## Formal Languages, Automata and

 Codes
## Oleg Gutik



## Lecture 2

Three fundamental ideas are the major themes of this course: languages,
grammars, and automata. In the course we shall explore many results about these concepts and about their relationship to each other. First, we must understand the meaning of the terms.

Three fundamental ideas are the major themes of this course: languages, grammars, and automata. In the course we shall explore many results about these concepts and about their relationship to each other. First, we must understand the meaning of the terms.

Three fundamental ideas are the major themes of this course: languages, grammars, and automata. In the course we shall explore many results about these concepts and about their relationship to each other. First, we must understand the meaning of the terms.

Three fundamental ideas are the major themes of this course: languages, grammars, and automata.
these concepts and about their relationship to each other. First, we must understand the meaning of the terms.

Three fundamental ideas are the major themes of this course: languages, grammars, and automata. In the course we shall explore many results about these concepts and about their relationship to each other.
understand the meaning of the terms.

Three fundamental ideas are the major themes of this course: languages, grammars, and automata. In the course we shall explore many results about these concepts and about their relationship to each other. First, we must understand the meaning of the terms.

## 2 THREE BASIC CONCEPTS: Languages

We are all familiar with the notion of natural languages, such as English, Ukrainian, Polish, German, and French. Still, most of us would probably find it difficult to say exactly what the word "language" means. Dictionaries define the term informally as a system suitable for the expression of certain ideas, facts, or concepts, including a set of symbols and rules for their manipulation. While this gives us an intuitive idea of what a language is, it is not sufficient as a definition for the study of formal languages. We need a precise definition for the term.

We start with a finite, nonempty set $\Sigma$ of symbols, called an alphabet. From the individual symbols we construct strings (or words), which are finite sequences of symbols from the alphabet. For example, if the alphabet $\Sigma=\{a, b\}$, then $a b a b$ and $a a a b b b a$ are strings on $\Sigma$. With few exceptions, we will use lowercase letters $a, b, c, \ldots$ for elements of $\Sigma$ and $u, v, w, \ldots$ for string names. We shall write, for example,

$$
w=a b a a a
$$

to indicate that the string named $w$ has the specific value $a b a a a$.

## 2 THREE BASIC CONCEPTS: Languages

We are all familiar with the notion of natural languages, such as English, Ukrainian, Polish, German, and French.
difficult to say exactly what the word "language" means. Dictionaries define the term informally as a system suitable for the expression of certain ideas, facts, or concepts, including a set of symbols and rules for their manipulation. While this gives us an intuitive idea of what a language is, it is not sufficient as a definition for the study of formal languages. We need a precise definition for the term
We start with a finite, nonempty set $\Sigma$ of symbols, called an alphabet. From
the individual symbols we construct strings (or words), which are finite
sequences of symbols from the alphabet. For example, if the alphabet
$\Sigma=\{a, b\}$, then $a b a b$ and $a a a b b b a$ are strings on $\Sigma$. With few exceptions, we will use lowercase letters $a, b, c, \ldots$ for elements of $\Sigma$ and $u, v, w, \ldots$ for string names. We shall write, for example,
$w=a b a a a$
to indicate that the string named $w$ has the specific value abaaa.

## 2 THREE BASIC CONCEPTS: Languages

We are all familiar with the notion of natural languages, such as English, Ukrainian, Polish, German, and French. Still, most of us would probably find it difficult to say exactly what the word "language" means.

```
term informally as a system suitable for the expression of certain ideas, facts, or
concepts, including a set of symbols and rules for their manipulation. While this
gives us an intuitive idea of what a language is, it is not sufficient as a definition
for the study of formal languages. We need a precise definition for the term
We start with a finite, nonempty set }\Sigma\mathrm{ of symbols, called an alphabet. From
the individual symbols we construct strings (or words), which are finite
sequences of symbols from the alphabet. For example, if the alphabet
\Sigma={a,b}, then abab and aaabbba are strings on }\Sigma\mathrm{ . With few exceptions, we
will use lowercase letters }a,b,c,\ldots\mathrm{ for elements of }\Sigma\mathrm{ and }u,v,w,\ldots\mathrm{ for string
names. We shall write, for example,
```

$w=a b a a a$
to indicate that the string named $w$ has the specific value abaaa.

## 2 THREE BASIC CONCEPTS: Languages

We are all familiar with the notion of natural languages, such as English, Ukrainian, Polish, German, and French. Still, most of us would probably find it difficult to say exactly what the word "language" means. Dictionaries define the term informally as a system suitable for the expression of certain ideas, facts, or concepts,
gives us an intuitive idea of what a language is, it is not sufficient as a definition for the study of formal languages. We need a precise definition for the term We start with a finite, nonempty set $\Sigma$ of symbols, called an alphabet. From the individual symbols we construct strings (or words), which are finite sequences of symbols from the alphabet. For example, if the alphabet $\Sigma=\{a, b\}$, then $a b a b$ and $a a a b b b a$ are strings on $\Sigma$. With few exceptions, we will use lowercase letters $a, b, c, \ldots$ for elements of $\Sigma$ and $u, v, w, \ldots$ for string names. We shall write, for example,
$w=a b a a a$
to indicate that the string named $w$ has the specific value abaaa.

## 2 THREE BASIC CONCEPTS: Languages

We are all familiar with the notion of natural languages, such as English, Ukrainian, Polish, German, and French. Still, most of us would probably find it difficult to say exactly what the word "language" means. Dictionaries define the term informally as a system suitable for the expression of certain ideas, facts, or concepts, including a set of symbols and rules for their manipulation.
gives us an intuitive idea of what a language is, it is not sufficient as a definition
for the study of formal languages. We need a precise definition for the term
We start with a finite, nonempty set $\Sigma$ of symbols; called an almhabet From
the individual symbols we construct strings (or words), which are finite
sequences of symbols from the alphabet. For example, if the alphabet
$\Sigma=\{a, b\}$, then $a b a b$ and $a a a b b b a$ are strings on $\Sigma$. With few exceptions, we
will use lowercase letters $a, b, c, \ldots$ for elements of $\Sigma$ and $u, v, w, \ldots$ for string
names. We shall write, for example,
$w=a b a a a$
to indicate that the string named $w$ has the specific value abaaa.

## 2 THREE BASIC CONCEPTS: Languages

We are all familiar with the notion of natural languages, such as English, Ukrainian, Polish, German, and French. Still, most of us would probably find it difficult to say exactly what the word "language" means. Dictionaries define the term informally as a system suitable for the expression of certain ideas, facts, or concepts, including a set of symbols and rules for their manipulation. While this gives us an intuitive idea of what a language is, it is not sufficient as a definition for the study of formal languages.

to indicate that the string named $w$ has the specific value abaaa.

## 2 THREE BASIC CONCEPTS: Languages

We are all familiar with the notion of natural languages, such as English, Ukrainian, Polish, German, and French. Still, most of us would probably find it difficult to say exactly what the word "language" means. Dictionaries define the term informally as a system suitable for the expression of certain ideas, facts, or concepts, including a set of symbols and rules for their manipulation. While this gives us an intuitive idea of what a language is, it is not sufficient as a definition for the study of formal languages. We need a precise definition for the term.
the individual symbols we construct strings (or words), which are finite sequences of symbols from the alphabet. For example, if the alphabet $\Sigma=\{a, b\}$, then $a b a b$ and $a a a b b b a$ are strings on $\Sigma$. With few exceptions, we will use lowercase letters $a, b, c, \ldots$ for elements of $\Sigma$ and $u, v, w, \ldots$ for string names. We shall write, for example,
to indicate that the string named $w$ has the specific value abaaa.

## 2 THREE BASIC CONCEPTS: Languages

We are all familiar with the notion of natural languages, such as English, Ukrainian, Polish, German, and French. Still, most of us would probably find it difficult to say exactly what the word "language" means. Dictionaries define the term informally as a system suitable for the expression of certain ideas, facts, or concepts, including a set of symbols and rules for their manipulation. While this gives us an intuitive idea of what a language is, it is not sufficient as a definition for the study of formal languages. We need a precise definition for the term.
We start with a finite, nonempty set $\Sigma$ of symbols, called an alphabet. From
sequences of symbols from the alphabet. For example, if the alphabet $\Sigma=\{a, b\}$, then $a b a b$ and $a a a b b b a$ are strings on $\Sigma$. With few exceptions, we will use lowercase letters $a, b, c, \ldots$ for elements of $\Sigma$ and $u, v, w, \ldots$ for string names. We shall write, for example,
to indicate that the string named $w$ has the specific value abaaa.

## 2 THREE BASIC CONCEPTS: Languages

We are all familiar with the notion of natural languages, such as English, Ukrainian, Polish, German, and French. Still, most of us would probably find it difficult to say exactly what the word "language" means. Dictionaries define the term informally as a system suitable for the expression of certain ideas, facts, or concepts, including a set of symbols and rules for their manipulation. While this gives us an intuitive idea of what a language is, it is not sufficient as a definition for the study of formal languages. We need a precise definition for the term.
We start with a finite, nonempty set $\Sigma$ of symbols, called an alphabet. From the individual symbols we construct strings

to indicate that the string named $w$ has the specific value abaaa.

## 2 THREE BASIC CONCEPTS: Languages

We are all familiar with the notion of natural languages, such as English, Ukrainian, Polish, German, and French. Still, most of us would probably find it difficult to say exactly what the word "language" means. Dictionaries define the term informally as a system suitable for the expression of certain ideas, facts, or concepts, including a set of symbols and rules for their manipulation. While this gives us an intuitive idea of what a language is, it is not sufficient as a definition for the study of formal languages. We need a precise definition for the term.
We start with a finite, nonempty set $\Sigma$ of symbols, called an alphabet. From the individual symbols we construct strings (or words),
> $\Sigma=\{a, b\}$, then $a b a b$ and $a a a b b b a$ are strings on $\Sigma$. With few exceptions, we will use lowercase letters $a, b, c, \ldots$ for elements of $\Sigma$ and $u, v, w, \ldots$ for string names. We shall write, for example,

to indicate that the string named $w$ has the specific value abaaa.

## 2 THREE BASIC CONCEPTS: Languages

We are all familiar with the notion of natural languages, such as English, Ukrainian, Polish, German, and French. Still, most of us would probably find it difficult to say exactly what the word "language" means. Dictionaries define the term informally as a system suitable for the expression of certain ideas, facts, or concepts, including a set of symbols and rules for their manipulation. While this gives us an intuitive idea of what a language is, it is not sufficient as a definition for the study of formal languages. We need a precise definition for the term.
We start with a finite, nonempty set $\Sigma$ of symbols, called an alphabet. From the individual symbols we construct strings (or words), which are finite sequences of symbols from the alphabet.
will use lowercase letters $a, b, c, \ldots$ for elements of $\Sigma$ and $u, v, w, \ldots$ for string names. We shall write, for example,
to indicate that the string named $w$ has the specific value abaaa.

## 2 THREE BASIC CONCEPTS: Languages

We are all familiar with the notion of natural languages, such as English, Ukrainian, Polish, German, and French. Still, most of us would probably find it difficult to say exactly what the word "language" means. Dictionaries define the term informally as a system suitable for the expression of certain ideas, facts, or concepts, including a set of symbols and rules for their manipulation. While this gives us an intuitive idea of what a language is, it is not sufficient as a definition for the study of formal languages. We need a precise definition for the term.
We start with a finite, nonempty set $\Sigma$ of symbols, called an alphabet. From the individual symbols we construct strings (or words), which are finite sequences of symbols from the alphabet. For example, if the alphabet $\Sigma=\{a, b\}$, then $a b a b$ and $a a a b b b a$ are strings on $\Sigma$.
names. We shall write, for example,
to indicate that the string named $w$ has the specific value abaaa.

## 2 THREE BASIC CONCEPTS: Languages

We are all familiar with the notion of natural languages, such as English, Ukrainian, Polish, German, and French. Still, most of us would probably find it difficult to say exactly what the word "language" means. Dictionaries define the term informally as a system suitable for the expression of certain ideas, facts, or concepts, including a set of symbols and rules for their manipulation. While this gives us an intuitive idea of what a language is, it is not sufficient as a definition for the study of formal languages. We need a precise definition for the term.
We start with a finite, nonempty set $\Sigma$ of symbols, called an alphabet. From the individual symbols we construct strings (or words), which are finite sequences of symbols from the alphabet. For example, if the alphabet $\Sigma=\{a, b\}$, then $a b a b$ and $a a a b b b a$ are strings on $\Sigma$. With few exceptions, we will use lowercase letters $a, b, c, \ldots$ for elements of $\Sigma$ and $u, v, w, \ldots$ for string names.
to indicate that the string named $w$ has the specific value abaaa.

## 2 THREE BASIC CONCEPTS: Languages

We are all familiar with the notion of natural languages, such as English, Ukrainian, Polish, German, and French. Still, most of us would probably find it difficult to say exactly what the word "language" means. Dictionaries define the term informally as a system suitable for the expression of certain ideas, facts, or concepts, including a set of symbols and rules for their manipulation. While this gives us an intuitive idea of what a language is, it is not sufficient as a definition for the study of formal languages. We need a precise definition for the term.
We start with a finite, nonempty set $\Sigma$ of symbols, called an alphabet. From the individual symbols we construct strings (or words), which are finite sequences of symbols from the alphabet. For example, if the alphabet $\Sigma=\{a, b\}$, then $a b a b$ and $a a a b b b a$ are strings on $\Sigma$. With few exceptions, we will use lowercase letters $a, b, c, \ldots$ for elements of $\Sigma$ and $u, v, w, \ldots$ for string names. We shall write, for example,
to indicate that the string named $w$ has the specific value abaaa.

## 2 THREE BASIC CONCEPTS: Languages

We are all familiar with the notion of natural languages, such as English, Ukrainian, Polish, German, and French. Still, most of us would probably find it difficult to say exactly what the word "language" means. Dictionaries define the term informally as a system suitable for the expression of certain ideas, facts, or concepts, including a set of symbols and rules for their manipulation. While this gives us an intuitive idea of what a language is, it is not sufficient as a definition for the study of formal languages. We need a precise definition for the term.
We start with a finite, nonempty set $\Sigma$ of symbols, called an alphabet. From the individual symbols we construct strings (or words), which are finite sequences of symbols from the alphabet. For example, if the alphabet $\Sigma=\{a, b\}$, then $a b a b$ and $a a a b b b a$ are strings on $\Sigma$. With few exceptions, we will use lowercase letters $a, b, c, \ldots$ for elements of $\Sigma$ and $u, v, w, \ldots$ for string names. We shall write, for example,

$$
w=a b a a a
$$

to indicate that the string named $w$ has the specific value abaaa.

## 2 THREE BASIC CONCEPTS: Languages

We are all familiar with the notion of natural languages, such as English, Ukrainian, Polish, German, and French. Still, most of us would probably find it difficult to say exactly what the word "language" means. Dictionaries define the term informally as a system suitable for the expression of certain ideas, facts, or concepts, including a set of symbols and rules for their manipulation. While this gives us an intuitive idea of what a language is, it is not sufficient as a definition for the study of formal languages. We need a precise definition for the term.
We start with a finite, nonempty set $\Sigma$ of symbols, called an alphabet. From the individual symbols we construct strings (or words), which are finite sequences of symbols from the alphabet. For example, if the alphabet $\Sigma=\{a, b\}$, then $a b a b$ and $a a a b b b a$ are strings on $\Sigma$. With few exceptions, we will use lowercase letters $a, b, c, \ldots$ for elements of $\Sigma$ and $u, v, w, \ldots$ for string names. We shall write, for example,

$$
w=a b a a a
$$

to indicate that the string named $w$ has the specific value $a b a a a$.

## 2 THREE BASIC CONCEPTS: Languages

The concatenation of two strings $w$ and $v$ is the string obtained by appending the symbols of $v$ to the right end of $w$, that is, if

$$
w=a_{1} a_{2} \cdots a_{n}
$$

and

$$
v=b_{1} b_{2} \cdots b_{m},
$$

then the concatenation of $w$ and $v$, denoted by $w v$, is

$$
w v=a_{1} a_{2} \cdots a_{n} b_{1} b_{2} \cdots b_{m}
$$

The reverse of a string is obtained by writing the symbols in reverse order; if $w$ is a string as shown above, then its reverse $w^{R}$ is

$$
w^{R}=a_{n} \cdots a_{2} a_{1} .
$$

The length of a string $w$, denoted by $|w|$, is the number of symbols in the string. We shall frequently need to refer to the empty string, which is a string with no symbols at all. It will be denoted by $\lambda$. The following simple relations

$$
\begin{aligned}
|\lambda| & =0 \\
\lambda w & =w \lambda=w
\end{aligned}
$$

hold for all string $w$.

## 2 THREE BASIC CONCEPTS: Languages

The concatenation of two strings $w$ and $v$ is the string obtained by appending the symbols of $v$ to the right end of $w$, that is, if


## 2 THREE BASIC CONCEPTS: Languages

The concatenation of two strings $w$ and $v$ is the string obtained by appending the symbols of $v$ to the right end of $w$, that is, if

$$
w=a_{1} a_{2} \cdots a_{n}
$$

```
and
then the concatenation of w}\mathrm{ and v, denoted by wv, is
The reverse of a string is obtained by writing the symbols in reverse order; if w
is a string as shown above, then its reverse w
The length of a string w, denoted by }|w|\mathrm{ , is the number of symbols in the
string. We shall frequently need to refer to the empty string, which is a string
with no symbols at all. It will be denoted by }\lambda\mathrm{ . The following simple relations
```



## 2 THREE BASIC CONCEPTS: Languages

The concatenation of two strings $w$ and $v$ is the string obtained by appending the symbols of $v$ to the right end of $w$, that is, if

$$
w=a_{1} a_{2} \cdots a_{n}
$$

and
then the concatenation of $w$ and $v$, denoted by $w v$, is

The reverse of a string is obtained by writing the symbols in reverse order; if $w$ is a string as shown above, then its reverse $w^{R}$ is


The length of a string $w$, denoted by $|w|$, is the number of symbols in the string. We shall frequently need to refer to the empty string, which is a string with no symbols at all. It will be denoted by $\lambda$. The following simple relations

$$
|\lambda|=0,
$$

## 2 THREE BASIC CONCEPTS: Languages

The concatenation of two strings $w$ and $v$ is the string obtained by appending the symbols of $v$ to the right end of $w$, that is, if

$$
w=a_{1} a_{2} \cdots a_{n}
$$

and

$$
v=b_{1} b_{2} \cdots b_{m}
$$

then the concatenation of $w$ and $v$, denoted by $w v$, is

The reverse of a string is obtained by writing the symbols in reverse order; if $w$ is a string as shown above, then its reverse $w^{R}$ is

The length of a string $w$, denoted by $|w|$, is the number of symbols in the string. We shall frequently need to refer to the empty string, which is a string with no symbols at all. It will be denoted by $\lambda$. The following simple relations

$$
|\lambda|=0,
$$

## 2 THREE BASIC CONCEPTS: Languages

The concatenation of two strings $w$ and $v$ is the string obtained by appending the symbols of $v$ to the right end of $w$, that is, if

$$
w=a_{1} a_{2} \cdots a_{n}
$$

and

$$
v=b_{1} b_{2} \cdots b_{m}
$$

then the concatenation of $w$ and $v$, denoted by $w v$, is

The reverse of a string is obtained by writing the symbols in reverse order; if $w$ is a string as shown above, then its reverse $w^{R}$ is
$\square$ string. We shall frequently need to refer to the empty string, which is a string with no symbols at all. It will be denoted by $\lambda$. The following simple relations
$\qquad$

## 2 THREE BASIC CONCEPTS: Languages

The concatenation of two strings $w$ and $v$ is the string obtained by appending the symbols of $v$ to the right end of $w$, that is, if

$$
w=a_{1} a_{2} \cdots a_{n}
$$

and

$$
v=b_{1} b_{2} \cdots b_{m}
$$

then the concatenation of $w$ and $v$, denoted by $w v$, is

The reverse of a string is obtained by writing the symbols in reverse order; if $w$ is a string as shown above, then its reverse $w^{R}$ is
$\square$ string. We shall frequently need to refer to the empty string, which is a string with no symbols at all. It will be denoted by $\lambda$. The following simple relations

## 2 THREE BASIC CONCEPTS: Languages

The concatenation of two strings $w$ and $v$ is the string obtained by appending the symbols of $v$ to the right end of $w$, that is, if

$$
w=a_{1} a_{2} \cdots a_{n}
$$

and

$$
v=b_{1} b_{2} \cdots b_{m}
$$

then the concatenation of $w$ and $v$, denoted by $w v$, is

$$
w v=a_{1} a_{2} \cdots a_{n} b_{1} b_{2} \cdots b_{m} .
$$

The reverse of a string is obtained by writing the symbols in reverse order; if $w$ is a string as shown above, then its reverse $w^{R}$ is

## 2 THREE BASIC CONCEPTS: Languages

The concatenation of two strings $w$ and $v$ is the string obtained by appending the symbols of $v$ to the right end of $w$, that is, if

$$
w=a_{1} a_{2} \cdots a_{n}
$$

and

$$
v=b_{1} b_{2} \cdots b_{m}
$$

then the concatenation of $w$ and $v$, denoted by $w v$, is

$$
w v=a_{1} a_{2} \cdots a_{n} b_{1} b_{2} \cdots b_{m}
$$

The reverse of a string is obtained by writing the symbols in reverse order; is a string as shown above, then its reverse $w^{R}$ is string. We shall frequently need to refer to the empty string, which is a string with no symbols at all. It will be denoted by $\lambda$. The following simple relations

## 2 THREE BASIC CONCEPTS: Languages

The concatenation of two strings $w$ and $v$ is the string obtained by appending the symbols of $v$ to the right end of $w$, that is, if

$$
w=a_{1} a_{2} \cdots a_{n}
$$

and

$$
v=b_{1} b_{2} \cdots b_{m}
$$

then the concatenation of $w$ and $v$, denoted by $w v$, is

$$
w v=a_{1} a_{2} \cdots a_{n} b_{1} b_{2} \cdots b_{m}
$$

The reverse of a string is obtained by writing the symbols in reverse order; if $w$ is a string as shown above, then its reverse $w^{R}$ is
> string. We shall frequently need to refer to the empty string, which is a string with no symbols at all. It will be denoted by $\lambda$. The following simple relations

## 2 THREE BASIC CONCEPTS: Languages

The concatenation of two strings $w$ and $v$ is the string obtained by appending the symbols of $v$ to the right end of $w$, that is, if

$$
w=a_{1} a_{2} \cdots a_{n}
$$

and

$$
v=b_{1} b_{2} \cdots b_{m}
$$

then the concatenation of $w$ and $v$, denoted by $w v$, is

$$
w v=a_{1} a_{2} \cdots a_{n} b_{1} b_{2} \cdots b_{m}
$$

The reverse of a string is obtained by writing the symbols in reverse order; if $w$ is a string as shown above, then its reverse $w^{R}$ is

$$
w^{R}=a_{n} \cdots a_{2} a_{1}
$$

> string. We shall frequently need to refer to the empty string, which is a string with no symbols at all. It will be denoted by $\lambda$. The following simple relations

## 2 THREE BASIC CONCEPTS: Languages

The concatenation of two strings $w$ and $v$ is the string obtained by appending the symbols of $v$ to the right end of $w$, that is, if

$$
w=a_{1} a_{2} \cdots a_{n}
$$

and

$$
v=b_{1} b_{2} \cdots b_{m}
$$

then the concatenation of $w$ and $v$, denoted by $w v$, is

$$
w v=a_{1} a_{2} \cdots a_{n} b_{1} b_{2} \cdots b_{m}
$$

The reverse of a string is obtained by writing the symbols in reverse order; if $w$ is a string as shown above, then its reverse $w^{R}$ is

$$
w^{R}=a_{n} \cdots a_{2} a_{1}
$$

The length of a string $w$, string. We shall frequently need to refer to the empty string, which is a string with no symbols at all. It will be denoted by $\lambda$. The following simple relations

## 2 THREE BASIC CONCEPTS: Languages

The concatenation of two strings $w$ and $v$ is the string obtained by appending the symbols of $v$ to the right end of $w$, that is, if

$$
w=a_{1} a_{2} \cdots a_{n}
$$

and

$$
v=b_{1} b_{2} \cdots b_{m}
$$

then the concatenation of $w$ and $v$, denoted by $w v$, is

$$
w v=a_{1} a_{2} \cdots a_{n} b_{1} b_{2} \cdots b_{m}
$$

The reverse of a string is obtained by writing the symbols in reverse order; if $w$ is a string as shown above, then its reverse $w^{R}$ is

$$
w^{R}=a_{n} \cdots a_{2} a_{1}
$$

The length of a string $w$, denoted by $|w|$, string. We shall frequently need to refer to the empty string, which is a string with no symbols at all. It will be denoted by $\lambda$. The following simple relations

## 2 THREE BASIC CONCEPTS: Languages

The concatenation of two strings $w$ and $v$ is the string obtained by appending the symbols of $v$ to the right end of $w$, that is, if

$$
w=a_{1} a_{2} \cdots a_{n}
$$

and

$$
v=b_{1} b_{2} \cdots b_{m}
$$

then the concatenation of $w$ and $v$, denoted by $w v$, is

$$
w v=a_{1} a_{2} \cdots a_{n} b_{1} b_{2} \cdots b_{m}
$$

The reverse of a string is obtained by writing the symbols in reverse order; if $w$ is a string as shown above, then its reverse $w^{R}$ is

$$
w^{R}=a_{n} \cdots a_{2} a_{1}
$$

The length of a string $w$, denoted by $|w|$, is the number of symbols in the string. with no symbols at all. It will be denoted by $\lambda$. The following simple relations

## 2 THREE BASIC CONCEPTS: Languages

The concatenation of two strings $w$ and $v$ is the string obtained by appending the symbols of $v$ to the right end of $w$, that is, if

$$
w=a_{1} a_{2} \cdots a_{n}
$$

and

$$
v=b_{1} b_{2} \cdots b_{m}
$$

then the concatenation of $w$ and $v$, denoted by $w v$, is

$$
w v=a_{1} a_{2} \cdots a_{n} b_{1} b_{2} \cdots b_{m}
$$

The reverse of a string is obtained by writing the symbols in reverse order; if $w$ is a string as shown above, then its reverse $w^{R}$ is

$$
w^{R}=a_{n} \cdots a_{2} a_{1}
$$

The length of a string $w$, denoted by $|w|$, is the number of symbols in the string. We shall frequently need to refer to the empty string, which is a string with no symbols at all.

## 2 THREE BASIC CONCEPTS: Languages

The concatenation of two strings $w$ and $v$ is the string obtained by appending the symbols of $v$ to the right end of $w$, that is, if

$$
w=a_{1} a_{2} \cdots a_{n}
$$

and

$$
v=b_{1} b_{2} \cdots b_{m}
$$

then the concatenation of $w$ and $v$, denoted by $w v$, is

$$
w v=a_{1} a_{2} \cdots a_{n} b_{1} b_{2} \cdots b_{m}
$$

The reverse of a string is obtained by writing the symbols in reverse order; if $w$ is a string as shown above, then its reverse $w^{R}$ is

$$
w^{R}=a_{n} \cdots a_{2} a_{1}
$$

The length of a string $w$, denoted by $|w|$, is the number of symbols in the string. We shall frequently need to refer to the empty string, which is a string with no symbols at all. It will be denoted by $\lambda$.

