
\begin{equation*}
A \stackrel{*}{\Rightarrow} B . \tag{4}
\end{equation*}
$$

We can do this by drawing a dependency graph with an edge ( $C, D$ ) whenever the grammar has a unit-production $C \rightarrow D$; 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 $\widehat{P}$

## where $B \rightarrow y_{1}\left|y_{2}\right| \cdots \mid y_{n}$ is the set of all rules in $P$ with $B$ on the left. Note

 that since $B \rightarrow y_{1}\left|y_{2}\right| \cdots \mid y_{n}$ 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

Proof. Obviously, any unit-production of the form $A \rightarrow A$ can be removed from the grammar without effect, and we need only consider $A \rightarrow B$, where $A$ and $B$ are different variables. At first sight, it may seem that we can use Theorem 6.1 directly with $x_{1}=x_{2}=\lambda$ to replace

$$
A \rightarrow B
$$

with

$$
A \rightarrow y_{1}\left|y_{2}\right| \cdots \mid y_{n} .
$$

But this will not always work; in the special case

$$
\begin{aligned}
& A \rightarrow B \\
& B \rightarrow A
\end{aligned}
$$

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

$$
\begin{equation*}
A \stackrel{*}{\Rightarrow} B . \tag{4}
\end{equation*}
$$

We can do this by drawing a dependency graph with an edge ( $C, D$ ) whenever the grammar has a unit-production $C \rightarrow D$; 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 $\widehat{P}$

$$
A \rightarrow y_{1}\left|y_{2}\right| \cdots \mid y_{n}
$$

where
variable, so that no unit-productions are created by the last step

### 6.1 Methods for Transforming Grammars

Proof. Obviously, any unit-production of the form $A \rightarrow A$ can be removed from the grammar without effect, and we need only consider $A \rightarrow B$, where $A$ and $B$ are different variables. At first sight, it may seem that we can use Theorem 6.1 directly with $x_{1}=x_{2}=\lambda$ to replace

$$
A \rightarrow B
$$

with

$$
A \rightarrow y_{1}\left|y_{2}\right| \cdots \mid y_{n} .
$$

But this will not always work; in the special case

$$
\begin{aligned}
& A \rightarrow B \\
& B \rightarrow A
\end{aligned}
$$

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

$$
\begin{equation*}
A \stackrel{*}{\Rightarrow} B . \tag{4}
\end{equation*}
$$

We can do this by drawing a dependency graph with an edge ( $C, D$ ) whenever the grammar has a unit-production $C \rightarrow D$; 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 $\widehat{P}$

$$
A \rightarrow y_{1}\left|y_{2}\right| \cdots \mid y_{n}
$$

where $B \rightarrow y_{1}\left|y_{2}\right| \cdots \mid y_{n}$ is the set of all rules in $\widehat{P}$ with $B$ on the left.
variable, so that no unit-productions are created by the last step.

### 6.1 Methods for Transforming Grammars

Proof. Obviously, any unit-production of the form $A \rightarrow A$ can be removed from the grammar without effect, and we need only consider $A \rightarrow B$, where $A$ and $B$ are different variables. At first sight, it may seem that we can use Theorem 6.1 directly with $x_{1}=x_{2}=\lambda$ to replace

$$
A \rightarrow B
$$

with

$$
A \rightarrow y_{1}\left|y_{2}\right| \cdots \mid y_{n} .
$$

But this will not always work; in the special case

$$
\begin{aligned}
& A \rightarrow B \\
& B \rightarrow A
\end{aligned}
$$

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

$$
\begin{equation*}
A \stackrel{*}{\Rightarrow} B . \tag{4}
\end{equation*}
$$

We can do this by drawing a dependency graph with an edge $(C, D)$ whenever the grammar has a unit-production $C \rightarrow D$; 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 $\widehat{P}$

$$
A \rightarrow y_{1}\left|y_{2}\right| \cdots \mid y_{n}
$$

where $B \rightarrow y_{1}\left|y_{2}\right| \cdots \mid y_{n}$ is the set of all rules in $\widehat{P}$ with $B$ on the left. Note that since $B \rightarrow y_{1}\left|y_{2}\right| \cdots \mid y_{n}$ is taken from $\widehat{P}$,