## 2 THREE BASIC CONCEPTS: Languages

The concatenation of two strings $w$ and $v$ is the string obtained by appending the symbols of $v$ to the right end of $w$, that is, if

$$
w=a_{1} a_{2} \cdots a_{n}
$$

and

$$
v=b_{1} b_{2} \cdots b_{m}
$$

then the concatenation of $w$ and $v$, denoted by $w v$, is

$$
w v=a_{1} a_{2} \cdots a_{n} b_{1} b_{2} \cdots b_{m}
$$

The reverse of a string is obtained by writing the symbols in reverse order; if $w$ is a string as shown above, then its reverse $w^{R}$ is

$$
w^{R}=a_{n} \cdots a_{2} a_{1}
$$

The length of a string $w$, denoted by $|w|$, is the number of symbols in the string. We shall frequently need to refer to the empty string, which is a string with no symbols at all. It will be denoted by $\lambda$. The following simple relations

## 2 THREE BASIC CONCEPTS: Languages

The concatenation of two strings $w$ and $v$ is the string obtained by appending the symbols of $v$ to the right end of $w$, that is, if

$$
w=a_{1} a_{2} \cdots a_{n}
$$

and

$$
v=b_{1} b_{2} \cdots b_{m}
$$

then the concatenation of $w$ and $v$, denoted by $w v$, is

$$
w v=a_{1} a_{2} \cdots a_{n} b_{1} b_{2} \cdots b_{m}
$$

The reverse of a string is obtained by writing the symbols in reverse order; if $w$ is a string as shown above, then its reverse $w^{R}$ is

$$
w^{R}=a_{n} \cdots a_{2} a_{1}
$$

The length of a string $w$, denoted by $|w|$, is the number of symbols in the string. We shall frequently need to refer to the empty string, which is a string with no symbols at all. It will be denoted by $\lambda$. The following simple relations

$$
\begin{aligned}
|\lambda| & =0 \\
\lambda w & =w \lambda=w
\end{aligned}
$$

## 2 THREE BASIC CONCEPTS: Languages

The concatenation of two strings $w$ and $v$ is the string obtained by appending the symbols of $v$ to the right end of $w$, that is, if

$$
w=a_{1} a_{2} \cdots a_{n}
$$

and

$$
v=b_{1} b_{2} \cdots b_{m}
$$

then the concatenation of $w$ and $v$, denoted by $w v$, is

$$
w v=a_{1} a_{2} \cdots a_{n} b_{1} b_{2} \cdots b_{m}
$$

The reverse of a string is obtained by writing the symbols in reverse order; if $w$ is a string as shown above, then its reverse $w^{R}$ is

$$
w^{R}=a_{n} \cdots a_{2} a_{1}
$$

The length of a string $w$, denoted by $|w|$, is the number of symbols in the string. We shall frequently need to refer to the empty string, which is a string with no symbols at all. It will be denoted by $\lambda$. The following simple relations

$$
\begin{aligned}
|\lambda| & =0 \\
\lambda w & =w \lambda=w
\end{aligned}
$$

hold for all string $w$.

## 2 THREE BASIC CONCEPTS: Languages

Any string of consecutive symbols in some string $w$ is said to be a substring of w. If

$$
w=v u
$$

then the substrings $v$ and $u$ are said to be a prefix and a suffix of $w$, respectively. For example, if $w=a b b a b$, then

$$
\{\lambda, a, a b, a b b, a b b a, a b b a b\}
$$

is the set of all prefixes of $w$, while $b a b, a b, b$ are some of its suffixes. Simple properties of strings, such as their length, are very intuitive and probably need little elaboration. For example, if $u$ and $v$ are strings, then the length of their concatenation is the sum of the individual lengths, that is,

$$
\begin{equation*}
|u v|=|u|+|v| \tag{1}
\end{equation*}
$$

But although this relationship is obvious, it is useful to be able to make it precise and prove it. The techniques for doing so are important in more complicated situations.

## 2 THREE BASIC CONCEPTS: Languages

Any string of consecutive symbols in some string $w$ is said to be a substring of $w$.

```
w=vu,
```

then the substrings $v$ and $u$ are said to be a prefix and a suffix of $w$, respectively. For example, if $w=a b b a b$, then
$\{\lambda, a, a b, a b b, a b b a, a b b a b\}$
is the set of all prefixes of $w$, while $b a b, a b, b$ are some of its suffixes. Simple properties of strings, such as their length, are very intuitive and probably need little elaboration. For example, if $u$ and $v$ are strings, then the length of their concatenation is the sum of the individual lengths, that is,

$$
\begin{equation*}
|u v|=|u|+|v| \tag{1}
\end{equation*}
$$

But although this relationship is obvious, it is useful to be able to make it precise and prove it. The techniques for doing so are important in more complicated situations.

## 2 THREE BASIC CONCEPTS: Languages

Any string of consecutive symbols in some string $w$ is said to be a substring of $w$. If

```
w=vu,
```

then the substrings $v$ and $u$ are said to be a prefix and a suffix of $w$, respectively. For example, if $w=a b b a b$, then
$\{\lambda, a, a b, a b b, a b b a, a b b a b\}$
is the set of all prefixes of $w$, while $b a b, a b, b$ are some of its suffixes. Simple properties of strings, such as their length, are very intuitive and probably need little elaboration. For example, if $u$ and $v$ are strings, then the length of their concatenation is the sum of the individual lengths, that is,

$$
\begin{equation*}
|u v|=|u|+|v| . \tag{1}
\end{equation*}
$$

But although this relationship is obvious, it is useful to be able to make it precise and prove it. The techniques for doing so are important in more complicated situations.

## 2 THREE BASIC CONCEPTS: Languages

Any string of consecutive symbols in some string $w$ is said to be a substring of $w$. If

$$
w=v u
$$

then the substrings $v$ and $u$ are said to be a prefix and a suffix of $w$, respectively. For example, if $w=a b b a b$, then
$\left\{\lambda, a, a^{\prime}, a^{\prime \prime} b, a b^{\prime} b a, a^{b} b a b\right\}$
is the set of all prefixes of $w$, while $b a b, a b, b$ are some of its suffixes. Simple properties of strings, such as their length, are very intuitive and probably need little elaboration. For example, if $u$ and $v$ are strings, then the length of their concatenation is the sum of the individual lengths, that is,

$$
\begin{equation*}
|u v|=|u|+|v| . \tag{1}
\end{equation*}
$$

But although this relationship is obvious, it is useful to be able to make it precise and prove it. The techniques for doing so are important in more complicated situations.

## 2 THREE BASIC CONCEPTS: Languages

Any string of consecutive symbols in some string $w$ is said to be a substring of $w$. If

$$
w=v u
$$

then the substrings $v$ and $u$ are said to be a prefix and a suffix of $w$,
respectively. For example, if $w=a b b a b$, then
$\{\lambda, a, a b, a b b, a b b a, a b b a b\}$
is the set of all prefixes of $w$, while $b a b, a b, b$ are some of its suffixes. Simple properties of strings, such as their length, are very intuitive and probably need little elaboration. For example, if $u$ and $v$ are strings, then the length of their concatenation is the sum of the individual lengths, that is,

$$
\begin{equation*}
|u v|=|u|+|v| . \tag{1}
\end{equation*}
$$

But although this relationship is obvious, it is useful to be able to make it precise and prove it. The techniques for doing so are important in more complicated situations.

## 2 THREE BASIC CONCEPTS: Languages

Any string of consecutive symbols in some string $w$ is said to be a substring of $w$. If

$$
w=v u
$$

then the substrings $v$ and $u$ are said to be a prefix and a suffix of $w$, respectively. For example, if $w=a b b a b$, then
$\{\lambda, a, a b, a b b, a b b a, a b b a b\}$
> is the set of all prefixes of $\omega$, while $b a^{\prime}, a^{\prime}, b$ are some of its suffixes. Simple properties of strings, such as their length, are very intuitive and probably need little elaboration. For example, if $u$ and $v$ are strings, then the length of their concatenation is the sum of the individual lengths, that is,

But although this relationship is obvious, it is useful to be able to make it precise and prove it. The techniques for doing so are important in more complicated situations.

## 2 THREE BASIC CONCEPTS: Languages

Any string of consecutive symbols in some string $w$ is said to be a substring of $w$. If

$$
w=v u
$$

then the substrings $v$ and $u$ are said to be a prefix and a suffix of $w$, respectively. For example, if $w=a b b a b$, then
$\{\lambda, a, a b, a b b, a b b a, a b b a b\}$
> is the set of all prefixes of $w$, while $b a b, a b, b$ are some of its suffixes. Simple properties of strings, such as their length, are very intuitive and probably need little elaboration. For example, if $u$ and $v$ are strings, then the length of their concatenation is the sum of the individual lengths, that is,

But although this relationship is obvious, it is useful to be able to make it precise and prove it. The techniques for doing so are important in more complicated situations.

## 2 THREE BASIC CONCEPTS: Languages

Any string of consecutive symbols in some string $w$ is said to be a substring of $w$. If

$$
w=v u
$$

then the substrings $v$ and $u$ are said to be a prefix and a suffix of $w$, respectively. For example, if $w=a b b a b$, then

$$
\{\lambda, a, a b, a b b, a b b a, a b b a b\}
$$

> is the set of all prefixes of $w$, while $b a b, a b, b$ are some of its suffixes. Simple properties of strings, such as their length, are very intuitive and probably need little elaboration. For example, if $u$ and $v$ are strings, then the length of their concatenation is the sum of the individual lengths, that is,

But although this relationship is obvious, it is useful to be able to make it precise and prove it. The techniques for doing so are important in more complicated situations.

## 2 THREE BASIC CONCEPTS: Languages

Any string of consecutive symbols in some string $w$ is said to be a substring of w. If

$$
w=v u
$$

then the substrings $v$ and $u$ are said to be a prefix and a suffix of $w$, respectively. For example, if $w=a b b a b$, then

$$
\{\lambda, a, a b, a b b, a b b a, a b b a b\}
$$

is the set of all prefixes of $w$, while $b a b, a b, b$ are some of its suffixes.
> properties of strings, such as their length, are very intuitive and probably need little elaboration. For example, if $u$ and $v$ are strings, then the length of their concatenation is the sum of the individual lengths, that is,

But although this relationship is obvious, it is useful to be able to make it precise and prove it. The techniques for doing so are important in more complicated situations.

## 2 THREE BASIC CONCEPTS: Languages

Any string of consecutive symbols in some string $w$ is said to be a substring of w. If

$$
w=v u
$$

then the substrings $v$ and $u$ are said to be a prefix and a suffix of $w$, respectively. For example, if $w=a b b a b$, then

$$
\{\lambda, a, a b, a b b, a b b a, a b b a b\}
$$

is the set of all prefixes of $w$, while $b a b, a b, b$ are some of its suffixes. Simple properties of strings, such as their length, are very intuitive and probably need little elaboration.
concatenation is the sum of the individual lengths, that is,

But although this relationship is obvious, it is useful to be able to make it precise and prove it. The techniques for doing so are important in more complicated situations.

## 2 THREE BASIC CONCEPTS: Languages

Any string of consecutive symbols in some string $w$ is said to be a substring of w. If

$$
w=v u
$$

then the substrings $v$ and $u$ are said to be a prefix and a suffix of $w$, respectively. For example, if $w=a b b a b$, then

$$
\{\lambda, a, a b, a b b, a b b a, a b b a b\}
$$

is the set of all prefixes of $w$, while $b a b, a b, b$ are some of its suffixes. Simple properties of strings, such as their length, are very intuitive and probably need little elaboration. For example, if $u$ and $v$ are strings,
concatenation is the sum of the individual lengths, that is,

But although this relationship is obvious, it is useful to be able to make it precise and prove it. The techniques for doing so are important in more complicated situations.

## 2 THREE BASIC CONCEPTS: Languages

Any string of consecutive symbols in some string $w$ is said to be a substring of w. If

$$
w=v u
$$

then the substrings $v$ and $u$ are said to be a prefix and a suffix of $w$, respectively. For example, if $w=a b b a b$, then

$$
\{\lambda, a, a b, a b b, a b b a, a b b a b\}
$$

is the set of all prefixes of $w$, while $b a b, a b, b$ are some of its suffixes. Simple properties of strings, such as their length, are very intuitive and probably need little elaboration. For example, if $u$ and $v$ are strings, then the length of their concatenation is the sum of the individual lengths,

But although this relationship is obvious, it is useful to be able to make it precise and prove it. The techniques for doing so are important in more complicated situations.

## 2 THREE BASIC CONCEPTS: Languages

Any string of consecutive symbols in some string $w$ is said to be a substring of w. If

$$
w=v u
$$

then the substrings $v$ and $u$ are said to be a prefix and a suffix of $w$, respectively. For example, if $w=a b b a b$, then

$$
\{\lambda, a, a b, a b b, a b b a, a b b a b\}
$$

is the set of all prefixes of $w$, while $b a b, a b, b$ are some of its suffixes. Simple properties of strings, such as their length, are very intuitive and probably need little elaboration. For example, if $u$ and $v$ are strings, then the length of their concatenation is the sum of the individual lengths, that is,

[^0]
## 2 THREE BASIC CONCEPTS: Languages

Any string of consecutive symbols in some string $w$ is said to be a substring of w. If

$$
w=v u
$$

then the substrings $v$ and $u$ are said to be a prefix and a suffix of $w$, respectively. For example, if $w=a b b a b$, then

$$
\{\lambda, a, a b, a b b, a b b a, a b b a b\}
$$

is the set of all prefixes of $w$, while $b a b, a b, b$ are some of its suffixes. Simple properties of strings, such as their length, are very intuitive and probably need little elaboration. For example, if $u$ and $v$ are strings, then the length of their concatenation is the sum of the individual lengths, that is,

$$
\begin{equation*}
|u v|=|u|+|v| \tag{1}
\end{equation*}
$$

But although this relationship is obvious, it is useful to be able to make it precise and prove it. The techniques for doing so are important in more complicated situations.

## 2 THREE BASIC CONCEPTS: Languages

Any string of consecutive symbols in some string $w$ is said to be a substring of w. If

$$
w=v u
$$

then the substrings $v$ and $u$ are said to be a prefix and a suffix of $w$, respectively. For example, if $w=a b b a b$, then

$$
\{\lambda, a, a b, a b b, a b b a, a b b a b\}
$$

is the set of all prefixes of $w$, while $b a b, a b, b$ are some of its suffixes. Simple properties of strings, such as their length, are very intuitive and probably need little elaboration. For example, if $u$ and $v$ are strings, then the length of their concatenation is the sum of the individual lengths, that is,

$$
\begin{equation*}
|u v|=|u|+|v| . \tag{1}
\end{equation*}
$$

But although this relationship is obvious, $\qquad$
precise and prove it. The techniques for doing so are important in more
complicated situations.

## 2 THREE BASIC CONCEPTS: Languages

Any string of consecutive symbols in some string $w$ is said to be a substring of w. If

$$
w=v u
$$

then the substrings $v$ and $u$ are said to be a prefix and a suffix of $w$, respectively. For example, if $w=a b b a b$, then

$$
\{\lambda, a, a b, a b b, a b b a, a b b a b\}
$$

is the set of all prefixes of $w$, while $b a b, a b, b$ are some of its suffixes. Simple properties of strings, such as their length, are very intuitive and probably need little elaboration. For example, if $u$ and $v$ are strings, then the length of their concatenation is the sum of the individual lengths, that is,

$$
\begin{equation*}
|u v|=|u|+|v| . \tag{1}
\end{equation*}
$$

But although this relationship is obvious, it is useful to be able to make it precise and prove it.
complicated situations.

## 2 THREE BASIC CONCEPTS: Languages

Any string of consecutive symbols in some string $w$ is said to be a substring of w. If

$$
w=v u
$$

then the substrings $v$ and $u$ are said to be a prefix and a suffix of $w$, respectively. For example, if $w=a b b a b$, then

$$
\{\lambda, a, a b, a b b, a b b a, a b b a b\}
$$

is the set of all prefixes of $w$, while $b a b, a b, b$ are some of its suffixes. Simple properties of strings, such as their length, are very intuitive and probably need little elaboration. For example, if $u$ and $v$ are strings, then the length of their concatenation is the sum of the individual lengths, that is,

$$
\begin{equation*}
|u v|=|u|+|v| . \tag{1}
\end{equation*}
$$

But although this relationship is obvious, it is useful to be able to make it precise and prove it. The techniques for doing so are important in more complicated situations.

## Example 1.8

## Example 1.8

Show that formula

$$
\begin{equation*}
|u v|=|u|+|v| . \tag{1}
\end{equation*}
$$

holds for any strings $u$ and $v$. To prove this, we first need a definition of the length of a string. We make such a definition in a recursive fashion by

$$
\begin{aligned}
|a| & =1 \\
|w a| & =|w|+1
\end{aligned}
$$

for all $a \in \Sigma$ and $w$ any string on $\Sigma$. This definition is a formal statement of our intuitive understanding of the length of a string: The length of a single symbol is one, and the length of any string is increased by one if we add another symbol to it. With this formal definition, we are ready to prove equality (1) by induction characters.

## Example 1.8

## Show that formula

$$
\begin{equation*}
|u v|=|u|+|v| . \tag{1}
\end{equation*}
$$

holds for any strings $u$ and $v$. To prove this, we first need a definition of the length of a string. We make such a definition in a recursive fashion by

$$
\begin{aligned}
|a| & =1 \\
|w a| & =|w|+1
\end{aligned}
$$

for all $a \in \Sigma$ and $w$ any string on $\Sigma$. This definition is a formal statement of our intuitive understanding of the length of a string: The length of a single symbol is one, and the length of any string is increased by one if we add another symbol to it. With this formal definition, we are ready to prove equality (1) by induction characters.

## Example 1.8

Show that formula

$$
\begin{equation*}
|u v|=|u|+|v| \tag{1}
\end{equation*}
$$

holds for any strings $u$ and $v$. To prove this, we first need a definition of the length of a string. We make such a definition in a recursive fashion by

$$
\begin{aligned}
\mid a^{1} & =1 \\
|w a| & =|w|+1
\end{aligned}
$$

for all $a \in \Sigma$ and $w$ any string on $\Sigma$. This definition is a formal statement of our intuitive understanding of the length of a string: The length of a single symbol is one, and the length of any string is increased by one if we add another symbol to it. With this formal definition, we are ready to prove equality (1) by induction characters.

## 2 THREE BASIC CONCEPTS: Languages

## Example 1.8

Show that formula

$$
\begin{equation*}
|u v|=|u|+|v| . \tag{1}
\end{equation*}
$$

holds for any strings $u$ and $v$. To prove this, we first need a definition of the length of a string. We make such a definition in a recursive fashion by

$$
\begin{aligned}
|a| & =1 \\
|w a| & =|w|+1
\end{aligned}
$$

for all $a \in \Sigma$ and $w$ any string on $\Sigma$. This definition is a formal statement of our intuitive understanding of the length of a string: The length of a single symbol is one, and the length of any string is increased by one if we add another symbol to it. With this formal definition, we are ready to prove equality (1) by induction characters.

## 2 THREE BASIC CONCEPTS: Languages

## Example 1.8

Show that formula

$$
\begin{equation*}
|u v|=|u|+|v| . \tag{1}
\end{equation*}
$$

holds for any strings $u$ and $v$. To prove this, we first need a definition of the length of a string. We make such a definition in a recursive fashion by

$$
\begin{aligned}
|a| & =1, \\
|w a| & =|w|+1,
\end{aligned}
$$

for all $a \in \Sigma$ and $w$ any string on $\Sigma$. This definition is a formal statement of our intuitive understanding of the length of a string: The length of a single symbol is one, and the length of any string is increased by one if we add another symbol to it. With this formal definition, we are ready to prove equality (1) by induction characters.

## 2 THREE BASIC CONCEPTS: Languages

## Example 1.8

Show that formula

$$
\begin{equation*}
|u v|=|u|+|v| . \tag{1}
\end{equation*}
$$

holds for any strings $u$ and $v$. To prove this, we first need a definition of the length of a string. We make such a definition in a recursive fashion by
> for all $a \in \Sigma$ and $w$ any string on $\Sigma$. This definition is a formal statement of our intuitive understanding of the length of a string: The length of a single symbol is one, and the length of any string is increased by one if we add another symbol to it. With this formal definition, we are ready to prove equality (1) by induction characters.

## 2 THREE BASIC CONCEPTS: Languages

## Example 1.8

Show that formula

$$
\begin{equation*}
|u v|=|u|+|v| . \tag{1}
\end{equation*}
$$

holds for any strings $u$ and $v$. To prove this, we first need a definition of the length of a string. We make such a definition in a recursive fashion by

$$
\begin{aligned}
|a| & =1 \\
|w a| & =|w|+1
\end{aligned}
$$

for all $a \in \Sigma$ and $w$ any string on $\Sigma$. This definition is a formal statement of our intuitive understanding of the length of a string: The length of a single symbol is one, and the length of any string is increased by one if we add another symbol to it. With this formal definition, we are ready to prove equality (1) by induction characters.

## 2 THREE BASIC CONCEPTS: Languages

## Example 1.8

Show that formula

$$
\begin{equation*}
|u v|=|u|+|v| . \tag{1}
\end{equation*}
$$

holds for any strings $u$ and $v$. To prove this, we first need a definition of the length of a string. We make such a definition in a recursive fashion by

$$
\begin{aligned}
|a| & =1 \\
|w a| & =|w|+1
\end{aligned}
$$

for all $a \in \Sigma$ and $w$ any string on $\Sigma$. This definition is a formal statement of our intuitive understanding of the length of a string: The length of a single symbol is one, and the length of any string is increased by one if we add another symbol to it. With this formal definition, we are ready to prove equality (1) by induction characters.

## 2 THREE BASIC CONCEPTS: Languages

## Example 1.8

Show that formula

$$
\begin{equation*}
|u v|=|u|+|v| . \tag{1}
\end{equation*}
$$

holds for any strings $u$ and $v$. To prove this, we first need a definition of the length of a string. We make such a definition in a recursive fashion by

$$
\begin{aligned}
|a| & =1 \\
|w a| & =|w|+1
\end{aligned}
$$

for all $a \in \Sigma$ and $w$ any string on $\Sigma$. This definition is a formal statement of our intuitive understanding of the length of a string:
symbol is one, and the length of any string is increased by one if we add
another symbol to it. With this formal definition, we are ready to prove equality (1) by induction characters.

## 2 THREE BASIC CONCEPTS: Languages

## Example 1.8

Show that formula

$$
\begin{equation*}
|u v|=|u|+|v| . \tag{1}
\end{equation*}
$$

holds for any strings $u$ and $v$. To prove this, we first need a definition of the length of a string. We make such a definition in a recursive fashion by

$$
\begin{aligned}
|a| & =1 \\
|w a| & =|w|+1
\end{aligned}
$$

for all $a \in \Sigma$ and $w$ any string on $\Sigma$. This definition is a formal statement of our intuitive understanding of the length of a string: The length of a single symbol is one, and the length of any string is increased by one if we add
another symbol to it. With this formal definition, we are ready to prove equality (1) by induction characters.

## 2 THREE BASIC CONCEPTS: Languages

## Example 1.8

Show that formula

$$
\begin{equation*}
|u v|=|u|+|v| . \tag{1}
\end{equation*}
$$

holds for any strings $u$ and $v$. To prove this, we first need a definition of the length of a string. We make such a definition in a recursive fashion by

$$
\begin{aligned}
|a| & =1 \\
|w a| & =|w|+1
\end{aligned}
$$

for all $a \in \Sigma$ and $w$ any string on $\Sigma$. This definition is a formal statement of our intuitive understanding of the length of a string: The length of a single symbol is one, and the length of any string is increased by one if we add another symbol to it.
(1) by induction characters.

## 2 THREE BASIC CONCEPTS: Languages

## Example 1.8

Show that formula

$$
\begin{equation*}
|u v|=|u|+|v| . \tag{1}
\end{equation*}
$$

holds for any strings $u$ and $v$. To prove this, we first need a definition of the length of a string. We make such a definition in a recursive fashion by

$$
\begin{aligned}
|a| & =1 \\
|w a| & =|w|+1
\end{aligned}
$$

for all $a \in \Sigma$ and $w$ any string on $\Sigma$. This definition is a formal statement of our intuitive understanding of the length of a string: The length of a single symbol is one, and the length of any string is increased by one if we add another symbol to it. With this formal definition, we are ready to prove equality (1) by induction characters.

## Example 1.8 (continuation)

## 2 THREE BASIC CONCEPTS: Languages

Example 1.8 (continuation)
By definition, the equality

$$
\begin{equation*}
|u v|=|u|+|v| . \tag{1}
\end{equation*}
$$

holds for all strings $u$ of any length and all strings $v$ of length 1 , so we have a basis. As an inductive assumption, we take that equality (1) holds for all strings $u$ of any length and all strings $v$ of length $1,2, \ldots, n$. Now take any string $v$ of length $n+1$ and write it as $v=w a$. Then,

$$
\begin{aligned}
& v^{\prime}\left|=|\omega|^{\prime}+1\right. \\
& |u v|=|u \omega a|=|u \omega|+1 .
\end{aligned}
$$

By the inductive hypothesis (which is applicable since $w$ is of length $n$ ),

$$
\left.\right|_{u w}|=|u|+|w|
$$

so that

$$
|u v|=|u|+|w|+1=|u|+|v| .
$$

Therefore, equality (1) holds for all $u$ and all $v$ of length up to $n+1$, completing the inductive step and the argument.

## 2 THREE BASIC CONCEPTS: Languages

## Example 1.8 (continuation)

By definition, the equality

$$
\begin{equation*}
|u v|=|u|+|v| . \tag{1}
\end{equation*}
$$

holds for all strings $u$ of any length and all strings $v$ of length 1 , so we have a basis. As an inductive assumption, we take that equality (1) holds for all strings $u$ of any length and all strings $v$ of length $1,2, \ldots, n$. Now take any string $v$ of length $n+1$ and write it as $v=w a$. Then,

$$
\begin{aligned}
& \left|v^{\prime}={ }^{\prime}\right| w^{\prime}+1 \\
& |w v|=|u w a|=|u w|+1 .
\end{aligned}
$$

By the inductive hypothesis (which is applicable since $w$ is of length $n$ ),

$$
\left.\right|_{u n u}\left|=|u|+\left|{ }_{u}\right|\right.
$$

so that

$$
|u v|=|u|+|w|+1=|u|+|v| .
$$

Therefore, equality (1) holds for all $u$ and all $v$ of length up to $n+1$, completing the inductive step and the argument.

## 2 THREE BASIC CONCEPTS: Languages

Example 1.8 (continuation)
By definition, the equality

$$
\begin{equation*}
|u v|=|u|+|v| . \tag{1}
\end{equation*}
$$

holds for all strings $u$ of any length and all strings $v$ of length 1 , so we have a basis. As an inductive assumption, we take that equality (1) holds for all strings $u$ of any length and all strings $v$ of length $1,2, \ldots, n$. Now take any string $v$ of length $n+1$ and write it as $v=w a$. Then,

$$
\begin{aligned}
|v| & =|w|+1 \\
|u v| & =|u w a|=|u w|+1 .
\end{aligned}
$$

By the inductive hypothesis (which is applicable since $w$ is of length $n$ ),

$$
\left.{ }^{\mid z u w}\right|^{\prime}=|u|+\left|w^{\prime}\right|
$$

so that

$$
|u v|=|u|+|w|+1=|u|+|v| .
$$

Therefore, equality (1) holds for all $u$ and all $v$ of length up to $n+1$, completing the inductive step and the argument.

## 2 THREE BASIC CONCEPTS: Languages

Example 1.8 (continuation)
By definition, the equality

$$
\begin{equation*}
|u v|=|u|+|v| . \tag{1}
\end{equation*}
$$

holds for all strings $u$ of any length and all strings $v$ of length 1 , so we have a basis. As an inductive assumption, we take that equality (1) holds for all strings $u$ of any length and all strings $v$ of length $1,2, \ldots, n$. Now take any string $v$ of length $n+1$ and write it as $v=w a$. Then,

$$
|u v|=|u w a|=|u w|+1 .
$$

By the inductive hypothesis (which is applicable since $w$ is of length $n$ ),

$$
\left.t_{u v}\right|^{\prime}=|u|+|w|
$$

so that

Therefore, equality (1) holds for all $u$ and all $v$ of length up to $n+1$, completing the inductive step and the argument.

## 2 THREE BASIC CONCEPTS: Languages

Example 1.8 (continuation)
By definition, the equality

$$
\begin{equation*}
|u v|=|u|+|v| . \tag{1}
\end{equation*}
$$

holds for all strings $u$ of any length and all strings $v$ of length 1 , so we have a basis.
$u$ of any length and all strings $v$ of length 1,2 ,
length $n+1$ and write it as $v=w a$. Then,

$$
|u v|=|u w a|=|u w|+1 .
$$

By the inductive hypothesis (which is applicable since $w$ is of length $n$ ),
so that

Therefore, equality (1) holds for all $u$ and all $v$ of length up to $n+1$, completing the inductive step and the argument.

## 2 THREE BASIC CONCEPTS: Languages

Example 1.8 (continuation)
By definition, the equality

$$
\begin{equation*}
|u v|=|u|+|v| . \tag{1}
\end{equation*}
$$

holds for all strings $u$ of any length and all strings $v$ of length 1 , so we have a basis. As an inductive assumption,
length $n+1$ and write it as $v=w a$. Then,

$$
|u v|=|u w a|=|u w|+1 .
$$

By the inductive hypothesis (which is applicable since $w$ is of length $n$ ),
so that

Therefore, equality (1) holds for all $u$ and all $v$ of length up to $n+1$, completing the inductive step and the argument.

## 2 THREE BASIC CONCEPTS: Languages

## Example 1.8 (continuation)

By definition, the equality

$$
\begin{equation*}
|u v|=|u|+|v| . \tag{1}
\end{equation*}
$$

holds for all strings $u$ of any length and all strings $v$ of length 1 , so we have a basis. As an inductive assumption, we take that equality (1) holds for all strings $u$ of any length and all strings $v$ of length $1,2, \ldots, n$.
length $n+1$ and write it as $v=w a$. Then,

By the inductive hypothesis (which is applicable since $w$ is of length $n$ ),
so that

Therefore, equality (1) holds for all $u$ and all $v$ of length up to $n+1$, completing the inductive step and the argument.

## 2 THREE BASIC CONCEPTS: Languages

## Example 1.8 (continuation)

By definition, the equality

$$
\begin{equation*}
|u v|=|u|+|v| . \tag{1}
\end{equation*}
$$

holds for all strings $u$ of any length and all strings $v$ of length 1 , so we have a basis. As an inductive assumption, we take that equality (1) holds for all strings $u$ of any length and all strings $v$ of length $1,2, \ldots, n$. Now take any string $v$ of length $n+1$ and write it as $v=w a$.

By the inductive hypothesis (which is applicable since $w$ is of length $n$ ),
so that

Therefore, equality (1) holds for all $u$ and all $v$ of length up to $n+1$, completing the inductive step and the argument.

## 2 THREE BASIC CONCEPTS: Languages

## Example 1.8 (continuation)

By definition, the equality

$$
\begin{equation*}
|u v|=|u|+|v| . \tag{1}
\end{equation*}
$$

holds for all strings $u$ of any length and all strings $v$ of length 1 , so we have a basis. As an inductive assumption, we take that equality (1) holds for all strings $u$ of any length and all strings $v$ of length $1,2, \ldots, n$. Now take any string $v$ of length $n+1$ and write it as $v=w a$. Then,

$$
\text { By the inductive hypothesis (which is applicable since } w \text { is of length } n \text { ), }
$$

## so that

## 2 THREE BASIC CONCEPTS: Languages

## Example 1.8 (continuation)

By definition, the equality

$$
\begin{equation*}
|u v|=|u|+|v| . \tag{1}
\end{equation*}
$$

holds for all strings $u$ of any length and all strings $v$ of length 1 , so we have a basis. As an inductive assumption, we take that equality (1) holds for all strings $u$ of any length and all strings $v$ of length $1,2, \ldots, n$. Now take any string $v$ of length $n+1$ and write it as $v=w a$. Then,

$$
\begin{aligned}
|v| & =|w|+1 \\
|u v| & =|u w a|=|u w|+1
\end{aligned}
$$

By the inductive hypothesis (which is applicable since $w$ is of length $n$ ),

## so that

## 2 THREE BASIC CONCEPTS: Languages

## Example 1.8 (continuation)

By definition, the equality

$$
\begin{equation*}
|u v|=|u|+|v| . \tag{1}
\end{equation*}
$$

holds for all strings $u$ of any length and all strings $v$ of length 1 , so we have a basis. As an inductive assumption, we take that equality (1) holds for all strings $u$ of any length and all strings $v$ of length $1,2, \ldots, n$. Now take any string $v$ of length $n+1$ and write it as $v=w a$. Then,

$$
\begin{aligned}
|v| & =|w|+1 \\
|u v| & =|u w a|=|u w|+1
\end{aligned}
$$

By the inductive hypothesis

$$
\text { (which is applicable since } w \text { is of length } n \text { ), }
$$

## so that

## 2 THREE BASIC CONCEPTS: Languages

## Example 1.8 (continuation)

By definition, the equality

$$
\begin{equation*}
|u v|=|u|+|v| . \tag{1}
\end{equation*}
$$

holds for all strings $u$ of any length and all strings $v$ of length 1 , so we have a basis. As an inductive assumption, we take that equality (1) holds for all strings $u$ of any length and all strings $v$ of length $1,2, \ldots, n$. Now take any string $v$ of length $n+1$ and write it as $v=w a$. Then,

$$
\begin{aligned}
|v| & =|w|+1 \\
|u v| & =|u w a|=|u w|+1
\end{aligned}
$$

By the inductive hypothesis (which is applicable since $w$ is of length $n$ ),

## so that

Therefore, equality (1) holds for all $u$ and all $v$ of length up to $n+1$, completing the inductive step and the argument.

## 2 THREE BASIC CONCEPTS: Languages

## Example 1.8 (continuation)

By definition, the equality

$$
\begin{equation*}
|u v|=|u|+|v| . \tag{1}
\end{equation*}
$$

holds for all strings $u$ of any length and all strings $v$ of length 1 , so we have a basis. As an inductive assumption, we take that equality (1) holds for all strings $u$ of any length and all strings $v$ of length $1,2, \ldots, n$. Now take any string $v$ of length $n+1$ and write it as $v=w a$. Then,

$$
\begin{aligned}
|v| & =|w|+1 \\
|u v| & =|u w a|=|u w|+1
\end{aligned}
$$

By the inductive hypothesis (which is applicable since $w$ is of length $n$ ),

$$
|u w|=|u|+|w|
$$

## so that

Therefore, equality (1) holds for all $u$ and all $v$ of length up to $n+1$, completing the inductive step and the argument.

## 2 THREE BASIC CONCEPTS: Languages

## Example 1.8 (continuation)

By definition, the equality

$$
\begin{equation*}
|u v|=|u|+|v| . \tag{1}
\end{equation*}
$$

holds for all strings $u$ of any length and all strings $v$ of length 1 , so we have a basis. As an inductive assumption, we take that equality (1) holds for all strings $u$ of any length and all strings $v$ of length $1,2, \ldots, n$. Now take any string $v$ of length $n+1$ and write it as $v=w a$. Then,

$$
\begin{aligned}
|v| & =|w|+1 \\
|u v| & =|u w a|=|u w|+1
\end{aligned}
$$

By the inductive hypothesis (which is applicable since $w$ is of length $n$ ),

$$
|u w|=|u|+|w|
$$

so that

Therefore, equality (1) holds for all $u$ and all $v$ of length up to $n+1$, completing the inductive step and the argument.

## 2 THREE BASIC CONCEPTS: Languages

## Example 1.8 (continuation)

By definition, the equality

$$
\begin{equation*}
|u v|=|u|+|v| . \tag{1}
\end{equation*}
$$

holds for all strings $u$ of any length and all strings $v$ of length 1 , so we have a basis. As an inductive assumption, we take that equality (1) holds for all strings $u$ of any length and all strings $v$ of length $1,2, \ldots, n$. Now take any string $v$ of length $n+1$ and write it as $v=w a$. Then,

$$
\begin{aligned}
|v| & =|w|+1 \\
|u v| & =|u w a|=|u w|+1
\end{aligned}
$$

By the inductive hypothesis (which is applicable since $w$ is of length $n$ ),

$$
|u w|=|u|+|w|
$$

so that

$$
|u v|=|u|+|w|+1=|u|+|v| .
$$

Therefore, equality (1) holds for all $u$ and all $v$ of length up to $n+1$, completing the inductive step and the argument.

## 2 THREE BASIC CONCEPTS: Languages

## Example 1.8 (continuation)

By definition, the equality

$$
\begin{equation*}
|u v|=|u|+|v| . \tag{1}
\end{equation*}
$$

holds for all strings $u$ of any length and all strings $v$ of length 1 , so we have a basis. As an inductive assumption, we take that equality (1) holds for all strings $u$ of any length and all strings $v$ of length $1,2, \ldots, n$. Now take any string $v$ of length $n+1$ and write it as $v=w a$. Then,

$$
\begin{aligned}
|v| & =|w|+1 \\
|u v| & =|u w a|=|u w|+1
\end{aligned}
$$

By the inductive hypothesis (which is applicable since $w$ is of length $n$ ),

$$
|u w|=|u|+|w|
$$

so that

$$
|u v|=|u|+|w|+1=|u|+|v| .
$$

Therefore, equality (1) holds for all $u$ and all $v$ of length up to $n+1$, completing the inductive step and the argument.

## 2 THREE BASIC CONCEPTS: Languages

## Example 1.8 (continuation)

By definition, the equality

$$
\begin{equation*}
|u v|=|u|+|v| . \tag{1}
\end{equation*}
$$

holds for all strings $u$ of any length and all strings $v$ of length 1 , so we have a basis. As an inductive assumption, we take that equality (1) holds for all strings $u$ of any length and all strings $v$ of length $1,2, \ldots, n$. Now take any string $v$ of length $n+1$ and write it as $v=w a$. Then,

$$
\begin{aligned}
|v| & =|w|+1 \\
|u v| & =|u w a|=|u w|+1
\end{aligned}
$$

By the inductive hypothesis (which is applicable since $w$ is of length $n$ ),

$$
|u w|=|u|+|w|
$$

so that

$$
|u v|=|u|+|w|+1=|u|+|v| .
$$

Therefore, equality (1) holds for all $u$ and all $v$ of length up to $n+1$, completing the inductive step and the argument.

## 2 THREE BASIC CONCEPTS: Languages

## Example 1.8 (continuation)

By definition, the equality

$$
\begin{equation*}
|u v|=|u|+|v| . \tag{1}
\end{equation*}
$$

holds for all strings $u$ of any length and all strings $v$ of length 1 , so we have a basis. As an inductive assumption, we take that equality (1) holds for all strings $u$ of any length and all strings $v$ of length $1,2, \ldots, n$. Now take any string $v$ of length $n+1$ and write it as $v=w a$. Then,

$$
\begin{aligned}
|v| & =|w|+1 \\
|u v| & =|u w a|=|u w|+1
\end{aligned}
$$

By the inductive hypothesis (which is applicable since $w$ is of length $n$ ),

$$
|u w|=|u|+|w|
$$

so that

$$
|u v|=|u|+|w|+1=|u|+|v| .
$$

Therefore, equality (1) holds for all $u$ and all $v$ of length up to $n+1$, completing the inductive step and the argument.

## 2 THREE BASIC CONCEPTS: Languages

If $w$ is a string, then $w^{n}$ stands for the string obtained by repeating $w n$ times. As a special case, we define

$$
w^{0}=\lambda,
$$

for all strings $w$.
If $\Sigma$ is an alphabet, then we use $\Sigma^{*}$ to denote the set of strings obtained by concatenating zero or more symbols from $\Sigma$. The set $\Sigma^{*}$ always contains the empty string $\lambda$. To exclude the empty string, we define

$$
\Sigma^{+}=\Sigma^{*}-\{\lambda\}
$$

While $\Sigma$ is finite by assumption, $\Sigma^{*}$ and $\Sigma^{+}$are always infinite because there is no limit on the length of the strings in these sets. A language is defined very generally as a subset of $\Sigma^{*}$. A string in a language $L$ will be called a sentence of $L$. This definition is quite broad; any set of strings on an alphabet $\Sigma$ can be considered as a language. Later we will study methods by which specific languages can be defined and described; this will enable us to give some structure to this rather broad concept. For the moment, though, we shall just look at a few specific examples.

## 2 THREE BASIC CONCEPTS: Languages

If $w$ is a string, then $w^{n}$ stands for the string obtained by repeating $w n$ times. As a special case, we define
for all strings $w$.
If $\Sigma$ is an alphabet, then we use $\Sigma^{*}$ to denote the set of strings obtained by concatenating zero or more symbols from $\Sigma$. The set $\Sigma^{*}$ always contains the empty string $\lambda$. To exclude the empty string, we define

While $\Sigma$ is finite by assumption, $\Sigma^{*}$ and $\Sigma^{+}$are always infinite because there is no limit on the length of the strings in these sets. A language is defined very generally as a subset of $\Sigma^{*}$. A string in a language $L$ will be called a sentence of $L$. This definition is quite broad; any set of strings on an alphabet $\Sigma$ can be considered as a language. Later we will study methods by which specific languages can be defined and described; this will enable us to give some structure to this rather broad concept. For the moment, though, we shall just look at a few specific examples.

## 2 THREE BASIC CONCEPTS: Languages

If $w$ is a string, then $w^{n}$ stands for the string obtained by repeating $w n$ times. As a special case, we define

```
for all strings w.
```




```
empty string }\lambda\mathrm{ . To exclude the empty string, we define
```

While $\Sigma$ is finite by assumption, $\Sigma^{*}$ and $\Sigma^{+}$are always infinite because there is no limit on the length of the strings in these sets. A language is defined very generally as a subset of $\Sigma^{*}$. A string in a language $L$ will be called a sentence of $L$. This definition is quite broad; any set of strings on an alphabet $\Sigma$ can be considered as a language. Later we will study methods by which specific languages can be defined and described; this will enable us to give some structure to this rather broad concept. For the moment, though, we shall just look at a few specific examples.

## 2 THREE BASIC CONCEPTS: Languages

If $w$ is a string, then $w^{n}$ stands for the string obtained by repeating $w n$ times. As a special case, we define

$$
w^{0}=\lambda,
$$

for all strings $w$.
If $\Sigma$ is an alphabet, then we use $\Sigma^{*}$ to denote the set of strings obtained by concatenating zero or more symbols from $\Sigma$. The set $\Sigma^{*}$ always contains the empty string $\lambda$. To exclude the empty string, we define

While $\Sigma$ is finite by assumption, $\Sigma^{*}$ and $\Sigma^{+}$are always infinite because there is no limit on the length of the strings in these sets. A language is defined very generally as a subset of $\Sigma^{*}$. A string in a language $L$ will be called a sentence of $L$. This definition is quite broad; any set of strings on an alphabet $\Sigma$ can be considered as a language. Later we will study methods by which specific languages can be defined and described; this will enable us to give some structure to this rather broad concept. For the moment, though, we shall just look at a few specific examples.

## 2 THREE BASIC CONCEPTS: Languages

If $w$ is a string, then $w^{n}$ stands for the string obtained by repeating $w n$ times. As a special case, we define

$$
w^{0}=\lambda,
$$

for all strings $w$.
If $\Sigma$ is an alphabet, then we use $\Sigma^{*}$ to denote the set of strings obtained by concatenating zero or more symbols from $\Sigma$. The set $\Sigma^{*}$ always contains the empty string $\lambda$. To exclude the empty string, we define

While $\Sigma$ is finite by assumption, $\Sigma^{*}$ and $\Sigma^{+}$are always infinite because there is no limit on the length of the strings in these sets. A language is defined very generally as a subset of $\Sigma^{*}$. A string in a language $L$ will be called a sentence of $L$. This definition is quite broad; any set of strings on an alphabet $\Sigma$ can be considered as a language. Later we will study methods by which specific languages can be defined and described; this will enable us to give some structure to this rather broad concept. For the moment, though, we shall just look at a few specific examples.

## 2 THREE BASIC CONCEPTS: Languages

If $w$ is a string, then $w^{n}$ stands for the string obtained by repeating $w n$ times. As a special case, we define

$$
w^{0}=\lambda,
$$

for all strings $w$.
If $\Sigma$ is an alphabet, then we use $\Sigma^{*}$ to denote the set of strings obtained by concatenating zero or more symbols from $\Sigma$.
empty string $\lambda$. To exclude the empty string, we define

While $\Sigma$ is finite by assumption, $\Sigma^{*}$ and $\Sigma^{+}$are always infinite because there
is no limit on the length of the strings in these sets. A language is defined very generally as a subset of $\Sigma^{*}$. A string in a language $L$ will be called a sentence of $L$. This definition is quite broad; any set of strings on an alphabet $\Sigma$ can be considered as a language. Later we will study methods by which specific languages can be defined and described; this will enable us to give some structure to this rather broad concept. For the moment, though, we shall just look at a few specific examples.

## 2 THREE BASIC CONCEPTS: Languages

If $w$ is a string, then $w^{n}$ stands for the string obtained by repeating $w n$ times. As a special case, we define

$$
w^{0}=\lambda,
$$

for all strings $w$.
If $\Sigma$ is an alphabet, then we use $\Sigma^{*}$ to denote the set of strings obtained by concatenating zero or more symbols from $\Sigma$. The set $\Sigma^{*}$ always contains the empty string $\lambda$. To exclude the empty string, we define

> While $\Sigma$ is finite by assumption, $\Sigma^{*}$ and $\Sigma^{+}$are always infinite because there is no limit on the length of the strings in these sets. A language is defined very generally as a subset of $\Sigma^{*}$. A string in a language $L$ will be called a sentence of $L$. This definition is quite broad; any set of strings on an alphabet $\Sigma$ can be considered as a language. Later we will study methods by which specific languages can be defined and described; this will enable us to give some structure to this rather broad concept. For the moment, though, we shall just look at a few specific examples.

## 2 THREE BASIC CONCEPTS: Languages

If $w$ is a string, then $w^{n}$ stands for the string obtained by repeating $w n$ times. As a special case, we define

$$
w^{0}=\lambda,
$$

for all strings $w$.
If $\Sigma$ is an alphabet, then we use $\Sigma^{*}$ to denote the set of strings obtained by concatenating zero or more symbols from $\Sigma$. The set $\Sigma^{*}$ always contains the empty string $\lambda$. To exclude the empty string, we define

> While $\Sigma$ is finite by assumption, $\Sigma^{*}$ and $\Sigma^{+}$are always infinite because there is no limit on the length of the strings in these sets. A language is defined very generally as a subset of $\Sigma^{*}$. A string in a language $L$ will be called a sentence of $L$. This definition is quite broad; any set of strings on an alphabet $\Sigma$ can be considered as a language. Later we will study methods by which specific languages can be defined and described; this will enable us to give some structure to this rather broad concept. For the moment, though, we shall just look at a few specific examples.

## 2 THREE BASIC CONCEPTS: Languages

If $w$ is a string, then $w^{n}$ stands for the string obtained by repeating $w n$ times. As a special case, we define

$$
w^{0}=\lambda,
$$

for all strings $w$.
If $\Sigma$ is an alphabet, then we use $\Sigma^{*}$ to denote the set of strings obtained by concatenating zero or more symbols from $\Sigma$. The set $\Sigma^{*}$ always contains the empty string $\lambda$. To exclude the empty string, we define

$$
\Sigma^{+}=\Sigma^{*}-\{\lambda\}
$$

> While $\Sigma$ is finite by assumption, $\Sigma^{*}$ and $\Sigma^{+}$are always infinite because there is no limit on the length of the strings in these sets. A language is defined very generally as a subset of $\Sigma^{*}$. A string in a language $L$ will be called a sentence of $L$. This definition is quite broad; any set of strings on an alphabet $\Sigma$ can be considered as a language. Later we will study methods by which specific languages can be defined and described; this will enable us to give some structure to this rather broad concept. For the moment, though, we shall just look at a few specific examples.

## 2 THREE BASIC CONCEPTS: Languages

If $w$ is a string, then $w^{n}$ stands for the string obtained by repeating $w n$ times. As a special case, we define

$$
w^{0}=\lambda,
$$

for all strings $w$.
If $\Sigma$ is an alphabet, then we use $\Sigma^{*}$ to denote the set of strings obtained by concatenating zero or more symbols from $\Sigma$. The set $\Sigma^{*}$ always contains the empty string $\lambda$. To exclude the empty string, we define

$$
\Sigma^{+}=\Sigma^{*}-\{\lambda\}
$$

While $\Sigma$ is finite by assumption, are always infinite because there
> generally as a subset of $\Sigma^{*}$. A string in a language $L$ will be called a sentence of $L$. This definition is quite broad; any set of strings on an alphabet $\Sigma$ can be considered as a language. Later we will study methods by which specific languages can be defined and described; this will enable us to give some structure to this rather broad concept. For the moment, though, we shall just look at a few specific examples.

## 2 THREE BASIC CONCEPTS: Languages

If $w$ is a string, then $w^{n}$ stands for the string obtained by repeating $w n$ times. As a special case, we define

$$
w^{0}=\lambda,
$$

for all strings $w$.
If $\Sigma$ is an alphabet, then we use $\Sigma^{*}$ to denote the set of strings obtained by concatenating zero or more symbols from $\Sigma$. The set $\Sigma^{*}$ always contains the empty string $\lambda$. To exclude the empty string, we define

$$
\Sigma^{+}=\Sigma^{*}-\{\lambda\}
$$

While $\Sigma$ is finite by assumption, $\Sigma^{*}$ and $\Sigma^{+}$are always infinite because there is no limit on the length of the strings in these sets.
of $L$. This definition is quite broad; any set of strings on an alphabet $\Sigma$ can be considered as a language. Later we will study methods by which specific languages can be defined and described; this will enable us to give some structure to this rather broad concept. For the moment, though, we shall just look at a few specific examples.

## 2 THREE BASIC CONCEPTS: Languages

If $w$ is a string, then $w^{n}$ stands for the string obtained by repeating $w n$ times. As a special case, we define

$$
w^{0}=\lambda,
$$

for all strings $w$.
If $\Sigma$ is an alphabet, then we use $\Sigma^{*}$ to denote the set of strings obtained by concatenating zero or more symbols from $\Sigma$. The set $\Sigma^{*}$ always contains the empty string $\lambda$. To exclude the empty string, we define

$$
\Sigma^{+}=\Sigma^{*}-\{\lambda\}
$$

While $\Sigma$ is finite by assumption, $\Sigma^{*}$ and $\Sigma^{+}$are always infinite because there is no limit on the length of the strings in these sets. A language is defined very generally as a subset of $\Sigma^{*}$.
considered as a language. Later we will study methods by which specific
languages can be defined and described; this will enable us to give some structure to this rather broad concept. For the moment, though, we shall just look at a few specific examples.

## 2 THREE BASIC CONCEPTS: Languages

If $w$ is a string, then $w^{n}$ stands for the string obtained by repeating $w n$ times. As a special case, we define

$$
w^{0}=\lambda,
$$

for all strings $w$.
If $\Sigma$ is an alphabet, then we use $\Sigma^{*}$ to denote the set of strings obtained by concatenating zero or more symbols from $\Sigma$. The set $\Sigma^{*}$ always contains the empty string $\lambda$. To exclude the empty string, we define

$$
\Sigma^{+}=\Sigma^{*}-\{\lambda\}
$$

While $\Sigma$ is finite by assumption, $\Sigma^{*}$ and $\Sigma^{+}$are always infinite because there is no limit on the length of the strings in these sets. A language is defined very generally as a subset of $\Sigma^{*}$. A string in a language $L$ will be called a sentence of $L$.
considered as a language. Later we will study methods by which specific
languages can be defined and described; this will enable us to give some structure to this rather broad concept. For the moment, though, we shall just look at a few specific examples.

## 2 THREE BASIC CONCEPTS: Languages

If $w$ is a string, then $w^{n}$ stands for the string obtained by repeating $w n$ times. As a special case, we define

$$
w^{0}=\lambda,
$$

for all strings $w$.
If $\Sigma$ is an alphabet, then we use $\Sigma^{*}$ to denote the set of strings obtained by concatenating zero or more symbols from $\Sigma$. The set $\Sigma^{*}$ always contains the empty string $\lambda$. To exclude the empty string, we define

$$
\Sigma^{+}=\Sigma^{*}-\{\lambda\} .
$$

While $\Sigma$ is finite by assumption, $\Sigma^{*}$ and $\Sigma^{+}$are always infinite because there is no limit on the length of the strings in these sets. A language is defined very generally as a subset of $\Sigma^{*}$. A string in a language $L$ will be called a sentence of $L$. This definition is quite broad;
considered as a language. Later we will study methods by which specific
languages can be defined and described; this will enable us to give some structure to this rather broad concept. For the moment, though, we shall just look at a few specific examples.

## 2 THREE BASIC CONCEPTS: Languages

If $w$ is a string, then $w^{n}$ stands for the string obtained by repeating $w n$ times. As a special case, we define

$$
w^{0}=\lambda,
$$

for all strings $w$.
If $\Sigma$ is an alphabet, then we use $\Sigma^{*}$ to denote the set of strings obtained by concatenating zero or more symbols from $\Sigma$. The set $\Sigma^{*}$ always contains the empty string $\lambda$. To exclude the empty string, we define

$$
\Sigma^{+}=\Sigma^{*}-\{\lambda\} .
$$

While $\Sigma$ is finite by assumption, $\Sigma^{*}$ and $\Sigma^{+}$are always infinite because there is no limit on the length of the strings in these sets. A language is defined very generally as a subset of $\Sigma^{*}$. A string in a language $L$ will be called a sentence of $L$. This definition is quite broad; any set of strings on an alphabet $\Sigma$ can be considered as a language.
languages can be defined and described; this will enable us to give some structure to this rather broad concept. For the moment, though, we shall just look at a few specific examples.

## 2 THREE BASIC CONCEPTS: Languages

If $w$ is a string, then $w^{n}$ stands for the string obtained by repeating $w n$ times. As a special case, we define

$$
w^{0}=\lambda,
$$

for all strings $w$.
If $\Sigma$ is an alphabet, then we use $\Sigma^{*}$ to denote the set of strings obtained by concatenating zero or more symbols from $\Sigma$. The set $\Sigma^{*}$ always contains the empty string $\lambda$. To exclude the empty string, we define

$$
\Sigma^{+}=\Sigma^{*}-\{\lambda\}
$$

While $\Sigma$ is finite by assumption, $\Sigma^{*}$ and $\Sigma^{+}$are always infinite because there is no limit on the length of the strings in these sets. A language is defined very generally as a subset of $\Sigma^{*}$. A string in a language $L$ will be called a sentence of $L$. This definition is quite broad; any set of strings on an alphabet $\Sigma$ can be considered as a language. Later we will study methods by which specific languages can be defined and described;
look at a few specific examples

## 2 THREE BASIC CONCEPTS: Languages

If $w$ is a string, then $w^{n}$ stands for the string obtained by repeating $w n$ times. As a special case, we define

$$
w^{0}=\lambda,
$$

for all strings $w$.
If $\Sigma$ is an alphabet, then we use $\Sigma^{*}$ to denote the set of strings obtained by concatenating zero or more symbols from $\Sigma$. The set $\Sigma^{*}$ always contains the empty string $\lambda$. To exclude the empty string, we define

$$
\Sigma^{+}=\Sigma^{*}-\{\lambda\} .
$$

While $\Sigma$ is finite by assumption, $\Sigma^{*}$ and $\Sigma^{+}$are always infinite because there is no limit on the length of the strings in these sets. A language is defined very generally as a subset of $\Sigma^{*}$. A string in a language $L$ will be called a sentence of $L$. This definition is quite broad; any set of strings on an alphabet $\Sigma$ can be considered as a language. Later we will study methods by which specific languages can be defined and described; this will enable us to give some structure to this rather broad concept.

## 2 THREE BASIC CONCEPTS: Languages

If $w$ is a string, then $w^{n}$ stands for the string obtained by repeating $w n$ times. As a special case, we define

$$
w^{0}=\lambda,
$$

for all strings $w$.
If $\Sigma$ is an alphabet, then we use $\Sigma^{*}$ to denote the set of strings obtained by concatenating zero or more symbols from $\Sigma$. The set $\Sigma^{*}$ always contains the empty string $\lambda$. To exclude the empty string, we define

$$
\Sigma^{+}=\Sigma^{*}-\{\lambda\} .
$$

While $\Sigma$ is finite by assumption, $\Sigma^{*}$ and $\Sigma^{+}$are always infinite because there is no limit on the length of the strings in these sets. A language is defined very generally as a subset of $\Sigma^{*}$. A string in a language $L$ will be called a sentence of $L$. This definition is quite broad; any set of strings on an alphabet $\Sigma$ can be considered as a language. Later we will study methods by which specific languages can be defined and described; this will enable us to give some structure to this rather broad concept. For the moment, though, we shall just look at a few specific examples.

## 2 THREE BASIC CONCEPTS: Languages

## Example 1.9

## Example 1.9

Let $\Sigma=\{a, b\}$. Then

$$
\Sigma^{*}=\{\lambda, a, b, a a, a b, b a, b b, a a a, a a b, \ldots\}
$$

The set

$$
\{a, a a, a a b\}
$$

is a language on $\Sigma$. Since it has a finite number of sentences, we call it a finite language. The set

$$
L=\left\{a^{n} b^{n}: n \geq 0\right\}
$$

is also a language on $\Sigma$. The strings $a a b b$ and $a a a a b b b b$ are in the language $L$, but the string $a b b$ is not in $L$. This language is infinite. Most interesting languages are infinite.