### 6.1 Methods for Transforming Grammars

Proof. Obviously, any unit-production of the form $A \rightarrow A$ can be removed from the grammar without effect, and we need only consider $A \rightarrow B$, where $A$ and $B$ are different variables. At first sight, it may seem that we can use Theorem 6.1 directly with $x_{1}=x_{2}=\lambda$ to replace

$$
A \rightarrow B
$$

with

$$
A \rightarrow y_{1}\left|y_{2}\right| \cdots \mid y_{n} .
$$

But this will not always work; in the special case

$$
\begin{aligned}
& A \rightarrow B \\
& B \rightarrow A
\end{aligned}
$$

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

$$
\begin{equation*}
A \stackrel{*}{\Rightarrow} B . \tag{4}
\end{equation*}
$$

We can do this by drawing a dependency graph with an edge $(C, D)$ whenever the grammar has a unit-production $C \rightarrow D$; 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 $\widehat{P}$

$$
A \rightarrow y_{1}\left|y_{2}\right| \cdots \mid y_{n}
$$

where $B \rightarrow y_{1}\left|y_{2}\right| \cdots \mid y_{n}$ is the set of all rules in $\widehat{P}$ with $B$ on the left. Note that since $B \rightarrow y_{1}\left|y_{2}\right| \cdots \mid y_{n}$ is taken from $\widehat{P}$, none of the $y_{i}$ can be a single variable,

### 6.1 Methods for Transforming Grammars

Proof. Obviously, any unit-production of the form $A \rightarrow A$ can be removed from the grammar without effect, and we need only consider $A \rightarrow B$, where $A$ and $B$ are different variables. At first sight, it may seem that we can use Theorem 6.1 directly with $x_{1}=x_{2}=\lambda$ to replace

$$
A \rightarrow B
$$

with

$$
A \rightarrow y_{1}\left|y_{2}\right| \cdots \mid y_{n}
$$

But this will not always work; in the special case

$$
\begin{aligned}
& A \rightarrow B \\
& B \rightarrow A
\end{aligned}
$$

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

$$
\begin{equation*}
A \stackrel{*}{\Rightarrow} B . \tag{4}
\end{equation*}
$$

We can do this by drawing a dependency graph with an edge $(C, D)$ whenever the grammar has a unit-production $C \rightarrow D$; 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 $\widehat{P}$

$$
A \rightarrow y_{1}\left|y_{2}\right| \cdots \mid y_{n}
$$

where $B \rightarrow y_{1}\left|y_{2}\right| \cdots \mid y_{n}$ is the set of all rules in $\widehat{P}$ with $B$ on the left. Note that since $B \rightarrow y_{1}\left|y_{2}\right| \cdots \mid y_{n}$ 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
```


### 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
```


### 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

### 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

### 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



We see from it that $S \Rightarrow A, S \stackrel{*}{\Rightarrow} B, B \Rightarrow A$, and $A \Rightarrow B$. Hence, we add to the original non-unit productions


### 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


The dependency graph for the unit-productions is given in the Figure.


We see from it that $S \Rightarrow A, S \stackrel{*}{\Rightarrow} B, B \Rightarrow A$, and $A \Rightarrow B$. Hence, we add to the original non-unit productions


### 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

$$
\begin{aligned}
& S \rightarrow A a \mid B, \\
& B \rightarrow A \mid b b, \\
& A \rightarrow a|b c| B .
\end{aligned}
$$

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

$B \rightarrow b b$,

### 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

$$
\begin{aligned}
S & \rightarrow A a \mid B \\
B & \rightarrow A \mid b b \\
A & \rightarrow a|b c| B .
\end{aligned}
$$

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

### 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

$$
\begin{aligned}
& S \rightarrow A a \mid B \\
& B \rightarrow A \mid b b, \\
& A \rightarrow a|b c| B .
\end{aligned}
$$

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

### 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

$$
\begin{aligned}
& S \rightarrow A a \mid B, \\
& B \rightarrow A \mid b b, \\
& A \rightarrow a|b c| B .
\end{aligned}
$$

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

### 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

$$
\begin{aligned}
S & \rightarrow A a \mid B, \\
B & \rightarrow A \mid b b, \\
A & \rightarrow a|b c| B .
\end{aligned}
$$

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

### 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

$$
\begin{aligned}
S & \rightarrow A a \mid B \\
B & \rightarrow A \mid b b \\
A & \rightarrow a|b c| B .
\end{aligned}
$$

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

$$
\begin{aligned}
& S \rightarrow A a \\
& A \rightarrow a \mid b c \\
& B \rightarrow b b
\end{aligned}
$$

## Example 6.6 (continuation)

We can put all these results together to show that grammars for context-free languages can be made free of useless productions, $\lambda$-productions, and unit-productions.