## Example 1.9

Let $\Sigma=\{a, b\}$. Then

$$
\Sigma^{*}=\{\lambda, a, b, a a, a b, b a, b b, a a a, a a b, \ldots\} .
$$

The set

$$
\{a, a a, a a b\}
$$

is a language on $\Sigma$. Since it has a finite number of sentences, we call it a finite language. The set

$$
L=\left\{a^{n} b^{n}: n \geq 0\right\}
$$

is also a language on $\Sigma$. The strings $a a b b$ and $a a a a b b b b$ are in the language $L$, but the string $a b b$ is not in $L$. This language is infinite. Most interesting languages are infinite.

## Example 1.9

Let $\Sigma=\{a, b\}$. Then

$$
\Sigma^{*}=\{\lambda, a, b, a a, a b, b a, b b, a a a, a a b, \ldots\} .
$$

The set

$$
\{a, a a, a a b\}
$$

is a language on $\Sigma$. Since it has a finite number of sentences, we call it a finite language. The set

$$
L=\left\{a^{n} b^{n}: n \geq 0\right\}
$$

is also a language on $\Sigma$. The strings $a a b b$ and $a a a a b b b b$ are in the language $L$, but the string $a b b$ is not in $L$. This language is infinite. Most interesting languages are infinite.

## Example 1.9

Let $\Sigma=\{a, b\}$. Then

$$
\Sigma^{*}=\{\lambda, a, b, a a, a b, b a, b b, a a a, a a b, \ldots\}
$$

The set
$\{a, a a, a a b\}$
is a language on $\Sigma$. Since it has a finite number of sentences, we call it a finite language. The set

$$
L=\left\{a^{n} b^{n}: n \geq 0\right\}
$$

is also a language on $\Sigma$. The strings $a a b b$ and $a a a a b b b b$ are in the language $L$, but the string $a b b$ is not in $L$. This language is infinite. Most interesting languages are infinite.

## Example 1.9

Let $\Sigma=\{a, b\}$. Then

$$
\Sigma^{*}=\{\lambda, a, b, a a, a b, b a, b b, a a a, a a b, \ldots\}
$$

The set
$\square$
$\{a, a a, a a b\}$
is a language on $\Sigma$. Since it has a finite number of sentences, we call it a finite
language. The set
$L=\left\{a^{n} b^{n}: n \geq 0\right\}$
is also a language on $\Sigma$. The strings $a a b b$ and aaaabbbb are in the language $L$,
but the string $a b b$ is not in $L$. This language is infinite. Most interesting
languages are infinite.

## Example 1.9

Let $\Sigma=\{a, b\}$. Then

$$
\Sigma^{*}=\{\lambda, a, b, a a, a b, b a, b b, a a a, a a b, \ldots\}
$$

The set

$$
\{a, a a, a a b\}
$$

is a language on $\Sigma$. Since it has a finite number of sentences, we call it a finite language. The set

$$
L=\left\{a^{n} b^{n}: n \geq 0\right\}
$$

is also a language on $\Sigma$. The strings $a a b b$ and $a a a a b b b b$ are in the language $L$, but the string $a b b$ is not in $L$. This language is infinite. Most interesting languages are infinite.

## Example 1.9

Let $\Sigma=\{a, b\}$. Then

$$
\Sigma^{*}=\{\lambda, a, b, a a, a b, b a, b b, a a a, a a b, \ldots\}
$$

The set

$$
\{a, a a, a a b\}
$$

is a language on $\Sigma$. Since it has a finite number of sentences, we call it a finite language. The set
is also a language on $\Sigma$. The strings $a a b b$ and $a a a a b b b b$ are in the language $L$, but the string $a b b$ is not in $L$. This language is infinite. Most interesting languages are infinite.

## 2 THREE BASIC CONCEPTS: Languages

## Example 1.9

Let $\Sigma=\{a, b\}$. Then

$$
\Sigma^{*}=\{\lambda, a, b, a a, a b, b a, b b, a a a, a a b, \ldots\}
$$

The set

$$
\{a, a a, a a b\}
$$

is a language on $\Sigma$. Since it has a finite number of sentences, we call it a finite language. The set
is also a language on $\Sigma$. The strings $a a b b$ and $a a a a b b b b$ are in the language $L$, but the string $a b b$ is not in $L$. This language is infinite. Most interesting languages are infinite.

## 2 THREE BASIC CONCEPTS: Languages

## Example 1.9

Let $\Sigma=\{a, b\}$. Then

$$
\Sigma^{*}=\{\lambda, a, b, a a, a b, b a, b b, a a a, a a b, \ldots\}
$$

The set

$$
\{a, a a, a a b\}
$$

is a language on $\Sigma$. Since it has a finite number of sentences, we call it a finite language. The set
is also a language on $\Sigma$. The strings $a a b b$ and $a a a a b b b b$ are in the language $L$, but the string $a b b$ is not in $L$. This language is infinite. Most interesting languages are infinite.

## 2 THREE BASIC CONCEPTS: Languages

## Example 1.9

Let $\Sigma=\{a, b\}$. Then

$$
\Sigma^{*}=\{\lambda, a, b, a a, a b, b a, b b, a a a, a a b, \ldots\}
$$

The set

$$
\{a, a a, a a b\}
$$

is a language on $\Sigma$. Since it has a finite number of sentences, we call it a finite language. The set

$$
L=\left\{a^{n} b^{n}: n \geq 0\right\}
$$

is also a language on $\Sigma$. The strings $a a b b$ and $a a a a b b b b$ are in the language $L$, but the string $a b b$ is not in $L$. This language is infinite. Most interesting languages are infinite.

## 2 THREE BASIC CONCEPTS: Languages

## Example 1.9

Let $\Sigma=\{a, b\}$. Then

$$
\Sigma^{*}=\{\lambda, a, b, a a, a b, b a, b b, a a a, a a b, \ldots\}
$$

The set

$$
\{a, a a, a a b\}
$$

is a language on $\Sigma$. Since it has a finite number of sentences, we call it a finite language. The set

$$
L=\left\{a^{n} b^{n}: n \geq 0\right\}
$$

is also a language on $\Sigma$. The strings $a a b b$ and aaaabbbb are in the language $L$, but the string $a b b$ is not in $L$. This language is infinite. Most interesting languages are infinite.

## 2 THREE BASIC CONCEPTS: Languages

## Example 1.9

Let $\Sigma=\{a, b\}$. Then

$$
\Sigma^{*}=\{\lambda, a, b, a a, a b, b a, b b, a a a, a a b, \ldots\}
$$

The set

$$
\{a, a a, a a b\}
$$

is a language on $\Sigma$. Since it has a finite number of sentences, we call it a finite language. The set

$$
L=\left\{a^{n} b^{n}: n \geq 0\right\}
$$

is also a language on $\Sigma$. The strings $a a b b$ and $a a a a b b b b$ are in the language $L$, but the string $a b b$ is not in $L$. This language is infinite. Most interesting languages are infinite.

## 2 THREE BASIC CONCEPTS: Languages

## Example 1.9

Let $\Sigma=\{a, b\}$. Then

$$
\Sigma^{*}=\{\lambda, a, b, a a, a b, b a, b b, a a a, a a b, \ldots\}
$$

The set

$$
\{a, a a, a a b\}
$$

is a language on $\Sigma$. Since it has a finite number of sentences, we call it a finite language. The set

$$
L=\left\{a^{n} b^{n}: n \geq 0\right\}
$$

is also a language on $\Sigma$. The strings $a a b b$ and $a a a a b b b b$ are in the language $L$, but the string $a b b$ is not in $L$.
languages are infinite.

## 2 THREE BASIC CONCEPTS: Languages

## Example 1.9

Let $\Sigma=\{a, b\}$. Then

$$
\Sigma^{*}=\{\lambda, a, b, a a, a b, b a, b b, a a a, a a b, \ldots\}
$$

The set

$$
\{a, a a, a a b\}
$$

is a language on $\Sigma$. Since it has a finite number of sentences, we call it a finite language. The set

$$
L=\left\{a^{n} b^{n}: n \geq 0\right\}
$$

is also a language on $\Sigma$. The strings $a a b b$ and $a a a a b b b b$ are in the language $L$, but the string $a b b$ is not in $L$. This language is infinite. Most interesting

## 2 THREE BASIC CONCEPTS: Languages

## Example 1.9

Let $\Sigma=\{a, b\}$. Then

$$
\Sigma^{*}=\{\lambda, a, b, a a, a b, b a, b b, a a a, a a b, \ldots\}
$$

The set

$$
\{a, a a, a a b\}
$$

is a language on $\Sigma$. Since it has a finite number of sentences, we call it a finite language. The set

$$
L=\left\{a^{n} b^{n}: n \geq 0\right\}
$$

is also a language on $\Sigma$. The strings $a a b b$ and $a a a a b b b b$ are in the language $L$, but the string $a b b$ is not in $L$. This language is infinite. Most interesting languages are infinite.

## 2 THREE BASIC CONCEPTS: Languages

## Example 1.9

Let $\Sigma=\{a, b\}$. Then

$$
\Sigma^{*}=\{\lambda, a, b, a a, a b, b a, b b, a a a, a a b, \ldots\}
$$

The set

$$
\{a, a a, a a b\}
$$

is a language on $\Sigma$. Since it has a finite number of sentences, we call it a finite language. The set

$$
L=\left\{a^{n} b^{n}: n \geq 0\right\}
$$

is also a language on $\Sigma$. The strings $a a b b$ and $a a a a b b b b$ are in the language $L$, but the string $a b b$ is not in $L$. This language is infinite. Most interesting languages are infinite.

## 2 THREE BASIC CONCEPTS: Languages

Since languages are sets, the union, intersection, and difference of two languages are immediately defined. The complement of a language is defined with respect to $\Sigma^{*}$; that is, the complement of $L$ is

$$
\bar{L}=\Sigma^{*}-L .
$$

The reverse of a language is the set of all string reversals, that is,

$$
L^{R}=\left\{w^{n}: w \in I\right\}
$$

The concatenation of two languages $L_{1}$ and $L_{2}$ is the set of all strings obtained by concatenating any element of $L_{1}$ with any element of $L_{2}$; specifically,

$$
L_{1} L_{2}=\left\{x y: x \in L_{1}, y \in L_{2}\right\}
$$

We define $L^{n}$ as $L$ concatenated with itself $n$ times, with the special cases

$$
L^{0}=\{\lambda\}
$$

and

$$
L^{1}=L
$$

or every language $L$.

## 2 THREE BASIC CONCEPTS: Languages

Since languages are sets, the union, intersection, and difference of two languages are immediately defined.
with respect to $\Sigma^{*}$; that is, the complement of $L$ is

and

or every language $L$.

## 2 THREE BASIC CONCEPTS: Languages

Since languages are sets, the union, intersection, and difference of two languages are immediately defined. The complement of a language is defined with respect to $\Sigma^{*}$;



## 2 THREE BASIC CONCEPTS: Languages

Since languages are sets, the union, intersection, and difference of two languages are immediately defined. The complement of a language is defined with respect to $\Sigma^{*}$; that is, the complement of $L$ is



## 2 THREE BASIC CONCEPTS: Languages

Since languages are sets, the union, intersection, and difference of two languages are immediately defined. The complement of a language is defined with respect to $\Sigma^{*}$; that is, the complement of $L$ is

$$
\bar{L}=\Sigma^{*}-L
$$

The reverse of a language is the set of all string reversals, that is,

The concatenation of two languages $L_{1}$ and $L_{2}$ is the set of all strings obtained by concatenating any element of $L_{1}$ with any element of $L_{2}$; specifically,

We define $L^{n}$ as $L$ concatenated with itself $n$ times, with the special cases

or every language $L$.

## 2 THREE BASIC CONCEPTS: Languages

Since languages are sets, the union, intersection, and difference of two languages are immediately defined. The complement of a language is defined with respect to $\Sigma^{*}$; that is, the complement of $L$ is

$$
\bar{L}=\Sigma^{*}-L
$$

The reverse of a language is the set of all string reversals, that is,


We define $L^{n}$ as $L$ concatenated with itself $n$ times, with the special cases

## 2 THREE BASIC CONCEPTS: Languages

Since languages are sets, the union, intersection, and difference of two languages are immediately defined. The complement of a language is defined with respect to $\Sigma^{*}$; that is, the complement of $L$ is

$$
\bar{L}=\Sigma^{*}-L
$$

The reverse of a language is the set of all string reversals, that is,

$$
L^{R}=\left\{w^{R}: w \in L\right\}
$$

$\square$

We define $L^{n}$ as $L$ concatenated with itself $n$ times, with the special cases

## 2 THREE BASIC CONCEPTS: Languages

Since languages are sets, the union, intersection, and difference of two languages are immediately defined. The complement of a language is defined with respect to $\Sigma^{*}$; that is, the complement of $L$ is

$$
\bar{L}=\Sigma^{*}-L
$$

The reverse of a language is the set of all string reversals, that is,

$$
L^{R}=\left\{w^{R}: w \in L\right\}
$$

The concatenation of two languages $L_{1}$ and $L_{2}$ $\qquad$ by concatenating any element of $L_{1}$ with any element of $L_{2}$; specifically,

We define $L^{n}$ as $L$ concatenated with itself $n$ times, with the special cases

## 2 THREE BASIC CONCEPTS: Languages

Since languages are sets, the union, intersection, and difference of two languages are immediately defined. The complement of a language is defined with respect to $\Sigma^{*}$; that is, the complement of $L$ is

$$
\bar{L}=\Sigma^{*}-L
$$

The reverse of a language is the set of all string reversals, that is,

$$
L^{R}=\left\{w^{R}: w \in L\right\}
$$

The concatenation of two languages $L_{1}$ and $L_{2}$ is the set of all strings obtained by concatenating any element of $L_{1}$ with any element of $L_{2}$; specifically,

We define $L^{n}$ as $L$ concatenated with itself $n$ times, with the special cases
and
or every language $L$

## 2 THREE BASIC CONCEPTS: Languages

Since languages are sets, the union, intersection, and difference of two languages are immediately defined. The complement of a language is defined with respect to $\Sigma^{*}$; that is, the complement of $L$ is

$$
\bar{L}=\Sigma^{*}-L
$$

The reverse of a language is the set of all string reversals, that is,

$$
L^{R}=\left\{w^{R}: w \in L\right\}
$$

The concatenation of two languages $L_{1}$ and $L_{2}$ is the set of all strings obtained by concatenating any element of $L_{1}$ with any element of $L_{2}$; specifically,

We define $L^{n}$ as $L$ concatenated with itself $n$ times, with the special cases
and
or every language $L$

## 2 THREE BASIC CONCEPTS: Languages

Since languages are sets, the union, intersection, and difference of two languages are immediately defined. The complement of a language is defined with respect to $\Sigma^{*}$; that is, the complement of $L$ is

$$
\bar{L}=\Sigma^{*}-L
$$

The reverse of a language is the set of all string reversals, that is,

$$
L^{R}=\left\{w^{R}: w \in L\right\}
$$

The concatenation of two languages $L_{1}$ and $L_{2}$ is the set of all strings obtained by concatenating any element of $L_{1}$ with any element of $L_{2}$; specifically,

$$
L_{1} L_{2}=\left\{x y: x \in L_{1}, y \in L_{2}\right\}
$$

We define $L^{n}$ as $L$ concatenated with itself $n$ times, with the special cases
and
or every language $L$

## 2 THREE BASIC CONCEPTS: Languages

Since languages are sets, the union, intersection, and difference of two languages are immediately defined. The complement of a language is defined with respect to $\Sigma^{*}$; that is, the complement of $L$ is

$$
\bar{L}=\Sigma^{*}-L
$$

The reverse of a language is the set of all string reversals, that is,

$$
L^{R}=\left\{w^{R}: w \in L\right\}
$$

The concatenation of two languages $L_{1}$ and $L_{2}$ is the set of all strings obtained by concatenating any element of $L_{1}$ with any element of $L_{2}$; specifically,

$$
L_{1} L_{2}=\left\{x y: x \in L_{1}, y \in L_{2}\right\}
$$

We define $L^{n}$ as $L$ concatenated with itself $n$ times, with the special cases

## 2 THREE BASIC CONCEPTS: Languages

Since languages are sets, the union, intersection, and difference of two languages are immediately defined. The complement of a language is defined with respect to $\Sigma^{*}$; that is, the complement of $L$ is

$$
\bar{L}=\Sigma^{*}-L
$$

The reverse of a language is the set of all string reversals, that is,

$$
L^{R}=\left\{w^{R}: w \in L\right\}
$$

The concatenation of two languages $L_{1}$ and $L_{2}$ is the set of all strings obtained by concatenating any element of $L_{1}$ with any element of $L_{2}$; specifically,

$$
L_{1} L_{2}=\left\{x y: x \in L_{1}, y \in L_{2}\right\}
$$

We define $L^{n}$ as $L$ concatenated with itself $n$ times, with the special cases

$$
L^{0}=\{\lambda\}
$$

and
or every language $L$.

## 2 THREE BASIC CONCEPTS: Languages

Since languages are sets, the union, intersection, and difference of two languages are immediately defined. The complement of a language is defined with respect to $\Sigma^{*}$; that is, the complement of $L$ is

$$
\bar{L}=\Sigma^{*}-L
$$

The reverse of a language is the set of all string reversals, that is,

$$
L^{R}=\left\{w^{R}: w \in L\right\}
$$

The concatenation of two languages $L_{1}$ and $L_{2}$ is the set of all strings obtained by concatenating any element of $L_{1}$ with any element of $L_{2}$; specifically,

$$
L_{1} L_{2}=\left\{x y: x \in L_{1}, y \in L_{2}\right\}
$$

We define $L^{n}$ as $L$ concatenated with itself $n$ times, with the special cases

$$
L^{0}=\{\lambda\}
$$

and
or every language $L$.

## 2 THREE BASIC CONCEPTS: Languages

Since languages are sets, the union, intersection, and difference of two languages are immediately defined. The complement of a language is defined with respect to $\Sigma^{*}$; that is, the complement of $L$ is

$$
\bar{L}=\Sigma^{*}-L
$$

The reverse of a language is the set of all string reversals, that is,

$$
L^{R}=\left\{w^{R}: w \in L\right\}
$$

The concatenation of two languages $L_{1}$ and $L_{2}$ is the set of all strings obtained by concatenating any element of $L_{1}$ with any element of $L_{2}$; specifically,

$$
L_{1} L_{2}=\left\{x y: x \in L_{1}, y \in L_{2}\right\}
$$

We define $L^{n}$ as $L$ concatenated with itself $n$ times, with the special cases

$$
L^{0}=\{\lambda\}
$$

and

$$
L^{1}=L
$$

or every language $L$.

## 2 THREE BASIC CONCEPTS: Languages

Since languages are sets, the union, intersection, and difference of two languages are immediately defined. The complement of a language is defined with respect to $\Sigma^{*}$; that is, the complement of $L$ is

$$
\bar{L}=\Sigma^{*}-L
$$

The reverse of a language is the set of all string reversals, that is,

$$
L^{R}=\left\{w^{R}: w \in L\right\}
$$

The concatenation of two languages $L_{1}$ and $L_{2}$ is the set of all strings obtained by concatenating any element of $L_{1}$ with any element of $L_{2}$; specifically,

$$
L_{1} L_{2}=\left\{x y: x \in L_{1}, y \in L_{2}\right\}
$$

We define $L^{n}$ as $L$ concatenated with itself $n$ times, with the special cases

$$
L^{0}=\{\lambda\}
$$

and

$$
L^{1}=L
$$

or every language $L$.

## 2 THREE BASIC CONCEPTS: Languages

Finally, we define the star-closure of a language $L$ as

and the positive closure of $L$ as


## Example 1.10

## 2 THREE BASIC CONCEPTS: Languages

Finally, we define the star-closure of a language $L$ as
and the positive closure of $L$ as

## Example 1.10

## 2 THREE BASIC CONCEPTS: Languages

Finally, we define the star-closure of a language $L$ as

$$
L^{*}=L^{0} \cup L_{1} \cup L_{2} \cup \cdots
$$

and the positive closure of $L$ as

## Example 1.10

## 2 THREE BASIC CONCEPTS: Languages

Finally, we define the star-closure of a language $L$ as

$$
L^{*}=L^{0} \cup L_{1} \cup L_{2} \cup \ldots
$$

and the positive closure of $L$ as

## Example 1.10

## 2 THREE BASIC CONCEPTS: Languages

Finally, we define the star-closure of a language $L$ as

$$
L^{*}=L^{0} \cup L_{1} \cup L_{2} \cup \ldots
$$

and the positive closure of $L$ as

$$
L^{+}=L^{1} \cup L_{2} \cup L_{3} \cup \cdots
$$

## Example 1.10

## 2 THREE BASIC CONCEPTS: Languages

Finally, we define the star-closure of a language $L$ as

$$
L^{*}=L^{0} \cup L_{1} \cup L_{2} \cup \ldots
$$

and the positive closure of $L$ as

$$
L^{+}=L^{1} \cup L_{2} \cup L_{3} \cup \cdots
$$

## Example 1.10

$$
L=\left\{a^{n} b^{n}: n \geq 0\right\},
$$

then

$$
L^{2}=\left\{a^{n} b^{n} a^{m} b^{m}: n \geq 0, m \geq 0\right\} .
$$

Note that $n$ and $m$ in the above are unrelated; the string aabbaaabbb is in $L^{2}$. The reverse of $L$ is easily described in set notation as

but it is considerably harder to describe $\bar{L}$ or $L^{*}$ this way. A few tries will quickly convince you of the limitation of set notation for the specification of complicated languages.

## 2 THREE BASIC CONCEPTS: Languages

Finally, we define the star-closure of a language $L$ as

$$
L^{*}=L^{0} \cup L_{1} \cup L_{2} \cup \ldots
$$

and the positive closure of $L$ as

$$
L^{+}=L^{1} \cup L_{2} \cup L_{3} \cup \cdots
$$

## Example 1.10

If

$$
L=\left\{a^{n} b^{n}: n \geq 0\right\},
$$

then

$$
L^{2}=\left\{a^{n} b^{n} a^{m} b^{m}: n \geq 0, m \geq 0\right\} .
$$

Note that $n$ and $m$ in the above are unrelated; the string aabbaaabbb is in $L^{2}$. The reverse of $L$ is easily described in set notation as

but it is considerably harder to describe $\bar{L}$ or $L^{*}$ this way. A few tries will quickly convince you of the limitation of set notation for the specification of complicated languages.

## 2 THREE BASIC CONCEPTS: Languages

Finally, we define the star-closure of a language $L$ as

$$
L^{*}=L^{0} \cup L_{1} \cup L_{2} \cup \ldots
$$

and the positive closure of $L$ as

$$
L^{+}=L^{1} \cup L_{2} \cup L_{3} \cup \cdots
$$

## Example 1.10

If

$$
L=\left\{a^{n} b^{n}: n \geq 0\right\}
$$

then

Note that $n$ and $m$ in the above are unrelated; the string aabbaaabbb is in $L^{2}$ The reverse of $L$ is easily described in set notation as

but it is considerably harder to describe $\bar{L}$ or $L^{*}$ this way. A few tries will quickly convince you of the limitation of set notation for the specification of complicated languages.

## 2 THREE BASIC CONCEPTS: Languages

Finally, we define the star-closure of a language $L$ as

$$
L^{*}=L^{0} \cup L_{1} \cup L_{2} \cup \ldots
$$

and the positive closure of $L$ as

$$
L^{+}=L^{1} \cup L_{2} \cup L_{3} \cup \cdots
$$

## Example 1.10

If

$$
L=\left\{a^{n} b^{n}: n \geq 0\right\}
$$

then

Note that $n$ and $m$ in the above are unrelated; the string aabbaaabbb is in $L^{2}$. The reverse of $L$ is easily described in set notation as
but it is considerably harder to describe $\bar{L}$ or $L^{*}$ this way. A few tries will quickly convince you of the limitation of set notation for the specification of complicated languages.

## 2 THREE BASIC CONCEPTS: Languages

Finally, we define the star-closure of a language $L$ as

$$
L^{*}=L^{0} \cup L_{1} \cup L_{2} \cup \ldots
$$

and the positive closure of $L$ as

$$
L^{+}=L^{1} \cup L_{2} \cup L_{3} \cup \cdots
$$

## Example 1.10

If

$$
L=\left\{a^{n} b^{n}: n \geq 0\right\}
$$

then

$$
L^{2}=\left\{a^{n} b^{n} a^{m} b^{m}: n \geq 0, m \geq 0\right\}
$$

Note that $n$ and $m$ in the above are unrelated; the string aabbaaabbb is in $L^{2}$. The reverse of $L$ is easily described in set notation as
but it is considerably harder to describe $\bar{L}$ or $L^{*}$ this way. A few tries will quickly convince you of the limitation of set notation for the specification of complicated languages.

## 2 THREE BASIC CONCEPTS: Languages

Finally, we define the star-closure of a language $L$ as

$$
L^{*}=L^{0} \cup L_{1} \cup L_{2} \cup \ldots
$$

and the positive closure of $L$ as

$$
L^{+}=L^{1} \cup L_{2} \cup L_{3} \cup \cdots
$$

## Example 1.10

If

$$
L=\left\{a^{n} b^{n}: n \geq 0\right\}
$$

then

$$
L^{2}=\left\{a^{n} b^{n} a^{m} b^{m}: n \geq 0, m \geq 0\right\}
$$

Note that $n$ and $m$ in the above are unrelated;
The reverse of $L$ is easily described in set notation as
but it is considerably harder to describe $\bar{L}$ or $L^{*}$ this way. A few tries will quickly convince you of the limitation of set notation for the specification of complicated languages.

## 2 THREE BASIC CONCEPTS: Languages

Finally, we define the star-closure of a language $L$ as

$$
L^{*}=L^{0} \cup L_{1} \cup L_{2} \cup \ldots
$$

and the positive closure of $L$ as

$$
L^{+}=L^{1} \cup L_{2} \cup L_{3} \cup \cdots
$$

## Example 1.10

If

$$
L=\left\{a^{n} b^{n}: n \geq 0\right\}
$$

then

$$
L^{2}=\left\{a^{n} b^{n} a^{m} b^{m}: n \geq 0, m \geq 0\right\}
$$

Note that $n$ and $m$ in the above are unrelated; the string aabbaaabbb is in $L^{2}$.
The reverse of $L$ is easily described in set notation as
but it is considerably harder to describe $\bar{L}$ or $L^{*}$ this way. A few tries will quickly convince you of the limitation of set notation for the specification of complicated languages.

## 2 THREE BASIC CONCEPTS: Languages

Finally, we define the star-closure of a language $L$ as

$$
L^{*}=L^{0} \cup L_{1} \cup L_{2} \cup \ldots
$$

and the positive closure of $L$ as

$$
L^{+}=L^{1} \cup L_{2} \cup L_{3} \cup \cdots
$$

## Example 1.10

If

$$
L=\left\{a^{n} b^{n}: n \geq 0\right\}
$$

then

$$
L^{2}=\left\{a^{n} b^{n} a^{m} b^{m}: n \geq 0, m \geq 0\right\}
$$

Note that $n$ and $m$ in the above are unrelated; the string aabbaaabbb is in $L^{2}$. The reverse of $L$ is easily described in set notation as
but it is considerably harder to describe $\bar{L}$ or $L^{*}$ this way. A few tries will quickly convince you of the limitation of set notation for the specification of complicated languages.

## 2 THREE BASIC CONCEPTS: Languages

Finally, we define the star-closure of a language $L$ as

$$
L^{*}=L^{0} \cup L_{1} \cup L_{2} \cup \ldots
$$

and the positive closure of $L$ as

$$
L^{+}=L^{1} \cup L_{2} \cup L_{3} \cup \cdots
$$

## Example 1.10

If

$$
L=\left\{a^{n} b^{n}: n \geq 0\right\}
$$

then

$$
L^{2}=\left\{a^{n} b^{n} a^{m} b^{m}: n \geq 0, m \geq 0\right\}
$$

Note that $n$ and $m$ in the above are unrelated; the string aabbaaabbb is in $L^{2}$. The reverse of $L$ is easily described in set notation as

$$
L^{R}=\left\{b^{n} a^{n}: n \geq 0\right\}
$$

but it is considerably harder to describe $\bar{L}$ or $L^{*}$ this way. A few tries will quickly convince you of the limitation of set notation for the specification of complicated languages.

## 2 THREE BASIC CONCEPTS: Languages

Finally, we define the star-closure of a language $L$ as

$$
L^{*}=L^{0} \cup L_{1} \cup L_{2} \cup \ldots
$$

and the positive closure of $L$ as

$$
L^{+}=L^{1} \cup L_{2} \cup L_{3} \cup \cdots
$$

## Example 1.10

If

$$
L=\left\{a^{n} b^{n}: n \geq 0\right\}
$$

then

$$
L^{2}=\left\{a^{n} b^{n} a^{m} b^{m}: n \geq 0, m \geq 0\right\}
$$

Note that $n$ and $m$ in the above are unrelated; the string aabbaaabbb is in $L^{2}$. The reverse of $L$ is easily described in set notation as

$$
L^{R}=\left\{b^{n} a^{n}: n \geq 0\right\}
$$

but it is considerably harder to describe $\bar{L}$ or $L^{*}$ this way.

## 2 THREE BASIC CONCEPTS: Languages

Finally, we define the star-closure of a language $L$ as

$$
L^{*}=L^{0} \cup L_{1} \cup L_{2} \cup \ldots
$$

and the positive closure of $L$ as

$$
L^{+}=L^{1} \cup L_{2} \cup L_{3} \cup \cdots
$$

## Example 1.10

If

$$
L=\left\{a^{n} b^{n}: n \geq 0\right\}
$$

then

$$
L^{2}=\left\{a^{n} b^{n} a^{m} b^{m}: n \geq 0, m \geq 0\right\}
$$

Note that $n$ and $m$ in the above are unrelated; the string aabbaaabbb is in $L^{2}$. The reverse of $L$ is easily described in set notation as

$$
L^{R}=\left\{b^{n} a^{n}: n \geq 0\right\}
$$

but it is considerably harder to describe $\bar{L}$ or $L^{*}$ this way. A few tries will quickly convince you of the limitation of set notation for the specification of complicated languages.

## 2 THREE BASIC CONCEPTS: Languages

Finally, we define the star-closure of a language $L$ as

$$
L^{*}=L^{0} \cup L_{1} \cup L_{2} \cup \ldots
$$

and the positive closure of $L$ as

$$
L^{+}=L^{1} \cup L_{2} \cup L_{3} \cup \cdots
$$

## Example 1.10

If

$$
L=\left\{a^{n} b^{n}: n \geq 0\right\}
$$

then

$$
L^{2}=\left\{a^{n} b^{n} a^{m} b^{m}: n \geq 0, m \geq 0\right\}
$$

Note that $n$ and $m$ in the above are unrelated; the string aabbaaabbb is in $L^{2}$. The reverse of $L$ is easily described in set notation as

$$
L^{R}=\left\{b^{n} a^{n}: n \geq 0\right\}
$$

but it is considerably harder to describe $\bar{L}$ or $L^{*}$ this way. A few tries will quickly convince you of the limitation of set notation for the specification of complicated languages.

## 2 THREE BASIC CONCEPTS：Grammars

To study languages mathematically，we need a mechanism to describe them． Everyday language is imprecise and ambiguous，so informal descriptions in English are often inadequate．The set notation used in Examples 1.9 and 1.10 is more suitable，but limited．As we proceed we will learn about several language－definition mechanisms that are useful in different circumstances．Here we introduce a common and powerful one，the notion of a grammar．
A grammar for the English language tells us whether a particular sentence is well formed or not．A typical rule of English grammar is＂a sentence can consist of a noun phrase followed by a predicate＂．More concisely we write this as

$$
\langle\text { sentence }\rangle \rightarrow\langle\text { noun_phrase }\rangle\langle\text { predicate }\rangle,
$$

with the obvious interpretation．This is，of course，not enough to deal with actual sentences．We must now provide definitions for the newly introduced constructs 〈noun＿phrase〉 and 〈predicate）．If we do so by
$\langle$ noun＿phrase $\rangle \rightarrow\langle$ article $\rangle\langle$ noun $\rangle$,

$$
\langle\text { predicate }\rangle \rightarrow\langle\text { verb }\rangle,
$$

and if we associate the actual words＂a＂and＂the＂with 〈article〉，＂boy＂and ＂dog＂with 〈noun〉，and＂runs＂and＂walks＂with 〈verb〉，then the grammar tells us that the sentences＂a boy runs＂and＂the dog walks＂are properly formed．If we were to give a complete grammar，then in theory，every proper sentence could be explained this way．

## 2 THREE BASIC CONCEPTS：Grammars

To study languages mathematically，we need a mechanism to describe them．
Everyday language is imprecise and ambiguous，so informal descriptions in English are often inadequate．The set notation used in Examples 1.9 and 1.10 is more suitable，but limited．As we proceed we will learn about several language－definition mechanisms that are useful in different circumstances．Here we introduce a common and powerful one，the notion of a grammar．
A grammar for the English language tells us whether a particular sentence is well formed or not．A typical rule of English grammar is＂a sentence can consist of a noun phrase followed by a predicate＂．More concisely we write this as〈sentence〉 $\rightarrow$ 〈noun＿phrase〉〈predicate〉
with the obvious interpretation．This is，of course，not enough to deal with actual sentences．We must now provide definitions for the newly introduced constructs 〈noun＿phrase〉 and 〈predicate〉．If we do so by
$\langle$ noun phrase〉 $\rightarrow$ 〈article〉 $\langle$ noun $\rangle$ ，
$\langle$ predicate〉 $\rightarrow$ 〈verb〉
and if we associate the actual words＂a＂and＂the＂with 〈article〉，＂boy＂and ＂dog＂with 〈noun〉，and＂runs＂and＂walks＂with 〈verb〉，then the grammar tells us that the sentences＂a boy runs＂and＂the dog walks＂are properly formed．If we were to give a complete grammar，then in theory，every proper sentence could be explained this way．

## 2 THREE BASIC CONCEPTS：Grammars

To study languages mathematically，we need a mechanism to describe them． Everyday language is imprecise and ambiguous，so informal descriptions in English are often inadequate．
is more suitable，but limited．As we proceed we will learn about several
language－definition mechanisms that are useful in different circumstances．Here we introduce a common and powerful one，the notion of a grammar．
A grammar for the English language tells us whether a particular sentence is well formed or not．A typical rule of English grammar is＂a sentence can consist of a noun phrase followed by a predicate＂．More concisely we write this as

$\langle$ noun＿phrase $\rangle \rightarrow\langle$ article $\rangle\langle$ noun $\rangle$
｜predicate｜$\rightarrow$｜verb｜

> and if we associate the actual words "a" and "the" with 〈article〉, "boy" and "dog" with 〈noun〉, and "runs" and "walks" with 〈verb〉, then the grammar tells us that the sentences "a boy runs" and "the dog walks" are properly formed. If we were to give a complete grammar, then in theory, every proper sentence could be explained this way.

## 2 THREE BASIC CONCEPTS: Grammars

To study languages mathematically, we need a mechanism to describe them. Everyday language is imprecise and ambiguous, so informal descriptions in English are often inadequate. The set notation used in Examples 1.9 and 1.10 is more suitable, but limited.
language-definition mechanisms that are useful in different circumstances. Here we introduce a common and powerful one, the notion of a grammar.
A grammar for the English language tells us whether a particular sentence is well formed or not. A typical rule of English grammar is "a sentence can consist of a noun phrase followed by a predicate". More concisely we write this as

$\langle$ noun phrase $\rangle \rightarrow\langle$ article $\rangle\langle$ noun $\rangle$


## 2 THREE BASIC CONCEPTS：Grammars

To study languages mathematically，we need a mechanism to describe them． Everyday language is imprecise and ambiguous，so informal descriptions in English are often inadequate．The set notation used in Examples 1.9 and 1.10 is more suitable，but limited．As we proceed we will learn about several language－definition mechanisms that are useful in different circumstances．
A grammar for the English language tells us whether a particular sentence is well formed or not．A typical rule of English grammar is＂a sentence can consist of a noun phrase followed by a predicate＂．More concisely we write this as
> with the obvious interpretation．This is，of course，not enough to deal with actual sentences．We must now provide definitions for the newly introduced constructs 〈noun＿phrase〉 and 〈predicate〉．If we do so by


## 2 THREE BASIC CONCEPTS：Grammars

To study languages mathematically，we need a mechanism to describe them． Everyday language is imprecise and ambiguous，so informal descriptions in English are often inadequate．The set notation used in Examples 1.9 and 1.10 is more suitable，but limited．As we proceed we will learn about several language－definition mechanisms that are useful in different circumstances．Here we introduce a common and powerful one，the notion of a grammar．

of a noun phrase followed by a predicate＂．More concisely we write this as
> （＇sentence）$\rightarrow$＇notun phrase）／＇predicate〉，
with the obvious interpretation．This is，of course，not enough to deal with
actual sentences．We must now provide definitions for the newly introduced
constructs 〈noum phrase）and 〈predicate）．If we do so by

noun phrase $\rangle \rightarrow\langle$ article $\rangle\langle$ noun $\rangle$
$\qquad$

## 2 THREE BASIC CONCEPTS：Grammars

To study languages mathematically，we need a mechanism to describe them． Everyday language is imprecise and ambiguous，so informal descriptions in English are often inadequate．The set notation used in Examples 1.9 and 1.10 is more suitable，but limited．As we proceed we will learn about several language－definition mechanisms that are useful in different circumstances．Here we introduce a common and powerful one，the notion of a grammar．
A grammar for the English language tells us whether a particular sentence is well formed or not．
of a noun phrase followed by a predicate＂．More concisely we write this as

> $$
\langle\text { sentence }\rangle \rightarrow\langle\text { noun＿phrase }\rangle\langle\text { predicate〉，}
$$

with the obvious interpretation．This is，of course，not enough to deal with
actual sentences．We must now provide definitions for the newly introduced
constructs＜noun＿phrase〉 and 〈predicate〉．If we do so by
$\qquad$

## 2 THREE BASIC CONCEPTS：Grammars

To study languages mathematically，we need a mechanism to describe them． Everyday language is imprecise and ambiguous，so informal descriptions in English are often inadequate．The set notation used in Examples 1.9 and 1.10 is more suitable，but limited．As we proceed we will learn about several language－definition mechanisms that are useful in different circumstances．Here we introduce a common and powerful one，the notion of a grammar．
A grammar for the English language tells us whether a particular sentence is well formed or not．A typical rule of English grammar is＂a sentence can consist of a noun phrase followed by a predicate＂．
with the obvious interpretation．This is，of course，not enough to deal with actual sentences．We must now provide definitions for the newly introduced constructs 〈noun＿phrase〉 and 〈predicate〉．If we do so by
$\qquad$
$\square$

## 2 THREE BASIC CONCEPTS：Grammars

To study languages mathematically，we need a mechanism to describe them． Everyday language is imprecise and ambiguous，so informal descriptions in English are often inadequate．The set notation used in Examples 1.9 and 1.10 is more suitable，but limited．As we proceed we will learn about several language－definition mechanisms that are useful in different circumstances．Here we introduce a common and powerful one，the notion of a grammar．
A grammar for the English language tells us whether a particular sentence is well formed or not．A typical rule of English grammar is＂a sentence can consist of a noun phrase followed by a predicate＂．More concisely we write this as

> with the obvious interpretation. This is, of course, not enough to deal with actual sentences．We must now provide definitions for the newly introduced constructs 〈noun phrase〉 and＜predicate〉．If we do so by
$\qquad$
$\square$

## 2 THREE BASIC CONCEPTS：Grammars

To study languages mathematically，we need a mechanism to describe them． Everyday language is imprecise and ambiguous，so informal descriptions in English are often inadequate．The set notation used in Examples 1.9 and 1.10 is more suitable，but limited．As we proceed we will learn about several language－definition mechanisms that are useful in different circumstances．Here we introduce a common and powerful one，the notion of a grammar．
A grammar for the English language tells us whether a particular sentence is well formed or not．A typical rule of English grammar is＂a sentence can consist of a noun phrase followed by a predicate＂．More concisely we write this as

$$
\langle\text { sentence }\rangle \rightarrow\langle\text { noun_phrase }\rangle\langle\text { predicate }\rangle,
$$

with the obvious interpretation．This is，of course，not enough to deal with actual sentences．We must now provide definitions for the newly introduced constructs 〈noun phrase〉 and 〈predicate〉．If we do so by
$\qquad$ could be explained this way．

## 2 THREE BASIC CONCEPTS：Grammars

To study languages mathematically，we need a mechanism to describe them． Everyday language is imprecise and ambiguous，so informal descriptions in English are often inadequate．The set notation used in Examples 1.9 and 1.10 is more suitable，but limited．As we proceed we will learn about several language－definition mechanisms that are useful in different circumstances．Here we introduce a common and powerful one，the notion of a grammar．
A grammar for the English language tells us whether a particular sentence is well formed or not．A typical rule of English grammar is＂a sentence can consist of a noun phrase followed by a predicate＂．More concisely we write this as

$$
\langle\text { sentence }\rangle \rightarrow\langle\text { noun_phrase }\rangle\langle\text { predicate }\rangle,
$$

with the obvious interpretation．
actual sentences．We must now provide definitions for the newly introduced
constructs＜noun phrase〉 and 〈predicate〉．If we do so by


#### Abstract

onstructs（noun＿phrase）and（predicate）．If we do so by


$\langle$ noun phrasel $\rightarrow$／article〉／nomn


#### Abstract

and if we associate the actual words＂a＂and＂the＂with 〈article〉，＂boy＂and us that the sentences＂a boy runs＂and＂the dog walks＂are properly formed．If we were to give a complete grammar，then in theory，every proper sentence


 could be explained this way．
## 2 THREE BASIC CONCEPTS：Grammars

To study languages mathematically，we need a mechanism to describe them． Everyday language is imprecise and ambiguous，so informal descriptions in English are often inadequate．The set notation used in Examples 1.9 and 1.10 is more suitable，but limited．As we proceed we will learn about several language－definition mechanisms that are useful in different circumstances．Here we introduce a common and powerful one，the notion of a grammar．
A grammar for the English language tells us whether a particular sentence is well formed or not．A typical rule of English grammar is＂a sentence can consist of a noun phrase followed by a predicate＂．More concisely we write this as

$$
\langle\text { sentence }\rangle \rightarrow\langle\text { noun_phrase }\rangle\langle\text { predicate }\rangle,
$$

with the obvious interpretation．This is，of course，not enough to deal with actual sentences．
constructs noun
phrase〉 and（predicate）．If we do so by
> and if we associate the actual words＂a＂and＂the＂with＜article〉，＂boy＂and us that the sentences＂a boy runs＂and＂the dog walks＂are properly formed．If we were to give a complete grammar，then in theory，every proper sentence could be explained this way．

## 2 THREE BASIC CONCEPTS：Grammars

To study languages mathematically，we need a mechanism to describe them． Everyday language is imprecise and ambiguous，so informal descriptions in English are often inadequate．The set notation used in Examples 1.9 and 1.10 is more suitable，but limited．As we proceed we will learn about several language－definition mechanisms that are useful in different circumstances．Here we introduce a common and powerful one，the notion of a grammar．
A grammar for the English language tells us whether a particular sentence is well formed or not．A typical rule of English grammar is＂a sentence can consist of a noun phrase followed by a predicate＂．More concisely we write this as

$$
\langle\text { sentence }\rangle \rightarrow\langle\text { noun_phrase }\rangle\langle\text { predicate }\rangle,
$$

with the obvious interpretation．This is，of course，not enough to deal with actual sentences．We must now provide definitions for the newly introduced constructs 〈noun＿phrase〉 and 〈predicate〉．
> and if we associate the actual words＂a＂and＂the＂with 〈article〉，＂boy＂and us that the sentences＂a boy runs＂and＂the dog walks＂are properly formed．If we were to give a complete grammar，then in theory，every proper sentence could be explained this way．

## 2 THREE BASIC CONCEPTS：Grammars

To study languages mathematically，we need a mechanism to describe them． Everyday language is imprecise and ambiguous，so informal descriptions in English are often inadequate．The set notation used in Examples 1.9 and 1.10 is more suitable，but limited．As we proceed we will learn about several language－definition mechanisms that are useful in different circumstances．Here we introduce a common and powerful one，the notion of a grammar．
A grammar for the English language tells us whether a particular sentence is well formed or not．A typical rule of English grammar is＂a sentence can consist of a noun phrase followed by a predicate＂．More concisely we write this as

$$
\langle\text { sentence }\rangle \rightarrow\langle\text { noun_phrase }\rangle\langle\text { predicate }\rangle,
$$

with the obvious interpretation．This is，of course，not enough to deal with actual sentences．We must now provide definitions for the newly introduced constructs 〈noun＿phrase〉 and 〈predicate〉．If we do so by

[^1]
## 2 THREE BASIC CONCEPTS：Grammars

To study languages mathematically，we need a mechanism to describe them． Everyday language is imprecise and ambiguous，so informal descriptions in English are often inadequate．The set notation used in Examples 1.9 and 1.10 is more suitable，but limited．As we proceed we will learn about several language－definition mechanisms that are useful in different circumstances．Here we introduce a common and powerful one，the notion of a grammar．
A grammar for the English language tells us whether a particular sentence is well formed or not．A typical rule of English grammar is＂a sentence can consist of a noun phrase followed by a predicate＂．More concisely we write this as

$$
\langle\text { sentence }\rangle \rightarrow\langle\text { noun_phrase }\rangle\langle\text { predicate }\rangle,
$$

with the obvious interpretation．This is，of course，not enough to deal with actual sentences．We must now provide definitions for the newly introduced constructs $\langle$ noun＿phrase〉 and 〈predicate〉．If we do so by

$$
\begin{aligned}
\langle\text { noun_phrase }\rangle & \rightarrow\langle\text { article }\rangle\langle\text { noun }\rangle, \\
\langle\text { predicate }\rangle & \rightarrow\langle\text { verb }\rangle
\end{aligned}
$$

and if we associate the actual words＂a＂and＂the＂with 〈article〉，＂boy＂and ＂dog＂with＜noun〉，and＂runs＂and＂walks＂with＜verb〉，then the grammar tells us that the sentences＂a boy runs＂and＂the dog walks＂are properly formed．If we were to give a complete grammar，then in theory，every proper sentence could be explained this way．

## 2 THREE BASIC CONCEPTS：Grammars

To study languages mathematically，we need a mechanism to describe them． Everyday language is imprecise and ambiguous，so informal descriptions in English are often inadequate．The set notation used in Examples 1.9 and 1.10 is more suitable，but limited．As we proceed we will learn about several language－definition mechanisms that are useful in different circumstances．Here we introduce a common and powerful one，the notion of a grammar．
A grammar for the English language tells us whether a particular sentence is well formed or not．A typical rule of English grammar is＂a sentence can consist of a noun phrase followed by a predicate＂．More concisely we write this as

$$
\langle\text { sentence }\rangle \rightarrow\langle\text { noun_phrase }\rangle\langle\text { predicate }\rangle,
$$

with the obvious interpretation．This is，of course，not enough to deal with actual sentences．We must now provide definitions for the newly introduced constructs $\langle$ noun＿phrase〉 and $\langle$ predicate $\rangle$ ．If we do so by

$$
\begin{aligned}
\langle\text { noun_phrase }\rangle & \rightarrow\langle\text { article }\rangle\langle\text { noun }\rangle, \\
\langle\text { predicate }\rangle & \rightarrow\langle\text { verb }\rangle
\end{aligned}
$$

and if we associate the actual words＂a＂and＂the＂with 〈article〉， us that the sentences＂a boy runs＂and＂the dog walks＂are properly formed．If we were to give a complete grammar，then in theory，every proper sentence could be explained this way．

## 2 THREE BASIC CONCEPTS：Grammars

To study languages mathematically，we need a mechanism to describe them． Everyday language is imprecise and ambiguous，so informal descriptions in English are often inadequate．The set notation used in Examples 1.9 and 1.10 is more suitable，but limited．As we proceed we will learn about several language－definition mechanisms that are useful in different circumstances．Here we introduce a common and powerful one，the notion of a grammar．
A grammar for the English language tells us whether a particular sentence is well formed or not．A typical rule of English grammar is＂a sentence can consist of a noun phrase followed by a predicate＂．More concisely we write this as

$$
\langle\text { sentence }\rangle \rightarrow\langle\text { noun_phrase }\rangle\langle\text { predicate }\rangle,
$$

with the obvious interpretation．This is，of course，not enough to deal with actual sentences．We must now provide definitions for the newly introduced constructs $\langle$ noun＿phrase〉 and $\langle$ predicate $\rangle$ ．If we do so by

$$
\begin{aligned}
\langle\text { noun_phrase }\rangle & \rightarrow\langle\text { article }\rangle\langle\text { noun }\rangle, \\
\langle\text { predicate }\rangle & \rightarrow\langle\text { verb }\rangle
\end{aligned}
$$

and if we associate the actual words＂a＂and＂the＂with 〈article〉，＂boy＂and ＂dog＂with 〈noun〉，
us that the sentences＂a boy runs＂and＂the dog walks＂are properly formed．
we were to give a complete grammar，then in theory，every proper sentence
could be explained this wav．

## 2 THREE BASIC CONCEPTS：Grammars

To study languages mathematically，we need a mechanism to describe them． Everyday language is imprecise and ambiguous，so informal descriptions in English are often inadequate．The set notation used in Examples 1.9 and 1.10 is more suitable，but limited．As we proceed we will learn about several language－definition mechanisms that are useful in different circumstances．Here we introduce a common and powerful one，the notion of a grammar．
A grammar for the English language tells us whether a particular sentence is well formed or not．A typical rule of English grammar is＂a sentence can consist of a noun phrase followed by a predicate＂．More concisely we write this as

$$
\langle\text { sentence }\rangle \rightarrow\langle\text { noun_phrase }\rangle\langle\text { predicate }\rangle,
$$

with the obvious interpretation．This is，of course，not enough to deal with actual sentences．We must now provide definitions for the newly introduced constructs $\langle$ noun＿phrase〉 and $\langle$ predicate $\rangle$ ．If we do so by

$$
\begin{aligned}
\langle\text { noun_phrase }\rangle & \rightarrow\langle\text { article }\rangle\langle\text { noun }\rangle, \\
\langle\text { predicate }\rangle & \rightarrow\langle\text { verb }\rangle
\end{aligned}
$$

and if we associate the actual words＂a＂and＂the＂with 〈article〉，＂boy＂and ＂dog＂with 〈noun〉，and＂runs＂and＂walks＂with 〈verb〉，
we were to give a complete grammar，then in theory，every proper sentence
could be explained this may：

## 2 THREE BASIC CONCEPTS：Grammars

To study languages mathematically，we need a mechanism to describe them． Everyday language is imprecise and ambiguous，so informal descriptions in English are often inadequate．The set notation used in Examples 1.9 and 1.10 is more suitable，but limited．As we proceed we will learn about several language－definition mechanisms that are useful in different circumstances．Here we introduce a common and powerful one，the notion of a grammar．
A grammar for the English language tells us whether a particular sentence is well formed or not．A typical rule of English grammar is＂a sentence can consist of a noun phrase followed by a predicate＂．More concisely we write this as

$$
\langle\text { sentence }\rangle \rightarrow\langle\text { noun_phrase }\rangle\langle\text { predicate }\rangle,
$$

with the obvious interpretation．This is，of course，not enough to deal with actual sentences．We must now provide definitions for the newly introduced constructs $\langle$ noun＿phrase〉 and $\langle$ predicate $\rangle$ ．If we do so by

$$
\begin{aligned}
\langle\text { noun_phrase }\rangle & \rightarrow\langle\text { article }\rangle\langle\text { noun }\rangle, \\
\langle\text { predicate }\rangle & \rightarrow\langle\text { verb }\rangle
\end{aligned}
$$

and if we associate the actual words＂a＂and＂the＂with 〈article〉，＂boy＂and ＂dog＂with 〈noun〉，and＂runs＂and＂walks＂with 〈verb〉，then the grammar tells us that
we were to give a complete grammar，then in theory，every proper sentence
could be explained this way．

## 2 THREE BASIC CONCEPTS：Grammars

To study languages mathematically，we need a mechanism to describe them． Everyday language is imprecise and ambiguous，so informal descriptions in English are often inadequate．The set notation used in Examples 1.9 and 1.10 is more suitable，but limited．As we proceed we will learn about several language－definition mechanisms that are useful in different circumstances．Here we introduce a common and powerful one，the notion of a grammar．
A grammar for the English language tells us whether a particular sentence is well formed or not．A typical rule of English grammar is＂a sentence can consist of a noun phrase followed by a predicate＂．More concisely we write this as

$$
\langle\text { sentence }\rangle \rightarrow\langle\text { noun_phrase }\rangle\langle\text { predicate }\rangle,
$$

with the obvious interpretation．This is，of course，not enough to deal with actual sentences．We must now provide definitions for the newly introduced constructs 〈noun＿phrase〉 and 〈predicate〉．If we do so by

$$
\begin{aligned}
\langle\text { noun_phrase }\rangle & \rightarrow\langle\text { article }\rangle\langle\text { noun }\rangle, \\
\langle\text { predicate }\rangle & \rightarrow\langle\text { verb }\rangle
\end{aligned}
$$

and if we associate the actual words＂a＂and＂the＂with 〈article〉，＂boy＂and ＂dog＂with 〈noun〉，and＂runs＂and＂walks＂with 〈verb〉，then the grammar tells us that the sentences＂a boy runs＂and＂the dog walks＂are properly formed．

## 2 THREE BASIC CONCEPTS：Grammars

To study languages mathematically，we need a mechanism to describe them． Everyday language is imprecise and ambiguous，so informal descriptions in English are often inadequate．The set notation used in Examples 1.9 and 1.10 is more suitable，but limited．As we proceed we will learn about several language－definition mechanisms that are useful in different circumstances．Here we introduce a common and powerful one，the notion of a grammar．
A grammar for the English language tells us whether a particular sentence is well formed or not．A typical rule of English grammar is＂a sentence can consist of a noun phrase followed by a predicate＂．More concisely we write this as

$$
\langle\text { sentence }\rangle \rightarrow\langle\text { noun_phrase }\rangle\langle\text { predicate }\rangle,
$$

with the obvious interpretation．This is，of course，not enough to deal with actual sentences．We must now provide definitions for the newly introduced constructs 〈noun＿phrase〉 and 〈predicate〉．If we do so by

$$
\begin{aligned}
\langle\text { noun_phrase }\rangle & \rightarrow\langle\text { article }\rangle\langle\text { noun }\rangle, \\
\langle\text { predicate }\rangle & \rightarrow\langle\text { verb }\rangle
\end{aligned}
$$

and if we associate the actual words＂a＂and＂the＂with 〈article〉，＂boy＂and ＂dog＂with 〈noun〉，and＂runs＂and＂walks＂with 〈verb〉，then the grammar tells us that the sentences＂a boy runs＂and＂the dog walks＂are properly formed．If we were to give a complete grammar，

## 2 THREE BASIC CONCEPTS：Grammars

To study languages mathematically，we need a mechanism to describe them． Everyday language is imprecise and ambiguous，so informal descriptions in English are often inadequate．The set notation used in Examples 1.9 and 1.10 is more suitable，but limited．As we proceed we will learn about several language－definition mechanisms that are useful in different circumstances．Here we introduce a common and powerful one，the notion of a grammar．
A grammar for the English language tells us whether a particular sentence is well formed or not．A typical rule of English grammar is＂a sentence can consist of a noun phrase followed by a predicate＂．More concisely we write this as

$$
\langle\text { sentence }\rangle \rightarrow\langle\text { noun_phrase }\rangle\langle\text { predicate }\rangle,
$$

with the obvious interpretation．This is，of course，not enough to deal with actual sentences．We must now provide definitions for the newly introduced constructs 〈noun＿phrase〉 and 〈predicate〉．If we do so by

$$
\begin{aligned}
\langle\text { noun_phrase }\rangle & \rightarrow\langle\text { article }\rangle\langle\text { noun }\rangle, \\
\langle\text { predicate }\rangle & \rightarrow\langle\text { verb }\rangle
\end{aligned}
$$

and if we associate the actual words＂a＂and＂the＂with 〈article〉，＂boy＂and ＂dog＂with 〈noun〉，and＂runs＂and＂walks＂with 〈verb〉，then the grammar tells us that the sentences＂a boy runs＂and＂the dog walks＂are properly formed．If we were to give a complete grammar，then in theory，every proper sentence could be explained this way．

```
This example illustrates the definition of a general concept in terms of simple
ones. We start with the top-level concept, here <sentence\rangle, and successively
reduce it to the irreducible building blocks of the language. The generalization
of these ideas leads us to formal grammars.
```


## Definition 1.1

This example illustrates the definition of a general concept in terms of simple ones. We start with the top-level concept, here (sentence), and successively reduce it to the irreducible building blocks of the language. The generalization of these ideas leads us to formal grammars.