## Example 6.6 (continuation)

the new rules

$$
\begin{aligned}
& S \rightarrow a|b c| b b, \\
& A \rightarrow b b, \\
& B \rightarrow a \mid b c,
\end{aligned}
$$

to obtain the equivalent grammar

$$
\begin{aligned}
& S \rightarrow a|b c| b b \mid A a, \\
& A \rightarrow a|b b| b c, \\
& B \rightarrow a|b b| b c .
\end{aligned}
$$

> 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, $\lambda$-productions, and unit-productions.

## Example 6.6 (continuation)

the new rules

$$
\begin{aligned}
& S \rightarrow a|b c| b b, \\
& A \rightarrow b b, \\
& B \rightarrow a \mid b c,
\end{aligned}
$$

to obtain the equivalent grammar

$$
\begin{aligned}
& S \rightarrow a|b c| b b \mid A a, \\
& A \rightarrow a|b b| b c, \\
& B \rightarrow a|b b| b c .
\end{aligned}
$$

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, $\lambda$-productions, and unit-productions.

## Example 6.6 (continuation)

the new rules

$$
\begin{aligned}
& S \rightarrow a|b c| b b, \\
& A \rightarrow b b \\
& B \rightarrow a \mid b c
\end{aligned}
$$

to obtain the equivalent grammar

$$
\begin{aligned}
& S \rightarrow a|b c| b b \mid A a \\
& A \rightarrow a|b b| b c \\
& B \rightarrow a|b b| b c
\end{aligned}
$$

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, $\lambda$-productions, and unit-productions.

## Example 6.6 (continuation)

the new rules

$$
\begin{aligned}
& S \rightarrow a|b c| b b, \\
& A \rightarrow b b \\
& B \rightarrow a \mid b c
\end{aligned}
$$

to obtain the equivalent grammar
$S \rightarrow a|b c| b b \mid A a$,
$A \rightarrow a|b b| b c$,
$B \rightarrow a|b b| b c$.
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, $\lambda$-productions, and unit-productions.

## Example 6.6 (continuation)

the new rules

$$
\begin{aligned}
& S \rightarrow a|b c| b b, \\
& A \rightarrow b b \\
& B \rightarrow a \mid b c
\end{aligned}
$$

to obtain the equivalent grammar

$$
\begin{aligned}
& S \rightarrow a|b c| b b \mid A a \\
& A \rightarrow a|b b| b c \\
& B \rightarrow a|b b| b c
\end{aligned}
$$

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, $\lambda$-productions, and unit-productions.

## Example 6.6 (continuation)

the new rules

$$
\begin{aligned}
& S \rightarrow a|b c| b b, \\
& A \rightarrow b b, \\
& B \rightarrow a \mid b c,
\end{aligned}
$$

to obtain the equivalent grammar

$$
\begin{aligned}
& S \rightarrow a|b c| b b \mid A a, \\
& A \rightarrow a|b b| b c, \\
& B \rightarrow a|b b| b c .
\end{aligned}
$$

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, $\lambda$-productions, and unit-productions.

## Example 6.6 (continuation)

the new rules

$$
\begin{aligned}
& S \rightarrow a|b c| b b, \\
& A \rightarrow b b, \\
& B \rightarrow a \mid b c,
\end{aligned}
$$

to obtain the equivalent grammar

$$
\begin{aligned}
& S \rightarrow a|b c| b b \mid A a, \\
& A \rightarrow a|b b| b c \\
& B \rightarrow a|b b| b c .
\end{aligned}
$$

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, $\lambda$-productions, and unit-productions.