## Definition 1.1

This example illustrates the definition of a general concept in terms of simple ones. We start with the top-level concept, here (sentence), and successively reduce it to the irreducible building blocks of the language. The generalization of these ideas leads us to formal grammars.

## Definition 1.1

This example illustrates the definition of a general concept in terms of simple ones. We start with the top-level concept, here 〈sentence〉,
and successively reduce it to the irreducible building blocks of the language. The generalization of these ideas leads us to formal grammars.

## Definition 1.1

This example illustrates the definition of a general concept in terms of simple ones. We start with the top-level concept, here 〈sentence〉, and successively reduce it to the irreducible building blocks of the language.

## Definition 1.1

This example illustrates the definition of a general concept in terms of simple ones. We start with the top-level concept, here 〈sentence〉, and successively reduce it to the irreducible building blocks of the language. The generalization of these ideas leads us to formal grammars.

This example illustrates the definition of a general concept in terms of simple ones. We start with the top-level concept, here 〈sentence〉, and successively reduce it to the irreducible building blocks of the language. The generalization of these ideas leads us to formal grammars.

## Definition 1.1

A grammar $G$ is defined as a quadruple
where

It will be assumed without further mention that the sets $V$ and $T$ are
nonempty and disjoint.

This example illustrates the definition of a general concept in terms of simple ones. We start with the top-level concept, here 〈sentence〉, and successively reduce it to the irreducible building blocks of the language. The generalization of these ideas leads us to formal grammars.

## Definition 1.1

A grammar $G$ is defined as a quadruple

[^2]This example illustrates the definition of a general concept in terms of simple ones. We start with the top-level concept, here 〈sentence〉, and successively reduce it to the irreducible building blocks of the language. The generalization of these ideas leads us to formal grammars.

## Definition 1.1

A grammar $G$ is defined as a quadruple

$$
G=(V, T, S, P)
$$

It will be assumed without further mention that the sets $V$ and $T$ are
nonempty and disjoint.

This example illustrates the definition of a general concept in terms of simple ones. We start with the top-level concept, here 〈sentence〉, and successively reduce it to the irreducible building blocks of the language. The generalization of these ideas leads us to formal grammars.

## Definition 1.1

A grammar $G$ is defined as a quadruple

$$
G=(V, T, S, P)
$$

where

It will be assumed without further mention that the sets $V$ and $T$ are
nonempty and disjoint.

This example illustrates the definition of a general concept in terms of simple ones. We start with the top-level concept, here 〈sentence〉, and successively reduce it to the irreducible building blocks of the language. The generalization of these ideas leads us to formal grammars.

## Definition 1.1

A grammar $G$ is defined as a quadruple

$$
G=(V, T, S, P)
$$

where

- $V$ is a finite set of objects called variables;
- $T$ is a finite set of objects called terminal symbols;
- $S \in V$ is a snecial symbol called the start variahle.
- $P$ is a finite set of productions.

It will be assumed without further mention that the sets $V$ and $T$ are
nonempty and disjoint.

This example illustrates the definition of a general concept in terms of simple ones. We start with the top-level concept, here 〈sentence〉, and successively reduce it to the irreducible building blocks of the language. The generalization of these ideas leads us to formal grammars.

## Definition 1.1

A grammar $G$ is defined as a quadruple

$$
G=(V, T, S, P)
$$

where

- $V$ is a finite set of objects called variables;
- $T$ is a finite set of objects called terminal symbols;
- $S \in V$ is a special symbol called the start variable;
- $P$ is a finite set of productions.

It will be assumed without further mention that the sets $V$ and $T$ are
nonempty and disjoint.

This example illustrates the definition of a general concept in terms of simple ones. We start with the top-level concept, here 〈sentence〉, and successively reduce it to the irreducible building blocks of the language. The generalization of these ideas leads us to formal grammars.

## Definition 1.1

A grammar $G$ is defined as a quadruple

$$
G=(V, T, S, P)
$$

where

- $V$ is a finite set of objects called variables;
- $T$ is a finite set of objects called terminal symbols;
- $S \in V$ is a special symbol called the start variable;
- $P$ is a finite set of productions.

It will be assumed without further mention that the sets $V$ and $I$ are
nonempty and disjoint.

This example illustrates the definition of a general concept in terms of simple ones. We start with the top-level concept, here 〈sentence〉, and successively reduce it to the irreducible building blocks of the language. The generalization of these ideas leads us to formal grammars.

## Definition 1.1

A grammar $G$ is defined as a quadruple

$$
G=(V, T, S, P)
$$

where

- $V$ is a finite set of objects called variables;
- $T$ is a finite set of objects called terminal symbols;
- $S \in V$ is a special symbol called the start variable;

It will be assumed without further mention that the sets $V$ and $T$ are
nonempty and disjoint.

This example illustrates the definition of a general concept in terms of simple ones. We start with the top-level concept, here 〈sentence〉, and successively reduce it to the irreducible building blocks of the language. The generalization of these ideas leads us to formal grammars.

## Definition 1.1

A grammar $G$ is defined as a quadruple

$$
G=(V, T, S, P)
$$

where

- $V$ is a finite set of objects called variables;
- $T$ is a finite set of objects called terminal symbols;
- $S \in V$ is a special symbol called the start variable;
- $P$ is a finite set of productions.

It will be assumed without further mention that the sets $V$ and $T$ are
nonempty and disjoint.

This example illustrates the definition of a general concept in terms of simple ones. We start with the top-level concept, here 〈sentence〉, and successively reduce it to the irreducible building blocks of the language. The generalization of these ideas leads us to formal grammars.

## Definition 1.1

A grammar $G$ is defined as a quadruple

$$
G=(V, T, S, P)
$$

where

- $V$ is a finite set of objects called variables;
- $T$ is a finite set of objects called terminal symbols;
- $S \in V$ is a special symbol called the start variable;
- $P$ is a finite set of productions.

It will be assumed without further mention that the sets $V$ and $T$ are nonempty and disjoint.

## 2 THREE BASIC CONCEPTS: Grammars

The production rules are the heart of a grammar; they specity how the grammar transforms one string into another, and through this they define a language associated with the grammar. In our discussion we will assume that all production rules are of the form

$$
x \rightarrow y
$$

where $x$ is an element of $(V \cup T)^{+}$and $y$ is in $(V \cup T)^{*}$. The productions are applied in the following manner: Given a string $w$ of the form

$$
w=u x v
$$

we say the production $x \rightarrow y$ is applicable to this string, and we may use it to replace $x$ with $y$, thereby obtaining a new string

$$
z=u y v
$$

This is written as

$$
w \Rightarrow z
$$

We say that $w$ derives $z$ or that $z$ is derived from $w$. Successive strings are derived by applying the productions of the grammar in arbitrary order. A production can be used whenever it is applicable, and it can be applied as often as desired. If

$$
w_{1} \Rightarrow w_{2} \Rightarrow \cdots \Rightarrow w_{n}
$$

we say that $w_{1}$ derives $w_{n}$ and write

$$
w_{1} \stackrel{*}{\Rightarrow} w_{n}
$$

The star $*$ indicates that an unspecified number of steps (including zero) can be taken to derive $w_{n}$ from $w_{1}$.

## 2 THREE BASIC CONCEPTS: Grammars

The production rules are the heart of a grammar; they specify how the
grammar transforms one string into another, and through this they define a language associated with the grammar. In our discussion we will assume that all production rules are of the form

```
where }x\mathrm{ is an element of }(V\cupT\mp@subsup{)}{}{+}\mathrm{ and }y\mathrm{ is in ( }V\cupT\mp@subsup{)}{}{*}\mathrm{ . The productions are
applied in the following manner: Given a string w of the form
```

```
w = uxv,
```

we say the production $x \rightarrow y$ is applicable to this string, and we may use it to
replace $x$ with $y$, thereby obtaining a new string
$z=u y v$.
This is written as
We say that $w$ derives $z$ or that $z$ is derived from $w$. Successive strings are
derived by applying the productions of the grammar in arbitrary order. A
production can be used whenever it is applicable, and it can be applied as often
as desired. If
$H_{1} \Rightarrow H_{2} \Rightarrow \cdots \Rightarrow H_{n}$
we say that $w_{1}$ derives $w_{n}$ and write
$w_{1} \stackrel{*}{\Rightarrow} w_{n}$.
The star * indicates that an unspecified number of steps (including zero) can
be taken to derive $w_{n}$ from $w_{1}$

## 2 THREE BASIC CONCEPTS: Grammars

The production rules are the heart of a grammar; they specify how the grammar transforms one string into another, and through this they define a language associated with the grammar. In our discussion we will assume that all production rules are of the form

```
where }x\mathrm{ is an element of }(V\cupT\mp@subsup{)}{}{+}\mathrm{ and }y\mathrm{ is in (VUT)*. The productions are
applied in the following manner: Given a string w of the form
```

```
w = uxv
```

we say the production $x \rightarrow y$ is applicable to this string, and we may use it to
replace $x$ with $y$, thereby obtaining a new string
This is written as
We say that $w$ derives $z$ or that $z$ is derived from $w$. Successive strings are
derived by applying the productions of the grammar in arbitrary order. A
production can be used whenever it is applicable, and it can be applied as often
as desired. If
$w_{1} \Rightarrow w_{2} \Rightarrow \cdots \Rightarrow w_{n}$,
we say that $w_{1}$ derives $w_{n}$ and mrite

The star * indicates that an unspecified number of steps (including zero) can be taken to derive $w_{n}$ from $w_{1}$

## 2 THREE BASIC CONCEPTS: Grammars

The production rules are the heart of a grammar; they specify how the grammar transforms one string into another, and through this they define a language associated with the grammar.

```
all production rules are of the form
applied in the following manner: Given a string w of the form
```

we say the production $x \rightarrow y$ is applicable to this string, and we may use it to
replace $x$ with $y$, thereby obtaining a new string
This is written as
We say that $w$ derives $z$ or that $z$ is derived from $w$. Successive strings are
derived by applying the productions of the grammar in arbitrary order. A
production can be used whenever it is applicable, and it can be applied as often
as desired. If
we say that $w_{1}$ derives $w_{n}$ and write
The star * indicates that an unspecified number of steps (including zero) can
be taken to derive $w_{n}$ from $w_{1}$

## 2 THREE BASIC CONCEPTS: Grammars

The production rules are the heart of a grammar; they specify how the grammar transforms one string into another, and through this they define a language associated with the grammar. In our discussion we will assume that all production rules are of the form
 we say the production $x \rightarrow y$ is applicable to this string, and we may use it to replace $x$ with $y$, thereby obtaining a new string

This is written as

We say that $w$ derives $z$ or that $z$ is derived from $w$. Successive strings are derived by applying the productions of the grammar in arbitrary order. A production can be used whenever it is applicable, and it can be applied as often as desired. If
we say that $w_{1}$ derives $w_{n}$ and write

The star * indicates that an unspecified number of steps (including zero) can
be taken to derive $w_{n}$ from $w_{1}$

## 2 THREE BASIC CONCEPTS: Grammars

The production rules are the heart of a grammar; they specify how the grammar transforms one string into another, and through this they define a language associated with the grammar. In our discussion we will assume that all production rules are of the form

$$
x \rightarrow y
$$

 replace $x$ with $y$, thereby obtaining a new string

This is written as

We say that $w$ derives $z$ or that $z$ is derived from $w$. Successive strings are derived by applying the productions of the grammar in arbitrary order. A production can be used whenever it is applicable, and it can be applied as often as desired. If
we say that $w_{1}$ derives $w_{n}$ and write

The star * indicates that an unspecified number of steps (including zero) can be taken to derive $w_{n}$ from $w_{1}$

## 2 THREE BASIC CONCEPTS: Grammars

The production rules are the heart of a grammar; they specify how the grammar transforms one string into another, and through this they define a language associated with the grammar. In our discussion we will assume that all production rules are of the form

$$
x \rightarrow y
$$

where $x$ is an element of $(V \cup T)^{+}$and $y$ is in $(V \cup T)^{*}$. The productions are applied in the following manner: Given a string $w$ of the form

$$
w=u x v,
$$

we say the production $x \rightarrow y$ is applicable to this string, and we may use it to replace $x$ with $y$, thereby obtaining a new string

This is written as

We say that $w$ derives $z$ or that $z$ is derived from $w$. Successive strings are derived by applying the productions of the grammar in arbitrary order. A production can be used whenever it is applicable, and it can be applied as often as desired. If
we say that $w_{1}$ derives $w_{n}$ and write

The star * indicates that an unspecified number of steps (including zero) can
be taken to derive $w_{n}$ from $w_{1}$

## 2 THREE BASIC CONCEPTS: Grammars

The production rules are the heart of a grammar; they specify how the grammar transforms one string into another, and through this they define a language associated with the grammar. In our discussion we will assume that all production rules are of the form

$$
x \rightarrow y
$$

where $x$ is an element of $(V \cup T)^{+}$and $y$ is in $(V \cup T)^{*}$. The productions are applied in the following manner:

$$
\text { Given a string } w \text { of the form }
$$

$\square$ replace $x$ with $y$, thereby obtaining a new string

This is written as

We say that $w$ derives $z$ or that $z$ is derived from $w$. Successive strings are derived by applying the productions of the grammar in arbitrary order. A production can be used whenever it is applicable, and it can be applied as often as desired. If
we say that $w_{1}$ derives $w_{n}$ and write

The star * indicates that an unspecified number of steps (including zero) can be taken to derive $w_{n}$ from $w_{1}$

## 2 THREE BASIC CONCEPTS: Grammars

The production rules are the heart of a grammar; they specify how the grammar transforms one string into another, and through this they define a language associated with the grammar. In our discussion we will assume that all production rules are of the form

$$
x \rightarrow y
$$

where $x$ is an element of $(V \cup T)^{+}$and $y$ is in $(V \cup T)^{*}$. The productions are applied in the following manner: Given a string $w$ of the form
we say the production $x \rightarrow y$ is applicable to this string, and we may use it to replace $x$ with $y$, thereby obtaining a new string

This is written as

We say that $w$ derives $z$ or that $z$ is derived from $w$. Successive strings are derived by applying the productions of the grammar in arbitrary order. A production can be used whenever it is applicable, and it can be applied as often as desired. If
we say that $w_{1}$ derives $w_{n}$ and write

The star * indicates that an unspecified number of steps (including zero) can be taken to derive $w_{n}$ from $w_{1}$

## 2 THREE BASIC CONCEPTS: Grammars

The production rules are the heart of a grammar; they specify how the grammar transforms one string into another, and through this they define a language associated with the grammar. In our discussion we will assume that all production rules are of the form

$$
x \rightarrow y
$$

where $x$ is an element of $(V \cup T)^{+}$and $y$ is in $(V \cup T)^{*}$. The productions are applied in the following manner: Given a string $w$ of the form

$$
w=u x v
$$

> replace $x$ with $y$, thereby obtaining a new string

This is written as

We say that $w$ derives $z$ or that $z$ is derived from $w$. Successive strings are derived by applying the productions of the grammar in arbitrary order. A production can be used whenever it is applicable, and it can be applied as often as desired. If
we say that $w_{1}$ derives $w_{n}$ and write

The star * indicates that an unspecified number of steps (including zero) can be taken to derive $w_{n}$ from $w_{1}$

## 2 THREE BASIC CONCEPTS: Grammars

The production rules are the heart of a grammar; they specify how the grammar transforms one string into another, and through this they define a language associated with the grammar. In our discussion we will assume that all production rules are of the form

$$
x \rightarrow y
$$

where $x$ is an element of $(V \cup T)^{+}$and $y$ is in $(V \cup T)^{*}$. The productions are applied in the following manner: Given a string $w$ of the form

$$
w=u x v
$$

we say the production $x \rightarrow y$ is applicable to this string, and we may use it to replace $x$ with $y$, thereby obtaining a new string

This is written as

We say that $w$ derives $z$ or that $z$ is derived from $w$. Successive strings are derived by applying the productions of the grammar in arbitrary order. A production can be used whenever it is applicable, and it can be applied as often as desired. If
we say that $w_{1}$ derives $w_{n}$ and write

The star * indicates that an unspecified number of steps (including zero) can be taken to derive $w_{n}$ from $w_{1}$

## 2 THREE BASIC CONCEPTS: Grammars

The production rules are the heart of a grammar; they specify how the grammar transforms one string into another, and through this they define a language associated with the grammar. In our discussion we will assume that all production rules are of the form

$$
x \rightarrow y
$$

where $x$ is an element of $(V \cup T)^{+}$and $y$ is in $(V \cup T)^{*}$. The productions are applied in the following manner: Given a string $w$ of the form

$$
w=u x v
$$

we say the production $x \rightarrow y$ is applicable to this string, and we may use it to replace $x$ with $y$,

This is written as

We say that $w$ derives $z$ or that $z$ is derived from $w$. Successive strings are derived by applying the productions of the grammar in arbitrary order. A production can be used whenever it is applicable, and it can be applied as often as desired. If
we say that $w_{1}$ derives $w_{n}$ and write

The star * indicates that an unspecified number of steps (including zero) can be taken to derive $w_{n}$ from $w_{1}$

## 2 THREE BASIC CONCEPTS: Grammars

The production rules are the heart of a grammar; they specify how the grammar transforms one string into another, and through this they define a language associated with the grammar. In our discussion we will assume that all production rules are of the form

$$
x \rightarrow y
$$

where $x$ is an element of $(V \cup T)^{+}$and $y$ is in $(V \cup T)^{*}$. The productions are applied in the following manner: Given a string $w$ of the form

$$
w=u x v
$$

we say the production $x \rightarrow y$ is applicable to this string, and we may use it to replace $x$ with $y$, thereby obtaining a new string

This is written as

We say that $w$ derives $z$ or that $z$ is derived from $w$. Successive strings are derived by applying the productions of the grammar in arbitrary order. A production can be used whenever it is applicable, and it can be applied as often as desired. If
we say that $w_{1}$ derives $w_{n}$ and write

The star * indicates that an unspecified number of steps (including zero) can be taken to derive $w_{n}$ from $w_{1}$

## 2 THREE BASIC CONCEPTS: Grammars

The production rules are the heart of a grammar; they specify how the grammar transforms one string into another, and through this they define a language associated with the grammar. In our discussion we will assume that all production rules are of the form

$$
x \rightarrow y
$$

where $x$ is an element of $(V \cup T)^{+}$and $y$ is in $(V \cup T)^{*}$. The productions are applied in the following manner: Given a string $w$ of the form

$$
w=u x v
$$

we say the production $x \rightarrow y$ is applicable to this string, and we may use it to replace $x$ with $y$, thereby obtaining a new string

$$
z=u y v
$$

This is written as
We say that $w$ derives $z$ or that $z$ is derived from $w$. Successive strings are derived by applying the productions of the grammar in arbitrary order. A production can be used whenever it is applicable, and it can be applied as often as desired. If
we say that $w_{1}$ derives $w_{n}$ and write

The star * indicates that an unspecified number of steps (including zero) can be taken to derive $w_{n}$ from $w_{1}$

## 2 THREE BASIC CONCEPTS: Grammars

The production rules are the heart of a grammar; they specify how the grammar transforms one string into another, and through this they define a language associated with the grammar. In our discussion we will assume that all production rules are of the form

$$
x \rightarrow y
$$

where $x$ is an element of $(V \cup T)^{+}$and $y$ is in $(V \cup T)^{*}$. The productions are applied in the following manner: Given a string $w$ of the form

$$
w=u x v
$$

we say the production $x \rightarrow y$ is applicable to this string, and we may use it to replace $x$ with $y$, thereby obtaining a new string

$$
z=u y v
$$

This is written as

> We say that $w$ derives $z$ or that $z$ is derived from $w$. Successive strings are derived by applying the productions of the grammar in arbitrary order. A production can be used whenever it is applicable, and it can be applied as often as desired. If

we say that $w_{1}$ derives $w_{n}$ and write

The star * indicates that an unspecified number of steps (including zero) can be taken to derive $w_{n}$ from $w_{1}$

## 2 THREE BASIC CONCEPTS: Grammars

The production rules are the heart of a grammar; they specify how the grammar transforms one string into another, and through this they define a language associated with the grammar. In our discussion we will assume that all production rules are of the form

$$
x \rightarrow y
$$

where $x$ is an element of $(V \cup T)^{+}$and $y$ is in $(V \cup T)^{*}$. The productions are applied in the following manner: Given a string $w$ of the form

$$
w=u x v
$$

we say the production $x \rightarrow y$ is applicable to this string, and we may use it to replace $x$ with $y$, thereby obtaining a new string

$$
z=u y v
$$

This is written as

$$
w \Rightarrow z .
$$

> We say that $w$ derives $z$ or that $z$ is derived from $w$. Successive strings are derived by applying the productions of the grammar in arbitrary order. A production can be used whenever it is applicable, and it can be applied as often as desired. If

we say that $w_{1}$ derives $w_{n}$ and write

The star * indicates that an unspecified number of steps (including zero) can be taken to derive $w_{n}$ from $w_{1}$

## 2 THREE BASIC CONCEPTS: Grammars

The production rules are the heart of a grammar; they specify how the grammar transforms one string into another, and through this they define a language associated with the grammar. In our discussion we will assume that all production rules are of the form

$$
x \rightarrow y
$$

where $x$ is an element of $(V \cup T)^{+}$and $y$ is in $(V \cup T)^{*}$. The productions are applied in the following manner: Given a string $w$ of the form

$$
w=u x v
$$

we say the production $x \rightarrow y$ is applicable to this string, and we may use it to replace $x$ with $y$, thereby obtaining a new string

$$
z=u y v
$$

This is written as

$$
w \Rightarrow z .
$$

We say that $w$ derives $z$ or that $z$ is derived from $w$. Successive strings are production can be used whenever it is applicable, and it can be applied as often as desired. If
we say that $w_{1}$ derives $w_{n}$ and write

The star * indicates that an unspecified number of steps (including zero) can be taken to derive $w_{n}$ from $w_{1}$

## 2 THREE BASIC CONCEPTS: Grammars

The production rules are the heart of a grammar; they specify how the grammar transforms one string into another, and through this they define a language associated with the grammar. In our discussion we will assume that all production rules are of the form

$$
x \rightarrow y
$$

where $x$ is an element of $(V \cup T)^{+}$and $y$ is in $(V \cup T)^{*}$. The productions are applied in the following manner: Given a string $w$ of the form

$$
w=u x v
$$

we say the production $x \rightarrow y$ is applicable to this string, and we may use it to replace $x$ with $y$, thereby obtaining a new string

$$
z=u y v
$$

This is written as

$$
w \Rightarrow z .
$$

We say that $w$ derives $z$ or that $z$ is derived from $w$. Successive strings are derived by applying the productions of the grammar in arbitrary order.

production c<br>we say that $w_{1}$ derives $w_{n}$ and write

The star * indicates that an unspecified number of steps (including zero) can be taken to derive $w_{n}$ from $w_{1}$

## 2 THREE BASIC CONCEPTS: Grammars

The production rules are the heart of a grammar; they specify how the grammar transforms one string into another, and through this they define a language associated with the grammar. In our discussion we will assume that all production rules are of the form

$$
x \rightarrow y
$$

where $x$ is an element of $(V \cup T)^{+}$and $y$ is in $(V \cup T)^{*}$. The productions are applied in the following manner: Given a string $w$ of the form

$$
w=u x v
$$

we say the production $x \rightarrow y$ is applicable to this string, and we may use it to replace $x$ with $y$, thereby obtaining a new string

$$
z=u y v
$$

This is written as

$$
w \Rightarrow z .
$$

We say that $w$ derives $z$ or that $z$ is derived from $w$. Successive strings are derived by applying the productions of the grammar in arbitrary order. A production can be used whenever it is applicable, as desired. If
we say that $w_{1}$ derives $w_{n}$ and write

The star * indicates that an unspecified number of steps (including zero) can be taken to derive $w_{n}$ from $w_{1}$

## 2 THREE BASIC CONCEPTS: Grammars

The production rules are the heart of a grammar; they specify how the grammar transforms one string into another, and through this they define a language associated with the grammar. In our discussion we will assume that all production rules are of the form

$$
x \rightarrow y
$$

where $x$ is an element of $(V \cup T)^{+}$and $y$ is in $(V \cup T)^{*}$. The productions are applied in the following manner: Given a string $w$ of the form

$$
w=u x v
$$

we say the production $x \rightarrow y$ is applicable to this string, and we may use it to replace $x$ with $y$, thereby obtaining a new string

$$
z=u y v
$$

This is written as

$$
w \Rightarrow z .
$$

We say that $w$ derives $z$ or that $z$ is derived from $w$. Successive strings are derived by applying the productions of the grammar in arbitrary order. A production can be used whenever it is applicable, and it can be applied as often as desired.
we say that $w_{1}$ derives $w_{n}$ and write

The star * indicates that an unspecified number of steps (including zero) can be taken to derive $w_{n}$ from $w_{1}$

## 2 THREE BASIC CONCEPTS: Grammars

The production rules are the heart of a grammar; they specify how the grammar transforms one string into another, and through this they define a language associated with the grammar. In our discussion we will assume that all production rules are of the form

$$
x \rightarrow y
$$

where $x$ is an element of $(V \cup T)^{+}$and $y$ is in $(V \cup T)^{*}$. The productions are applied in the following manner: Given a string $w$ of the form

$$
w=u x v
$$

we say the production $x \rightarrow y$ is applicable to this string, and we may use it to replace $x$ with $y$, thereby obtaining a new string

$$
z=u y v
$$

This is written as

$$
w \Rightarrow z .
$$

We say that $w$ derives $z$ or that $z$ is derived from $w$. Successive strings are derived by applying the productions of the grammar in arbitrary order. A production can be used whenever it is applicable, and it can be applied as often as desired. If
we say that $w_{1}$ derives $w_{n}$ and write

The star * indicates that an unspecified number of steps (including zero) can be taken to derive $w_{n}$ from $w_{1}$

## 2 THREE BASIC CONCEPTS: Grammars

The production rules are the heart of a grammar; they specify how the grammar transforms one string into another, and through this they define a language associated with the grammar. In our discussion we will assume that all production rules are of the form

$$
x \rightarrow y
$$

where $x$ is an element of $(V \cup T)^{+}$and $y$ is in $(V \cup T)^{*}$. The productions are applied in the following manner: Given a string $w$ of the form

$$
w=u x v
$$

we say the production $x \rightarrow y$ is applicable to this string, and we may use it to replace $x$ with $y$, thereby obtaining a new string

$$
z=u y v
$$

This is written as

$$
w \Rightarrow z .
$$

We say that $w$ derives $z$ or that $z$ is derived from $w$. Successive strings are derived by applying the productions of the grammar in arbitrary order. A production can be used whenever it is applicable, and it can be applied as often as desired. If

$$
w_{1} \Rightarrow w_{2} \Rightarrow \cdots \Rightarrow w_{n}
$$

we say that $w_{1}$ derives $w_{n}$ and write

The star * indicates that an unspecified number of steps (including zero) can be taken to derive $w_{n}$ from $w_{1}$

## 2 THREE BASIC CONCEPTS: Grammars

The production rules are the heart of a grammar; they specify how the grammar transforms one string into another, and through this they define a language associated with the grammar. In our discussion we will assume that all production rules are of the form

$$
x \rightarrow y
$$

where $x$ is an element of $(V \cup T)^{+}$and $y$ is in $(V \cup T)^{*}$. The productions are applied in the following manner: Given a string $w$ of the form

$$
w=u x v
$$

we say the production $x \rightarrow y$ is applicable to this string, and we may use it to replace $x$ with $y$, thereby obtaining a new string

$$
z=u y v
$$

This is written as

$$
w \Rightarrow z .
$$

We say that $w$ derives $z$ or that $z$ is derived from $w$. Successive strings are derived by applying the productions of the grammar in arbitrary order. A production can be used whenever it is applicable, and it can be applied as often as desired. If

$$
w_{1} \Rightarrow w_{2} \Rightarrow \cdots \Rightarrow w_{n}
$$

we say that $w_{1}$ derives $w_{n}$ and write

The star * indicates that an unspecified number of steps (including zero) can be taken to derive $w_{n}$ from $w_{1}$

## 2 THREE BASIC CONCEPTS: Grammars

The production rules are the heart of a grammar; they specify how the grammar transforms one string into another, and through this they define a language associated with the grammar. In our discussion we will assume that all production rules are of the form

$$
x \rightarrow y
$$

where $x$ is an element of $(V \cup T)^{+}$and $y$ is in $(V \cup T)^{*}$. The productions are applied in the following manner: Given a string $w$ of the form

$$
w=u x v
$$

we say the production $x \rightarrow y$ is applicable to this string, and we may use it to replace $x$ with $y$, thereby obtaining a new string

$$
z=u y v
$$

This is written as

$$
w \Rightarrow z .
$$

We say that $w$ derives $z$ or that $z$ is derived from $w$. Successive strings are derived by applying the productions of the grammar in arbitrary order. A production can be used whenever it is applicable, and it can be applied as often as desired. If

$$
w_{1} \Rightarrow w_{2} \Rightarrow \cdots \Rightarrow w_{n}
$$

we say that $w_{1}$ derives $w_{n}$ and write

$$
w_{1} \stackrel{*}{\Rightarrow} w_{n}
$$

The star * indicates that an unspecified number of steps (including zero) can be taken to derive $w_{n}$ from $w_{1}$

## 2 THREE BASIC CONCEPTS: Grammars

The production rules are the heart of a grammar; they specify how the grammar transforms one string into another, and through this they define a language associated with the grammar. In our discussion we will assume that all production rules are of the form

$$
x \rightarrow y
$$

where $x$ is an element of $(V \cup T)^{+}$and $y$ is in $(V \cup T)^{*}$. The productions are applied in the following manner: Given a string $w$ of the form

$$
w=u x v
$$

we say the production $x \rightarrow y$ is applicable to this string, and we may use it to replace $x$ with $y$, thereby obtaining a new string

$$
z=u y v
$$

This is written as

$$
w \Rightarrow z .
$$

We say that $w$ derives $z$ or that $z$ is derived from $w$. Successive strings are derived by applying the productions of the grammar in arbitrary order. A production can be used whenever it is applicable, and it can be applied as often as desired. If

$$
w_{1} \Rightarrow w_{2} \Rightarrow \cdots \Rightarrow w_{n}
$$

we say that $w_{1}$ derives $w_{n}$ and write

$$
w_{1} \stackrel{*}{\Rightarrow} w_{n}
$$

The star $*$ indicates that an unspecified number of steps (including zero) can be taken to derive $w_{n}$ from $w_{1}$.

By applying the production rules in a different order, a given grammar can normally generate many strings. The set of all such terminal strings is the language defined or generated by the grammar.

## Definition 1.2

If $w \in L(G)$ then the sequence

$$
S \Rightarrow w_{1} \Rightarrow w_{2} \Rightarrow \cdots \Rightarrow w_{n} \Rightarrow w
$$

is a derivation of the sentence $w$. The strings $S, w_{1}, w_{2}, \ldots, w_{n}$, which contain variables as well as terminals, are called sentential forms of the derivation.

By applying the production rules in a different order, a given grammar can normally generate many strings. The set of all such terminal strings is the language defined or generated by the grammar.

Definition 1?

If $w \in L(G)$ then the sequence

$$
\begin{aligned}
& \qquad S \Rightarrow w_{1} \Rightarrow w_{2} \Rightarrow \cdots \Rightarrow w_{n} \Rightarrow w \\
& \text { is a derivation of the sentence } w \text {. The strings } S, w_{1}, w_{2}, \ldots, w_{n} \text {, which contain } \\
& \text { variables as well as terminals, are called sentential forms of the derivation. }
\end{aligned}
$$

By applying the production rules in a different order, a given grammar can normally generate many strings. The set of all such terminal strings is the language defined or generated by the grammar.

Definition 1.2

If $w \in L(G)$ then the sequence
is a derivation of the sentence $w$. The strings $S, w_{1}, w_{2}, \ldots, w_{n}$, which contain variables as well as terminals, are called sentential forms of the derivation.

By applying the production rules in a different order, a given grammar can normally generate many strings. The set of all such terminal strings is the language defined or generated by the grammar.

## Definition 1.2

If $w \in L(G)$ then the sequence
is a derivation of the sentence $w$. The strings $S, w_{1}, w_{2}, \ldots, w_{n}$, which contain variables as well as terminals, are called sentential forms of the derivation.

By applying the production rules in a different order, a given grammar can normally generate many strings. The set of all such terminal strings is the language defined or generated by the grammar.

## Definition 1.2



If $w \in L(G)$ then the sequence
is a derivation of the sentence $w$. The strings $S, w_{1}, w_{2}, \ldots, w_{n}$, which contain variables as well as terminals, are called sentential forms of the derivation.

By applying the production rules in a different order, a given grammar can normally generate many strings. The set of all such terminal strings is the language defined or generated by the grammar.

## Definition 1.2

Let $G=(V, T, S, P)$ be a grammar. Then the set


If $w \in L(G)$ then the sequence
is a derivation of the sentence $w$. The strings $S, w_{1}, w_{2}, \ldots, w_{n}$, which contain variables as well as terminals, are called sentential forms of the derivation.

By applying the production rules in a different order, a given grammar can normally generate many strings. The set of all such terminal strings is the language defined or generated by the grammar.

## Definition 1.2

Let $G=(V, T, S, P)$ be a grammar. Then the set


```
is the language generated by G.
```

If $w \in L(G)$ then the sequence
is a derivation of the sentence $w$. The strings $S, w_{1}, w_{2}, \ldots, w_{n}$, which contain variables as well as terminals, are called sentential forms of the derivation.

By applying the production rules in a different order, a given grammar can normally generate many strings. The set of all such terminal strings is the language defined or generated by the grammar.

## Definition 1.2

Let $G=(V, T, S, P)$ be a grammar. Then the set

$$
L(G)=\left\{w \in T^{*}: S \stackrel{*}{\Rightarrow} w\right\}
$$

If $w \in L(G)$ then the sequence
is a derivation of the sentence $w$. The strings $S, w_{1}, w_{2}, \ldots, w_{n}$, which contain variables as well as terminals, are called sentential forms of the derivation.

By applying the production rules in a different order, a given grammar can normally generate many strings. The set of all such terminal strings is the language defined or generated by the grammar.

## Definition 1.2

Let $G=(V, T, S, P)$ be a grammar. Then the set

$$
L(G)=\left\{w \in T^{*}: S \stackrel{*}{\Rightarrow} w\right\}
$$

is the language generated by $G$.
If $w \in L(G)$ then the sequence
is a derivation of the sentence $w$. The strings $S, w_{1}, w_{2}, \ldots, w_{n}$, which contain variables as well as terminals, are called sentential forms of the derivation.

By applying the production rules in a different order, a given grammar can normally generate many strings. The set of all such terminal strings is the language defined or generated by the grammar.

## Definition 1.2

Let $G=(V, T, S, P)$ be a grammar. Then the set

$$
L(G)=\left\{w \in T^{*}: S \stackrel{*}{\Rightarrow} w\right\}
$$

is the language generated by $G$.
If $w \in L(G)$ then the sequence
is a derivation of the sentence $w$. The strings $S, w_{1}, w_{2}, \ldots, w_{n}$, which contain variables as well as terminals, are called sentential forms of the derivation.

By applying the production rules in a different order, a given grammar can normally generate many strings. The set of all such terminal strings is the language defined or generated by the grammar.

## Definition 1.2

Let $G=(V, T, S, P)$ be a grammar. Then the set

$$
L(G)=\left\{w \in T^{*}: S \stackrel{*}{\Rightarrow} w\right\}
$$

is the language generated by $G$.
If $w \in L(G)$ then the sequence

$$
S \Rightarrow w_{1} \Rightarrow w_{2} \Rightarrow \cdots \Rightarrow w_{n} \Rightarrow w
$$

is a derivation of the sentence $w$. The strings $S, w_{1}, w_{2}, \ldots, w_{n}$, which contain variables as well as terminals, are called sentential forms of the derivation.

By applying the production rules in a different order, a given grammar can normally generate many strings. The set of all such terminal strings is the language defined or generated by the grammar.

## Definition 1.2

Let $G=(V, T, S, P)$ be a grammar. Then the set

$$
L(G)=\left\{w \in T^{*}: S \stackrel{*}{\Rightarrow} w\right\}
$$

is the language generated by $G$.
If $w \in L(G)$ then the sequence

$$
S \Rightarrow w_{1} \Rightarrow w_{2} \Rightarrow \cdots \Rightarrow w_{n} \Rightarrow w
$$

is a derivation of the sentence $w$.

By applying the production rules in a different order, a given grammar can normally generate many strings. The set of all such terminal strings is the language defined or generated by the grammar.

## Definition 1.2

Let $G=(V, T, S, P)$ be a grammar. Then the set

$$
L(G)=\left\{w \in T^{*}: S \stackrel{*}{\Rightarrow} w\right\}
$$

is the language generated by $G$.
If $w \in L(G)$ then the sequence

$$
S \Rightarrow w_{1} \Rightarrow w_{2} \Rightarrow \cdots \Rightarrow w_{n} \Rightarrow w
$$

is a derivation of the sentence $w$. The strings $S, w_{1}, w_{2}, \ldots, w_{n}$, which contain variables as well as terminals, are called sentential forms of the derivation.

By applying the production rules in a different order, a given grammar can normally generate many strings. The set of all such terminal strings is the language defined or generated by the grammar.

## Definition 1.2

Let $G=(V, T, S, P)$ be a grammar. Then the set

$$
L(G)=\left\{w \in T^{*}: S \stackrel{*}{\Rightarrow} w\right\}
$$

is the language generated by $G$.
If $w \in L(G)$ then the sequence

$$
S \Rightarrow w_{1} \Rightarrow w_{2} \Rightarrow \cdots \Rightarrow w_{n} \Rightarrow w
$$

is a derivation of the sentence $w$. The strings $S, w_{1}, w_{2}, \ldots, w_{n}$, which contain variables as well as terminals, are called sentential forms of the derivation.

## Example 1.11

## Example 1.11

## Consider the grammar

$$
G=(\{S\},\{a, b\}, S, P),
$$

with $P$ given by

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

Then

$$
S \Rightarrow a S b \Rightarrow a a S b b \Rightarrow a a b b,
$$

so we can write

$$
S \stackrel{*}{\Rightarrow} a a b b .
$$

The string $a a b b$ is a sentence in the language generated by $G$, while the string $a a S b b$ is a sentential form.

## Example 1.11

Consider the grammar

$$
G=(\{S\},\{a, b\}, S, P),
$$

with $P$ given by


Then

$$
S \Rightarrow a S b \Rightarrow a a S b b \Rightarrow a a b b,
$$

so we can write

$$
S \stackrel{*}{\Rightarrow} a a b b .
$$

The string $a a b b$ is a sentence in the language generated by $G$, while the string $a a S b b$ is a sentential form.

## Example 1.11

Consider the grammar

$$
G=(\{S\},\{a, b\}, S, P)
$$

with $P$ given by


Then

$$
S \Rightarrow a S b \Rightarrow a a S b b \Rightarrow a a b b
$$

so we can write

$$
S \stackrel{*}{\Rightarrow} a a b b .
$$

The string $a a b b$ is a sentence in the language generated by $G$, while the string $a a S b b$ is a sentential form.

## Example 1.11

Consider the grammar

$$
G=(\{S\},\{a, b\}, S, P)
$$

with $P$ given by


Then $S \Rightarrow a S b \Rightarrow a a S b b \Rightarrow a a b b$,
so we can write

$$
S \stackrel{*}{\Rightarrow} a a b b .
$$

The string $a a b b$ is a sentence in the language generated by $G$, while the string $a a S b b$ is a sentential form.

## Example 1.11

Consider the grammar

$$
G=(\{S\},\{a, b\}, S, P)
$$

with $P$ given by

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

## Then

$$
S \Rightarrow a S b \Rightarrow a a S b b \Rightarrow a a b b,
$$

so we can write

$$
S \stackrel{*}{\Rightarrow} a a b b .
$$

The string $a a b b$ is a sentence in the language generated by $G$, while the string $a a S b b$ is a sentential form.

## Example 1.11

Consider the grammar

$$
G=(\{S\},\{a, b\}, S, P)
$$

with $P$ given by

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

Then

$$
S \Rightarrow a S b \Rightarrow a a S b b \Rightarrow a a b b
$$

so we can write

$$
S \stackrel{*}{\Rightarrow} a a b b .
$$

The string $a a b b$ is a sentence in the language generated by $G$, while the string $a a S b b$ is a sentential form.

## Example 1.11

Consider the grammar

$$
G=(\{S\},\{a, b\}, S, P)
$$

with $P$ given by

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

Then

$$
S \Rightarrow a S b \Rightarrow a a S b b \Rightarrow a a b b
$$

so we can write
$S \stackrel{*}{\Rightarrow} a a b b$.
The string $a a b b$ is a sentence in the language generated by $G$, while the string $a a S b b$ is a sentential form.

## Example 1.11

Consider the grammar

$$
G=(\{S\},\{a, b\}, S, P)
$$

with $P$ given by

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

Then

$$
S \Rightarrow a S b \Rightarrow a a S b b \Rightarrow a a b b
$$

so we can write

$$
S \stackrel{*}{\Rightarrow} a a b b .
$$

The string $a a b b$ is a sentence in the language generated by $G$, while the string $a a S b b$ is a sentential form.

## Example 1.11

Consider the grammar

$$
G=(\{S\},\{a, b\}, S, P)
$$

with $P$ given by

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

Then

$$
S \Rightarrow a S b \Rightarrow a a S b b \Rightarrow a a b b
$$

so we can write

$$
S \stackrel{*}{\Rightarrow} a a b b .
$$

The string $a a b b$ is a sentence in the language generated by $G$, while the string $a a S b b$ is a sentential form.

## Example 1.11

Consider the grammar

$$
G=(\{S\},\{a, b\}, S, P)
$$

with $P$ given by

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

Then

$$
S \Rightarrow a S b \Rightarrow a a S b b \Rightarrow a a b b
$$

so we can write

$$
S \stackrel{*}{\Rightarrow} a a b b .
$$

The string $a a b b$ is a sentence in the language generated by $G$, while the string $a a S b b$ is a sentential form.

## Example 1.11

Consider the grammar

$$
G=(\{S\},\{a, b\}, S, P)
$$

with $P$ given by

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

Then

$$
S \Rightarrow a S b \Rightarrow a a S b b \Rightarrow a a b b
$$

so we can write

$$
S \stackrel{*}{\Rightarrow} a a b b .
$$

The string $a a b b$ is a sentence in the language generated by $G$, while the string $a a S b b$ is a sentential form.

Example 1.11 (continuation)

## 2 THREE BASIC CONCEPTS: Grammars

## Example 1.11 (continuation)

A grammar $G$ completely defines $L(G)$, but it may not be easy to get a very explicit description of the language from the grammar. Here, however, the answer is fairly clear. It is not hard to conjecture that

$$
L(G)=\left\{a^{n} b^{n}: n \geq 0\right\}
$$

and it is easy to prove it. If we notice that the rule $S \rightarrow a S b$ is recursive, a proof by induction readily suggests itself. We first show that all sentential forms must have the form

$$
\begin{equation*}
w_{i}=a^{i} S b^{i} \tag{2}
\end{equation*}
$$

Suppose that condition (2) holds for all sentential forms $w_{i}$ of length $2 i+1$ or less. To get another sentential form (which is not a sentence), we can only apply the production $S \rightarrow a S b$. This gets us

$$
a^{i} S b^{i} \Rightarrow a^{i+1} S b^{i+1}
$$

so that every sentential form of length $2 i+3$ is also of the form (2). Since (2) is obviously true for $i=1$, it holds by induction for all $i$. Finally, to get a sentence, we must apply the production $S \rightarrow \lambda$, and we see that
$S \Rightarrow a^{n} S b^{n} \Rightarrow a^{n} b^{n}$
represents all possible derivations. Thus, $G$ can derive only strings of the form $a^{n} b^{n}$
We also have to show that all strings of this form can be derived. This is easy; we simply apply $S \rightarrow a S b$ as many times as needed, followed by $S \rightarrow \lambda$.

## 2 THREE BASIC CONCEPTS: Grammars

## Example 1.11 (continuation)

A grammar $G$ completely defines $L(G)$, but it may not be easy to get a very explicit description of the language from the grammar. Here, however, the answer is fairly clear. It is not hard to conjecture that

$$
L(G)=\left\{a^{n} b^{n}: n \geq 0\right\}
$$

and it is easy to prove it. If we notice that the rule $S \rightarrow a S b$ is recursive, a proof by induction readily suggests itself. We first show that all sentential forms must have the form

$$
\begin{equation*}
w_{i}=a^{i} S b^{i} \tag{2}
\end{equation*}
$$

Suppose that condition (2) holds for all sentential forms $w_{i}$ of length $2 i+1$ or less. To get another sentential form (which is not a sentence), we can only apply the production $S \rightarrow a S b$. This gets us

so that every sentential form of length $2 i+3$ is also of the form (2). Since (2) is obviously true for $i=1$, it holds by induction for all $i$. Finally, to get a sentence, we must apply the production $S \rightarrow \lambda$, and we see that
$S \Rightarrow a^{n} S b^{n} \Rightarrow a^{n} b^{n}$
represents all possible derivations. Thus, $G$ can derive only strings of the form $a^{n} b^{n}$
We also have to show that all strings of this form can be derived. This is easy; we simply apply $S \rightarrow a S b$ as many times as needed, followed by $S \rightarrow \lambda$.

## Example 1.11 (continuation)

A grammar $G$ completely defines $L(G)$, but it may not be easy to get a very explicit description of the language from the grammar.
answer is fairly clear. It is not hard to conjecture that
$L(G)=\left\{a^{n} b^{n}: n \geq 0\right\}$,
and it is easy to prove it. If we notice that the rule $S \rightarrow a S b$ is recursive, a
proof by induction readily suggests itself. We first show that all sentential forms
must have the form

\[\)| $w_{i}=a^{i} S b^{i} .$ |
| :--- |

\]

Suppose that condition (2) holds for all sentential forms $w_{i}$ of length $2 i+1$ or
less. To get another sentential form (which is not a sentence), we can only
apply the production $S \rightarrow a S b$. This gets us

$$
a^{i} S b^{i} \Rightarrow a^{i+1} S b^{i+1}
$$

so that every sentential form of length $2 i+3$ is also of the form (2). Since (2)
is obviously true for $i=1$, it holds by induction for all $i$. Finally, to get a
sentence, we must apply the production $S \rightarrow \lambda$, and we see that

$$
S \Rightarrow a^{n} S b^{n} \Rightarrow a^{n} b^{n}
$$

represents all possible derivations. Thus, $G$ can derive only strings of the form
$a^{n} b^{n}$.
We also have to show that all strings of this form can be derived. This is easy;
we simply apply $S \rightarrow a S b$ as many times as needed, followed by $S \rightarrow \lambda$.

## Example 1.11 (continuation)

A grammar $G$ completely defines $L(G)$, but it may not be easy to get a very explicit description of the language from the grammar. Here, however, the answer is fairly clear. It is not hard to conjecture that
and it is easy to prove it. If we notice that the rule $S \rightarrow a S b$ is recursive, a proof by induction readily suggests itself. We first show that all sentential forms must have the form

$$
w_{i}=a^{2} S b^{2} .
$$

Suppose that condition (2) holds for all sentential forms $w_{i}$ of length $2 i+1$ or less. To get another sentential form (which is not a sentence), we can only apply the production $S \rightarrow a S b$. This gets us
so that every sentential form of length $2 i+3$ is also of the form (2). Since (2) is obviously true for $i=1$, it holds by induction for all $i$. Finally, to get a sentence, we must apply the production $S \rightarrow \lambda$, and we see that
represents all possible derivations. Thus, $G$ can derive only strings of the form $a^{n} b^{n}$
We also have to show that all strings of this form can be derived. This is easy; we simply apply $S \rightarrow a S b$ as many times as needed, followed by $S \rightarrow \lambda$.

## Example 1.11 (continuation)

A grammar $G$ completely defines $L(G)$, but it may not be easy to get a very explicit description of the language from the grammar. Here, however, the answer is fairly clear. It is not hard to conjecture that
and it is easy to prove it. If we notice that the rule $S \rightarrow a S b$ is recursive, a proof by induction readily suggests itself. We first show that all sentential forms must have the form

$$
w_{i}=a^{2} S b^{2} \text {. }
$$

Suppose that condition (2) holds for all sentential forms $w_{i}$ of length $2 i+1$ or
less. To get another sentential form (which is not a sentence), we can only
apply the production $S \rightarrow a S b$. This gets us

$$
a^{i} S b^{i} \Rightarrow a^{i+1} S b^{i+1}
$$

so that every sentential form of length $2 i+3$ is also of the form (2). Since (2)
is obviously true for $i=1$, it holds by induction for all $i$. Finally, to get a
sentence, we must apply the production $S \rightarrow \lambda$, and we see that

$$
S \Rightarrow a^{n} S b^{n} \Rightarrow a^{n} b^{n}
$$

represents all possible derivations. Thus, $G$ can derive only strings of the form
$a^{n} b^{n}$.
We also have to show that all strings of this form can be derived. This is easy;
we simply apply $S \rightarrow a S b$ as many times as needed, followed by $S \rightarrow \lambda$.

## Example 1.11 (continuation)

A grammar $G$ completely defines $L(G)$, but it may not be easy to get a very explicit description of the language from the grammar. Here, however, the answer is fairly clear. It is not hard to conjecture that

$$
L(G)=\left\{a^{n} b^{n}: n \geq 0\right\},
$$

and it is easy to prove it. If we notice that the rule $S \rightarrow a S b$ is recursive, a proof by induction readily suggests itself. We first show that all sentential forms must have the form

Suppose that condition (2) holds for all sentential forms $w_{i}$ of length $2 i+1$ or less. To get another sentential form (which is not a sentence), we can only apply the production $S \rightarrow a S b$. This gets us
so that every sentential form of length $2 i+3$ is also of the form (2). Since (2) is obviously true for $i=1$, it holds by induction for all $i$. Finally, to get a sentence, we must apply the production $S \rightarrow \lambda$, and we see that
represents all possible derivations. Thus, $G$ can derive only strings of the form $a^{n} b^{n}$
We also have to show that all strings of this form can be derived. This is easy; we simply apply $S \rightarrow a S b$ as many times as needed, followed by $S \rightarrow \lambda$. $\quad$.

## Example 1.11 (continuation)

A grammar $G$ completely defines $L(G)$, but it may not be easy to get a very explicit description of the language from the grammar. Here, however, the answer is fairly clear. It is not hard to conjecture that

$$
L(G)=\left\{a^{n} b^{n}: n \geq 0\right\}
$$

and it is easy to prove it.
proof by induction readily suggests itself. We first show that all sentential forms must have the form

Suppose that condition (2) holds for all sentential forms $w_{i}$ of length $2 i+1$ or less. To get another sentential form (which is not a sentence), we can only apply the production $S \rightarrow a S b$. This gets us
so that every sentential form of length $2 i+3$ is also of the form (2). Since (2) is obviously true for $i=1$, it holds by induction for all $i$. Finally, to get a sentence, we must apply the production $S \rightarrow \lambda$, and we see that
represents all possible derivations. Thus, $G$ can derive only strings of the form $a^{n} b^{n}$
We also have to show that all strings of this form can be derived. This is easy; we simply apply $S \rightarrow a S b$ as many times as needed, followed by $S \rightarrow \lambda$.

## Example 1.11 (continuation)

A grammar $G$ completely defines $L(G)$, but it may not be easy to get a very explicit description of the language from the grammar. Here, however, the answer is fairly clear. It is not hard to conjecture that

$$
L(G)=\left\{a^{n} b^{n}: n \geq 0\right\}
$$

and it is easy to prove it. If we notice that the rule $S \rightarrow a S b$ is recursive, proof by induction readily suggests itself. We first show that all sentential forms must have the form

Suppose that condition (2) holds for all sentential forms $w_{i}$ of length $2 i+1$ or less. To get another sentential form (which is not a sentence), we can only apply the production $S \rightarrow a S b$. This gets us
so that every sentential form of length $2 i+3$ is also of the form (2). Since (2) is obviously true for $i=1$, it holds by induction for all $i$. Finally, to get a sentence, we must apply the production $S \rightarrow \lambda$, and we see that
represents all possible derivations. Thus, $G$ can derive only strings of the form $a^{n} b$
We also have to show that all strings of this form can be derived. This is easy; we simply apply $S \rightarrow a S b$ as many times as needed, followed by $S \rightarrow \lambda$.

## Example 1.11 (continuation)

A grammar $G$ completely defines $L(G)$, but it may not be easy to get a very explicit description of the language from the grammar. Here, however, the answer is fairly clear. It is not hard to conjecture that

$$
L(G)=\left\{a^{n} b^{n}: n \geq 0\right\},
$$

and it is easy to prove it. If we notice that the rule $S \rightarrow a S b$ is recursive, a proof by induction readily suggests itself.
must have the form
Suppose that condition (2) holds for all sentential forms $w_{i}$ of length $2 i+1$ or less. To get another sentential form (which is not a sentence), we can only apply the production $S \rightarrow a S b$. This gets us
so that every sentential form of length $2 i+3$ is also of the form (2). Since (2) is obviously true for $i=1$, it holds by induction for all $i$. Finally, to get a sentence, we must apply the production $S \rightarrow \lambda$, and we see that
represents all possible derivations. Thus, $G$ can derive only strings of the form
We also have to show that all strings of this form can be derived. This is easy;
we simply apply $S \rightarrow a S b$ as many times as needed, followed by $S \rightarrow \lambda$.

## Example 1.11 (continuation)

A grammar $G$ completely defines $L(G)$, but it may not be easy to get a very explicit description of the language from the grammar. Here, however, the answer is fairly clear. It is not hard to conjecture that

$$
L(G)=\left\{a^{n} b^{n}: n \geq 0\right\},
$$

and it is easy to prove it. If we notice that the rule $S \rightarrow a S b$ is recursive, a proof by induction readily suggests itself. We first show that all sentential forms must have the form

Suppose that condition (2) holds for all sentential forms $w_{i}$ of length $2 i+1$ or less. To get another sentential form (which is not a sentence), we can only apply the production $S \rightarrow a S b$. This gets us
so that every sentential form of length $2 i+3$ is also of the form (2). Since (2) is obviously true for $i=1$, it holds by induction for all $i$. Finally, to get a sentence, we must apply the production $S \rightarrow \lambda$, and we see that
represents all possible derivations. Thus, $G$ can derive only strings of the form
We also have to show that all strings of this form can be derived. This is easy;
we simply apply $S \rightarrow a S b$ as many times as needed, followed by $S \rightarrow \lambda$.

## Example 1.11 (continuation)

A grammar $G$ completely defines $L(G)$, but it may not be easy to get a very explicit description of the language from the grammar. Here, however, the answer is fairly clear. It is not hard to conjecture that

$$
L(G)=\left\{a^{n} b^{n}: n \geq 0\right\},
$$

and it is easy to prove it. If we notice that the rule $S \rightarrow a S b$ is recursive, a proof by induction readily suggests itself. We first show that all sentential forms must have the form

$$
\begin{equation*}
w_{i}=a^{i} S b^{i} . \tag{2}
\end{equation*}
$$

Suppose that condition (2) holds for all sentential forms $w_{i}$ of length $2 i+1$ or
less. To get another sentential form (which is not a sentence), we can only
apply the production $S \rightarrow a S b$. This gets us
so that every sentential form of length $2 i+3$ is also of the form (2). Since (2) is obviously true for $i=1$, it holds by induction for all $i$. Finally, to get a sentence, we must apply the production $S \rightarrow \lambda$, and we see that
represents all possilite derivations. Thus, $G$ can derive only strings of the form
We also have to show that all strings of this form can be derived. This is easy;
we simply apply $S \rightarrow a S b$ as many times as needed, followed by $S \rightarrow \lambda$.

## Example 1.11 (continuation)

A grammar $G$ completely defines $L(G)$, but it may not be easy to get a very explicit description of the language from the grammar. Here, however, the answer is fairly clear. It is not hard to conjecture that

$$
L(G)=\left\{a^{n} b^{n}: n \geq 0\right\}
$$

and it is easy to prove it. If we notice that the rule $S \rightarrow a S b$ is recursive, a proof by induction readily suggests itself. We first show that all sentential forms must have the form

$$
\begin{equation*}
w_{i}=a^{i} S b^{i} \tag{2}
\end{equation*}
$$

Suppose that condition (2) holds for all sentential forms $w_{i}$ of length $2 i+1$ or less.
apply the production $S \rightarrow a S b$. This gets us
so that every sentential form of length $2 i+3$ is also of the form (2). Since (2) is obviously true for $i=1$, it holds by induction for all $i$. Finally, to get a sentence, we must apply the production $S \rightarrow \lambda$, and we see that
represents all possible derivations. Thus, $G$ can derive only strings of the form We also have to show that all strings of this form can be derived. This is easy; we simply apply $S \rightarrow a S b$ as many times as needed, followed by $S \rightarrow \lambda$.