### 6.1 Methods for Transforming Grammars

## Theorem 6.5

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 $\lambda$-productions can create new unit-productions. Also, Theorem 6.4 requires that the grammar have no $\lambda$-productions. But note that the removal of unit-productions does not create $\lambda$-productions, and the removal of useless productions does not create $\lambda$-productions or unit-productions. Therefore, we can remove all undesirable productions using the following sequence of steps:

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 $\lambda$. Then there exists a context-free grammar that generates $L$ and that does not have any useless productions, $\lambda$-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 $\lambda$-productions can create new unit-productions. Also, Theorem 6.4 requires that the grammar have no $\lambda$-productions. But note that the removal of unit-productions does not create $\lambda$-productions, and the removal of useless productions does not create $\lambda$-productions or unit-productions. Therefore, we can remove all undesirable productions using the following sequence of steps:

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 $\lambda$. Then there exists a context-free grammar that generates $L$ and that does not have any useless productions, $\lambda$-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 $\lambda$-productions can create new unit-productions. Also, Theorem 6.4 requires that the grammar have no $\lambda$-productions. But note that the removal of unit-productions does not create $\lambda$-productions, and the removal of useless productions does not create $\lambda$-productions or unit-productions. Therefore, we can remove all undesirable productions using the following sequence of steps:

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 $\lambda$. Then there exists a context-free grammar that generates $L$ and that does not have any useless productions, $\lambda$-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 $\lambda$-productions can create new unit-productions. Also, Theorem 6.4 requires that the grammar have no $\lambda$-productions. But note that the removal of unit-productions does not create $\lambda$-productions, and the removal of useless productions does not create $\lambda$-productions or unit-productions. Therefore, we can remove all undesirable productions using the following sequence of steps:

## Theorem 6.5

Let $L$ be a context-free language that does not contain $\lambda$. Then there exists a context-free grammar that generates $L$ and that does not have any useless productions, $\lambda$-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 $\lambda$-productions can create new unit-productions. Also, Theorem 6.4 requires that the grammar have no $\lambda$-productions. But note that the removal of unit-productions does not create $\lambda$-productions, and the removal of useless productions does not create $\lambda$-productions or unit-productions. Therefore, we can remove all undesirable productions using the following sequence of steps:

[^10]
## Theorem 6.5

Let $L$ be a context-free language that does not contain $\lambda$. Then there exists a context-free grammar that generates $L$ and that does not have any useless productions, $\lambda$-productions, or unit-productions.

Proof. The procedures given in Theorems 6.2, 6.3, and 6.4 remove these kinds of productions in turn.
removal of one type of production may introduce productions of another type for example, the procedure for removing $\lambda$-productions can create new unit-productions. Also, Theorem 6.4 requires that the grammar have no $\lambda$-productions. But note that the removal of unit-productions does not create $\lambda$-productions, and the removal of useless productions does not create $\lambda$-productions or unit-productions. Therefore, we can remove all undesirable productions using the following sequence of steps:

## Theorem 6.5

Let $L$ be a context-free language that does not contain $\lambda$. Then there exists a context-free grammar that generates $L$ and that does not have any useless productions, $\lambda$-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 $\lambda$-productions can create new unit-productions. Also, Theorem 6.4 requires that the grammar have no $\lambda$-productions. But note that the removal of unit-productions does not create $\lambda$-productions, and the removal of useless productions does not create $\lambda$-productions or unit-productions. Therefore, we can remove all undesirable productions using the following sequence of steps:

## Theorem 6.5

Let $L$ be a context-free language that does not contain $\lambda$. Then there exists a context-free grammar that generates $L$ and that does not have any useless productions, $\lambda$-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 $\lambda$-productions can create new unit-productions.
> $\lambda$-productions. But note that the removal of unit-productions does not create $\lambda$-productions, and the removal of useless productions does not create $\lambda$-productions or unit-productions. Therefore, we can remove all undesirable productions using the following sequence of steps:

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 $\lambda$. Then there exists a context-free grammar that generates $L$ and that does not have any useless productions, $\lambda$-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 $\lambda$-productions can create new unit-productions. Also, Theorem 6.4 requires that the grammar have no $\lambda$-productions.

[^11]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 $\lambda$. Then there exists a context-free grammar that generates $L$ and that does not have any useless productions, $\lambda$-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 $\lambda$-productions can create new unit-productions. Also, Theorem 6.4 requires that the grammar have no $\lambda$-productions. But note that the removal of unit-productions does not create $\lambda$-productions,
$\lambda$-productions or unit-productions. Therefore, we can remove all undesirable productions using the following sequence of steps:

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 $\lambda$. Then there exists a context-free grammar that generates $L$ and that does not have any useless productions, $\lambda$-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 $\lambda$-productions can create new unit-productions. Also, Theorem 6.4 requires that the grammar have no $\lambda$-productions. But note that the removal of unit-productions does not create $\lambda$-productions, and the removal of useless productions does not create $\lambda$-productions or unit-productions.
productions using the following sequence of steps:

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 $\lambda$. Then there exists a context-free grammar that generates $L$ and that does not have any useless productions, $\lambda$-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 $\lambda$-productions can create new unit-productions. Also, Theorem 6.4 requires that the grammar have no $\lambda$-productions. But note that the removal of unit-productions does not create $\lambda$-productions, and the removal of useless productions does not create $\lambda$-productions or unit-productions. Therefore, we can remove all undesirable productions using the following sequence of steps:

[^12]
## Theorem 6.5

Let $L$ be a context-free language that does not contain $\lambda$. Then there exists a context-free grammar that generates $L$ and that does not have any useless productions, $\lambda$-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 $\lambda$-productions can create new unit-productions. Also, Theorem 6.4 requires that the grammar have no $\lambda$-productions. But note that the removal of unit-productions does not create $\lambda$-productions, and the removal of useless productions does not create $\lambda$-productions or unit-productions. Therefore, we can remove all undesirable productions using the following sequence of steps:

[^13]
## Theorem 6.5

Let $L$ be a context-free language that does not contain $\lambda$. Then there exists a context-free grammar that generates $L$ and that does not have any useless productions, $\lambda$-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 $\lambda$-productions can create new unit-productions. Also, Theorem 6.4 requires that the grammar have no $\lambda$-productions. But note that the removal of unit-productions does not create $\lambda$-productions, and the removal of useless productions does not create $\lambda$-productions or unit-productions. Therefore, we can remove all undesirable productions using the following sequence of steps:
(1) Remove $\lambda$-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 $\lambda$. Then there exists a context-free grammar that generates $L$ and that does not have any useless productions, $\lambda$-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 $\lambda$-productions can create new unit-productions. Also, Theorem 6.4 requires that the grammar have no $\lambda$-productions. But note that the removal of unit-productions does not create $\lambda$-productions, and the removal of useless productions does not create $\lambda$-productions or unit-productions. Therefore, we can remove all undesirable productions using the following sequence of steps:
(1) Remove $\lambda$-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 $\lambda$. Then there exists a context-free grammar that generates $L$ and that does not have any useless productions, $\lambda$-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 $\lambda$-productions can create new unit-productions. Also, Theorem 6.4 requires that the grammar have no $\lambda$-productions. But note that the removal of unit-productions does not create $\lambda$-productions, and the removal of useless productions does not create $\lambda$-productions or unit-productions. Therefore, we can remove all undesirable productions using the following sequence of steps:
(1) Remove $\lambda$-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 $\lambda$. Then there exists a context-free grammar that generates $L$ and that does not have any useless productions, $\lambda$-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 $\lambda$-productions can create new unit-productions. Also, Theorem 6.4 requires that the grammar have no $\lambda$-productions. But note that the removal of unit-productions does not create $\lambda$-productions, and the removal of useless productions does not create $\lambda$-productions or unit-productions. Therefore, we can remove all undesirable productions using the following sequence of steps:
(1) Remove $\lambda$-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!


[^0]:    Therefore

[^1]:    Example 6.1

[^2]:    grammar

[^3]:    Definition 6.1

[^4]:    Definition 6.1

[^5]:    is possible. The steps in the algorithm are

[^6]:    A grammar may generate a language not containing $\lambda$, yet have some $\lambda$-productions or nullable variables. In such cases, the $\lambda$-productions can be removed.

[^7]:    A grammar may generate a language not containing $\lambda$, yet have some $\lambda$-productions or nullable variables. In such cases, the $\lambda$-productions can be removed.

[^8]:    with $\lambda$, and one in which both $x_{i}$ and $x_{j}$ are replaced with $\lambda$. There is one exception: If all $x_{i}$ are nullable, then the production $A \rightarrow \lambda$ is not put into $P$

    The argument that this grammar $\widehat{G}$ is equivalent to $G$ is straightformard and

[^9]:    where $B \rightarrow y_{1}\left|y_{2}\right|$

[^10]:    The result will then have none of these productions, and the theorem is proved.

[^11]:    $\lambda$-productions, and the removal of useless productions does not create $\lambda$-productions or unit-productions. Therefore, we can remove all undesirable productions using the following sequence of stens:

[^12]:    The result will then have none of these productions, and the theorem is proved.

[^13]:    (1) Remove $\lambda$-productions.
    (2) Remove unit-productions.

    - Remove useless productions.

    The result will then have none of these productions, and the theorem is proved.