## Example 1.11 (continuation)

A grammar $G$ completely defines $L(G)$, but it may not be easy to get a very explicit description of the language from the grammar. Here, however, the answer is fairly clear. It is not hard to conjecture that

$$
L(G)=\left\{a^{n} b^{n}: n \geq 0\right\}
$$

and it is easy to prove it. If we notice that the rule $S \rightarrow a S b$ is recursive, a proof by induction readily suggests itself. We first show that all sentential forms must have the form

$$
\begin{equation*}
w_{i}=a^{i} S b^{i} \tag{2}
\end{equation*}
$$

Suppose that condition (2) holds for all sentential forms $w_{i}$ of length $2 i+1$ or less. To get another sentential form (which is not a sentence),
apply the production $S \rightarrow a S b$. This gets us
so that every sentential form of length $2 i+3$ is also of the form (2). Since (2) is obviously true for $i=1$, it holds by induction for all $i$. Finally, to get a sentence, we must apply the production $S \rightarrow \lambda$, and we see that
represents all possible derivations. Thus, $G$ can derive only strings of the form We also have to show that all strings of this form can be derived. This is easy; we simply apply $S \rightarrow a S b$ as many times as needed, followed by $S \rightarrow \lambda$.

## Example 1.11 (continuation)

A grammar $G$ completely defines $L(G)$, but it may not be easy to get a very explicit description of the language from the grammar. Here, however, the answer is fairly clear. It is not hard to conjecture that

$$
L(G)=\left\{a^{n} b^{n}: n \geq 0\right\}
$$

and it is easy to prove it. If we notice that the rule $S \rightarrow a S b$ is recursive, a proof by induction readily suggests itself. We first show that all sentential forms must have the form

$$
\begin{equation*}
w_{i}=a^{i} S b^{i} \tag{2}
\end{equation*}
$$

Suppose that condition (2) holds for all sentential forms $w_{i}$ of length $2 i+1$ or less. To get another sentential form (which is not a sentence), we can only apply the production $S \rightarrow a S b$.
so that every sentential form of length $2 i+3$ is also of the form (2). Since (2) is obviously true for $i=1$, it holds by induction for all $i$. Finally, to get a sentence, we must apply the production $S \rightarrow \lambda$, and we see that
represents all possible derivations. Thus, $G$ can derive only strings of the form We also have to show that all strings of this form can be derived. This is easy; we simply apply $S \rightarrow a S b$ as many times as needed, followed by $S \rightarrow \lambda$.

## Example 1.11 (continuation)

A grammar $G$ completely defines $L(G)$, but it may not be easy to get a very explicit description of the language from the grammar. Here, however, the answer is fairly clear. It is not hard to conjecture that

$$
L(G)=\left\{a^{n} b^{n}: n \geq 0\right\}
$$

and it is easy to prove it. If we notice that the rule $S \rightarrow a S b$ is recursive, a proof by induction readily suggests itself. We first show that all sentential forms must have the form

$$
\begin{equation*}
w_{i}=a^{i} S b^{i} \tag{2}
\end{equation*}
$$

Suppose that condition (2) holds for all sentential forms $w_{i}$ of length $2 i+1$ or less. To get another sentential form (which is not a sentence), we can only apply the production $S \rightarrow a S b$. This gets us
so that every sentential form of length $2 i+3$ is also of the form (2). Since (2) is obviously true for $i=1$, it holds by induction for all $i$. Finally, to get a sentence, we must apply the production $S \rightarrow \lambda$, and we see that
represents all possible derivations. Thus, $G$ can derive only strings of the form We also have to show that all strings of this form can be derived. This is easy; we simply apply $S \rightarrow a S b$ as many times as needed, followed by $S \rightarrow \lambda$.

## Example 1.11 (continuation)

A grammar $G$ completely defines $L(G)$, but it may not be easy to get a very explicit description of the language from the grammar. Here, however, the answer is fairly clear. It is not hard to conjecture that

$$
L(G)=\left\{a^{n} b^{n}: n \geq 0\right\}
$$

and it is easy to prove it. If we notice that the rule $S \rightarrow a S b$ is recursive, a proof by induction readily suggests itself. We first show that all sentential forms must have the form

$$
\begin{equation*}
w_{i}=a^{i} S b^{i} \tag{2}
\end{equation*}
$$

Suppose that condition (2) holds for all sentential forms $w_{i}$ of length $2 i+1$ or less. To get another sentential form (which is not a sentence), we can only apply the production $S \rightarrow a S b$. This gets us

$$
a^{i} S b^{i} \Rightarrow a^{i+1} S b^{i+1}
$$

so that every sentential form of length $2 i+3$ is also of the form (2). Since (2) is obviously true for $i=1$, it holds by induction for all $i$. Finally, to get a sentence, we must apply the production $S \rightarrow \lambda$, and we see that
represents all possible derivations. Thus, $G$ can derive only strings of the form We also have to show that all strings of this form can be derived. This is easy; we simply apply $S \rightarrow a S b$ as many times as needed, followed by $S \rightarrow \lambda$.

## Example 1.11 (continuation)

A grammar $G$ completely defines $L(G)$, but it may not be easy to get a very explicit description of the language from the grammar. Here, however, the answer is fairly clear. It is not hard to conjecture that

$$
L(G)=\left\{a^{n} b^{n}: n \geq 0\right\}
$$

and it is easy to prove it. If we notice that the rule $S \rightarrow a S b$ is recursive, a proof by induction readily suggests itself. We first show that all sentential forms must have the form

$$
\begin{equation*}
w_{i}=a^{i} S b^{i} \tag{2}
\end{equation*}
$$

Suppose that condition (2) holds for all sentential forms $w_{i}$ of length $2 i+1$ or less. To get another sentential form (which is not a sentence), we can only apply the production $S \rightarrow a S b$. This gets us

$$
a^{i} S b^{i} \Rightarrow a^{i+1} S b^{i+1}
$$

so that every sentential form of length $2 i+3$ is also of the form (2). sentence, we must apply the production $S \rightarrow \lambda$, and we see that
represents all possible derivations. Thus, $G$ can derive only strings of the form We also have to show that all strings of this form can be derived. This is easy; we simply apply $S \rightarrow a S b$ as many times as needed, followed by $S \rightarrow \lambda$.

## Example 1.11 (continuation)

A grammar $G$ completely defines $L(G)$, but it may not be easy to get a very explicit description of the language from the grammar. Here, however, the answer is fairly clear. It is not hard to conjecture that

$$
L(G)=\left\{a^{n} b^{n}: n \geq 0\right\}
$$

and it is easy to prove it. If we notice that the rule $S \rightarrow a S b$ is recursive, a proof by induction readily suggests itself. We first show that all sentential forms must have the form

$$
\begin{equation*}
w_{i}=a^{i} S b^{i} \tag{2}
\end{equation*}
$$

Suppose that condition (2) holds for all sentential forms $w_{i}$ of length $2 i+1$ or less. To get another sentential form (which is not a sentence), we can only apply the production $S \rightarrow a S b$. This gets us

$$
a^{i} S b^{i} \Rightarrow a^{i+1} S b^{i+1}
$$

so that every sentential form of length $2 i+3$ is also of the form (2). Since (2) is obviously true for $i=1$, it holds by induction for all $i$.
sentence, we must apply the production $S \rightarrow \lambda$, and we see that
represents all possible derivations. Thus, $G$ can derive only strings of the form

We also have to show that all strings of this form can be derived. This is easy; we simply apply $S \rightarrow a S b$ as many times as needed, followed by $S \rightarrow \lambda$.

## Example 1.11 (continuation)

A grammar $G$ completely defines $L(G)$, but it may not be easy to get a very explicit description of the language from the grammar. Here, however, the answer is fairly clear. It is not hard to conjecture that

$$
L(G)=\left\{a^{n} b^{n}: n \geq 0\right\}
$$

and it is easy to prove it. If we notice that the rule $S \rightarrow a S b$ is recursive, a proof by induction readily suggests itself. We first show that all sentential forms must have the form

$$
\begin{equation*}
w_{i}=a^{i} S b^{i} \tag{2}
\end{equation*}
$$

Suppose that condition (2) holds for all sentential forms $w_{i}$ of length $2 i+1$ or less. To get another sentential form (which is not a sentence), we can only apply the production $S \rightarrow a S b$. This gets us

$$
a^{i} S b^{i} \Rightarrow a^{i+1} S b^{i+1}
$$

so that every sentential form of length $2 i+3$ is also of the form (2). Since (2) is obviously true for $i=1$, it holds by induction for all $i$. Finally, to get a sentence, we must apply the production $S \rightarrow \lambda$, and we see that
represents all possible derivations. Thus, $G$ can derive only strings of the form

We also have to show that all strings of this form can be derived. This is easy; we simply apply $S \rightarrow a S b$ as many times as needed, followed by $S \rightarrow \lambda$.

## Example 1.11 (continuation)

A grammar $G$ completely defines $L(G)$, but it may not be easy to get a very explicit description of the language from the grammar. Here, however, the answer is fairly clear. It is not hard to conjecture that

$$
L(G)=\left\{a^{n} b^{n}: n \geq 0\right\}
$$

and it is easy to prove it. If we notice that the rule $S \rightarrow a S b$ is recursive, a proof by induction readily suggests itself. We first show that all sentential forms must have the form

$$
\begin{equation*}
w_{i}=a^{i} S b^{i} \tag{2}
\end{equation*}
$$

Suppose that condition (2) holds for all sentential forms $w_{i}$ of length $2 i+1$ or less. To get another sentential form (which is not a sentence), we can only apply the production $S \rightarrow a S b$. This gets us

$$
a^{i} S b^{i} \Rightarrow a^{i+1} S b^{i+1}
$$

so that every sentential form of length $2 i+3$ is also of the form (2). Since (2) is obviously true for $i=1$, it holds by induction for all $i$. Finally, to get a sentence, we must apply the production $S \rightarrow \lambda$, and we see that
represents all possible derivations. Thus, $G$ can derive only strings of the form

We also have to show that all strings of this form can be derived. This is easy; we simply apply $S \rightarrow a S b$ as many times as needed, followed by $S \rightarrow \lambda$.

## Example 1.11 (continuation)

A grammar $G$ completely defines $L(G)$, but it may not be easy to get a very explicit description of the language from the grammar. Here, however, the answer is fairly clear. It is not hard to conjecture that

$$
L(G)=\left\{a^{n} b^{n}: n \geq 0\right\}
$$

and it is easy to prove it. If we notice that the rule $S \rightarrow a S b$ is recursive, a proof by induction readily suggests itself. We first show that all sentential forms must have the form

$$
\begin{equation*}
w_{i}=a^{i} S b^{i} \tag{2}
\end{equation*}
$$

Suppose that condition (2) holds for all sentential forms $w_{i}$ of length $2 i+1$ or less. To get another sentential form (which is not a sentence), we can only apply the production $S \rightarrow a S b$. This gets us

$$
a^{i} S b^{i} \Rightarrow a^{i+1} S b^{i+1}
$$

so that every sentential form of length $2 i+3$ is also of the form (2). Since (2) is obviously true for $i=1$, it holds by induction for all $i$. Finally, to get a sentence, we must apply the production $S \rightarrow \lambda$, and we see that

$$
S \stackrel{*}{\Rightarrow} a^{n} S b^{n} \Rightarrow a^{n} b^{n}
$$

represents all possible derivations. Thus, $G$ can derive only strings of the form

We also have to show that all strings of this form can be derived. This is easy; we simply apply $S \rightarrow a S b$ as many times as needed, followed by $S \rightarrow \lambda$.

## Example 1.11 (continuation)

A grammar $G$ completely defines $L(G)$, but it may not be easy to get a very explicit description of the language from the grammar. Here, however, the answer is fairly clear. It is not hard to conjecture that

$$
L(G)=\left\{a^{n} b^{n}: n \geq 0\right\}
$$

and it is easy to prove it. If we notice that the rule $S \rightarrow a S b$ is recursive, a proof by induction readily suggests itself. We first show that all sentential forms must have the form

$$
\begin{equation*}
w_{i}=a^{i} S b^{i} \tag{2}
\end{equation*}
$$

Suppose that condition (2) holds for all sentential forms $w_{i}$ of length $2 i+1$ or less. To get another sentential form (which is not a sentence), we can only apply the production $S \rightarrow a S b$. This gets us

$$
a^{i} S b^{i} \Rightarrow a^{i+1} S b^{i+1}
$$

so that every sentential form of length $2 i+3$ is also of the form (2). Since (2) is obviously true for $i=1$, it holds by induction for all $i$. Finally, to get a sentence, we must apply the production $S \rightarrow \lambda$, and we see that

$$
S \stackrel{*}{\Rightarrow} a^{n} S b^{n} \Rightarrow a^{n} b^{n}
$$

represents all possible derivations.

$$
\text { Thus, } G \text { can derive only strings of the form }
$$

We also have to show that all strings of this form can be derived. This is easy; we simply apply $S \rightarrow a S b$ as many times as needed, followed by $S \rightarrow \lambda$.

## Example 1.11 (continuation)

A grammar $G$ completely defines $L(G)$, but it may not be easy to get a very explicit description of the language from the grammar. Here, however, the answer is fairly clear. It is not hard to conjecture that

$$
L(G)=\left\{a^{n} b^{n}: n \geq 0\right\}
$$

and it is easy to prove it. If we notice that the rule $S \rightarrow a S b$ is recursive, a proof by induction readily suggests itself. We first show that all sentential forms must have the form

$$
\begin{equation*}
w_{i}=a^{i} S b^{i} \tag{2}
\end{equation*}
$$

Suppose that condition (2) holds for all sentential forms $w_{i}$ of length $2 i+1$ or less. To get another sentential form (which is not a sentence), we can only apply the production $S \rightarrow a S b$. This gets us

$$
a^{i} S b^{i} \Rightarrow a^{i+1} S b^{i+1}
$$

so that every sentential form of length $2 i+3$ is also of the form (2). Since (2) is obviously true for $i=1$, it holds by induction for all $i$. Finally, to get a sentence, we must apply the production $S \rightarrow \lambda$, and we see that

$$
S \stackrel{*}{\Rightarrow} a^{n} S b^{n} \Rightarrow a^{n} b^{n}
$$

represents all possible derivations. Thus, $G$ can derive only strings of the form $a^{n} b^{n}$.
We also have to show that all strings of this form can be derived. This is easy we simply apply $S \rightarrow a S b$ as many times as needed, followed by $S \rightarrow \lambda$.

## Example 1.11 (continuation)

A grammar $G$ completely defines $L(G)$, but it may not be easy to get a very explicit description of the language from the grammar. Here, however, the answer is fairly clear. It is not hard to conjecture that

$$
L(G)=\left\{a^{n} b^{n}: n \geq 0\right\}
$$

and it is easy to prove it. If we notice that the rule $S \rightarrow a S b$ is recursive, a proof by induction readily suggests itself. We first show that all sentential forms must have the form

$$
\begin{equation*}
w_{i}=a^{i} S b^{i} \tag{2}
\end{equation*}
$$

Suppose that condition (2) holds for all sentential forms $w_{i}$ of length $2 i+1$ or less. To get another sentential form (which is not a sentence), we can only apply the production $S \rightarrow a S b$. This gets us

$$
a^{i} S b^{i} \Rightarrow a^{i+1} S b^{i+1}
$$

so that every sentential form of length $2 i+3$ is also of the form (2). Since (2) is obviously true for $i=1$, it holds by induction for all $i$. Finally, to get a sentence, we must apply the production $S \rightarrow \lambda$, and we see that

$$
S \stackrel{*}{\Rightarrow} a^{n} S b^{n} \Rightarrow a^{n} b^{n}
$$

represents all possible derivations. Thus, $G$ can derive only strings of the form $a^{n} b^{n}$.
We also have to show that all strings of this form can be derived.

## Example 1.11 (continuation)

A grammar $G$ completely defines $L(G)$, but it may not be easy to get a very explicit description of the language from the grammar. Here, however, the answer is fairly clear. It is not hard to conjecture that

$$
L(G)=\left\{a^{n} b^{n}: n \geq 0\right\}
$$

and it is easy to prove it. If we notice that the rule $S \rightarrow a S b$ is recursive, a proof by induction readily suggests itself. We first show that all sentential forms must have the form

$$
\begin{equation*}
w_{i}=a^{i} S b^{i} \tag{2}
\end{equation*}
$$

Suppose that condition (2) holds for all sentential forms $w_{i}$ of length $2 i+1$ or less. To get another sentential form (which is not a sentence), we can only apply the production $S \rightarrow a S b$. This gets us

$$
a^{i} S b^{i} \Rightarrow a^{i+1} S b^{i+1}
$$

so that every sentential form of length $2 i+3$ is also of the form (2). Since (2) is obviously true for $i=1$, it holds by induction for all $i$. Finally, to get a sentence, we must apply the production $S \rightarrow \lambda$, and we see that

$$
S \stackrel{*}{\Rightarrow} a^{n} S b^{n} \Rightarrow a^{n} b^{n}
$$

represents all possible derivations. Thus, $G$ can derive only strings of the form $a^{n} b^{n}$.
We also have to show that all strings of this form can be derived. This is easy;

## Example 1.11 (continuation)

A grammar $G$ completely defines $L(G)$, but it may not be easy to get a very explicit description of the language from the grammar. Here, however, the answer is fairly clear. It is not hard to conjecture that

$$
L(G)=\left\{a^{n} b^{n}: n \geq 0\right\}
$$

and it is easy to prove it. If we notice that the rule $S \rightarrow a S b$ is recursive, a proof by induction readily suggests itself. We first show that all sentential forms must have the form

$$
\begin{equation*}
w_{i}=a^{i} S b^{i} \tag{2}
\end{equation*}
$$

Suppose that condition (2) holds for all sentential forms $w_{i}$ of length $2 i+1$ or less. To get another sentential form (which is not a sentence), we can only apply the production $S \rightarrow a S b$. This gets us

$$
a^{i} S b^{i} \Rightarrow a^{i+1} S b^{i+1}
$$

so that every sentential form of length $2 i+3$ is also of the form (2). Since (2) is obviously true for $i=1$, it holds by induction for all $i$. Finally, to get a sentence, we must apply the production $S \rightarrow \lambda$, and we see that

$$
S \stackrel{*}{\Rightarrow} a^{n} S b^{n} \Rightarrow a^{n} b^{n}
$$

represents all possible derivations. Thus, $G$ can derive only strings of the form $a^{n} b^{n}$.
We also have to show that all strings of this form can be derived. This is easy; we simply apply $S \rightarrow a S b$ as many times as needed, followed by $S \rightarrow \lambda$.

## Example 1.11 (continuation)

A grammar $G$ completely defines $L(G)$, but it may not be easy to get a very explicit description of the language from the grammar. Here, however, the answer is fairly clear. It is not hard to conjecture that

$$
L(G)=\left\{a^{n} b^{n}: n \geq 0\right\}
$$

and it is easy to prove it. If we notice that the rule $S \rightarrow a S b$ is recursive, a proof by induction readily suggests itself. We first show that all sentential forms must have the form

$$
\begin{equation*}
w_{i}=a^{i} S b^{i} \tag{2}
\end{equation*}
$$

Suppose that condition (2) holds for all sentential forms $w_{i}$ of length $2 i+1$ or less. To get another sentential form (which is not a sentence), we can only apply the production $S \rightarrow a S b$. This gets us

$$
a^{i} S b^{i} \Rightarrow a^{i+1} S b^{i+1}
$$

so that every sentential form of length $2 i+3$ is also of the form (2). Since (2) is obviously true for $i=1$, it holds by induction for all $i$. Finally, to get a sentence, we must apply the production $S \rightarrow \lambda$, and we see that

$$
S \stackrel{*}{\Rightarrow} a^{n} S b^{n} \Rightarrow a^{n} b^{n}
$$

represents all possible derivations. Thus, $G$ can derive only strings of the form $a^{n} b^{n}$.
We also have to show that all strings of this form can be derived. This is easy; we simply apply $S \rightarrow a S b$ as many times as needed, followed by $S \rightarrow \lambda$.

## Example 1.12

```
The previous examples are fairly easy ones, so rigorous arguments may seem
superfluous. But often it is not so easy to find a grammar for a language
described in an informal way or to give an intuitive characterization of the
language defined by a grammar. To show that a given language is indeed
generated by a certain grammar \(G\), we must be able to show (a) that every
\(w \in L\) can be derived from \(S\) using \(G\) and (b) that every string so derived is in
\(L\).
```


## 2 THREE BASIC CONCEPTS: Grammars

## Example 1.12

Find a grammar that generates

$$
L=\left\{a^{n} b^{n+1}: n \geq 0\right\}
$$

The idea behind the previous example can be extended to this case. All we need to do is generate an extra $b$. This can be done with a production $S \rightarrow A$ with other productions chosen so that $A$ can derive the language in the previous example. Reasoning in this fashion, we get the grammar $G=(\{S, A\},\{a, b\}, S, P)$, with productions

```
S->Al,
A ->aAb,
A->\lambda.
```

Derive a few specific sentences to convince yourself that this works.
The previous examples are fairly easy ones, so rigorous arguments may seem superfluous. But often it is not so easy to find a grammar for a language described in an informal way or to give an intuitive characterization of the language defined by a grammar. To show that a given language is indeed generated by a certain grammar $G$, we must be able to show (a) that every $w \in L$ can be derived from $S$ using $G$ and (b) that every string so derived is in $L$.

## 2 THREE BASIC CONCEPTS: Grammars

Example 1.12
Find a grammar that generates

The idea behind the previous example can be extended to this case. All we need to do is generate an extra $b$. This can be done with a production $S \rightarrow A$ with other productions chosen so that $A$ can derive the language in the previous example. Reasoning in this fashion, we get the grammar $G=(\{S, A\},\{a, b\}, S, P)$, with productions
$\qquad$
$A \rightarrow a A b$,
$A \rightarrow \lambda$.
Derive a few specific sentences to convince yourself that this works.

The previous examples are fairly easy ones, so rigorous arguments may seem superfluous. But often it is not so easy to find a grammar for a language described in an informal way or to give an intuitive characterization of the language defined by a grammar. To show that a given language is indeed generated by a certain grammar $G$, we must be able to show (a) that every $w \in L$ can be derived from $S$ using $G$ and (b) that every string so derived is in $L$.

## 2 THREE BASIC CONCEPTS: Grammars

Example 1.12
Find a grammar that generates

$$
L=\left\{a^{n} b^{n+1}: n \geq 0\right\}
$$

The idea behind the previous example can be extended to this case. All we need to do is generate an extra $b$. This can be done with a production $S \rightarrow A b$, with other productions chosen so that $A$ can derive the language in the previous example. Reasoning in this fashion, we get the grammar $G=(\{S, A\},\{a, b\}, S, P)$, with productions


Derive a few specific sentences to convince yourself that this works.

> The previous examples are fairly easy ones, so rigorous arguments may seem superfluous. But often it is not so easy to find a grammar for a language described in an informal way or to give an intuitive characterization of the language defined by a grammar. To show that a given language is indeed generated by a certain grammar $G$, we must be able to show (a) that every $w \in L$ can be derived from $S$ using $G$ and (b) that every string so derived is in $L$.

## 2 THREE BASIC CONCEPTS: Grammars

Example 1.12
Find a grammar that generates

$$
L=\left\{a^{n} b^{n+1}: n \geq 0\right\}
$$

The idea behind the previous example can be extended to this case. All we need to do is generate an extra $b$. This can be done with a production $S \rightarrow A b$, with other productions chosen so that $A$ can derive the language in the previous example. Reasoning in this fashion, we get the grammar $G=(\{S, A\},\{a, b\}, S, P)$, with productions

Derive a few specific sentences to convince yourself that this works.

> The previous examples are fairly easy ones, so rigorous arguments may seem superfluous. But often it is not so easy to find a grammar for a language described in an informal way or to give an intuitive characterization of the language defined by a grammar. To show that a given language is indeed generated by a certain grammar $G$, we must be able to show (a) that every $w \in L$ can be derived from $S$ using $G$ and (b) that every string so derived is in

Example 1.12
Find a grammar that generates

$$
L=\left\{a^{n} b^{n+1}: n \geq 0\right\}
$$

The idea behind the previous example can be extended to this case. All we need to do is generate an extra $b$.
with other productions chosen so that $A$ can derive the language in the previous example. Reasoning in this fashion, we get the grammar $G=(\{S, A\},\{a, b\}, S, P)$, with productions

Derive a few specific sentences to convince yourself that this works.

> The previous examples are fairly easy ones, so rigorous arguments may seem superfluous. But often it is not so easy to find a grammar for a language described in an informal way or to give an intuitive characterization of the language defined by a grammar. To show that a given language is indeed generated by a certain grammar $G$, we must be able to show (a) that every $w \in L$ can be derived from $S$ using $G$ and (b) that every string so derived is in

Example 1.12
Find a grammar that generates

$$
L=\left\{a^{n} b^{n+1}: n \geq 0\right\}
$$

The idea behind the previous example can be extended to this case. All we need to do is generate an extra $b$. This can be done with a production $S \rightarrow A b$, with other productions chosen so that $A$ can derive the language in the previous example. Reasoning in this fashion, we get the grammar $G=(\{S, A\},\{a, b\}, S, P)$, with productions

Derive a few specific sentences to convince yourself that this works.

> The previous examples are fairly easy ones, so rigorous arguments may seem superfluous. But often it is not so easy to find a grammar for a language described in an informal way or to give an intuitive characterization of the language defined by a grammar. To show that a given language is indeed generated by a certain grammar $G$, we must be able to show (a) that every $w \in L$ can be derived from $S$ using $G$ and (b) that every string so derived is in

Example 1.12
Find a grammar that generates

$$
L=\left\{a^{n} b^{n+1}: n \geq 0\right\}
$$

The idea behind the previous example can be extended to this case. All we need to do is generate an extra $b$. This can be done with a production $S \rightarrow A b$, with other productions chosen so that $A$ can derive the language in the previous example.

with productions

> The previous examples are fairly easy ones, so rigorous arguments may seem
> superfluous. But often it is not so easy to find a grammar for a language
> described in an informal way or to give an intuitive characterization of the
> language defined by a grammar. To show that a given language is indeed generated by a certain grammar $G$, we must be able to show (a) that every $w \in L$ can be derived from $S$ using $G$ and (b) that every string so derived is in

## Example 1.12

Find a grammar that generates

$$
L=\left\{a^{n} b^{n+1}: n \geq 0\right\}
$$

The idea behind the previous example can be extended to this case. All we need to do is generate an extra $b$. This can be done with a production $S \rightarrow A b$, with other productions chosen so that $A$ can derive the language in the previous example. Reasoning in this fashion, we get the grammar $G=(\{S, A\},\{a, b\}, S, P)$, with productions

The previous examples are fairly easy ones, so rigorous arguments may seem superfluous. But often it is not so easy to find a grammar for a language described in an informal way or to give an intuitive characterization of the language defined by a grammar. To show that a given language is indeed generated by a certain grammar $G$, we must be able to show (a) that every $w \in L$ can be derived from $S$ using $G$ and (b) that every string so derived is in

## Example 1.12

Find a grammar that generates

$$
L=\left\{a^{n} b^{n+1}: n \geq 0\right\}
$$

The idea behind the previous example can be extended to this case. All we need to do is generate an extra $b$. This can be done with a production $S \rightarrow A b$, with other productions chosen so that $A$ can derive the language in the previous example. Reasoning in this fashion, we get the grammar $G=(\{S, A\},\{a, b\}, S, P)$, with productions

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

Derive a few specific sentences to convince yourself that this works.

> The previous examples are fairly easy ones, so rigorous arguments may seem superfluous. But often it is not so easy to find a grammar for a language described in an informal way or to give an intuitive characterization of the language defined by a grammar. To show that a given language is indeed generated by a certain grammar $G$, we must be able to show (a) that every $w \in L$ can be derived from $S$ using $G$ and (b) that every string so derived is in

## Example 1.12

Find a grammar that generates

$$
L=\left\{a^{n} b^{n+1}: n \geq 0\right\}
$$

The idea behind the previous example can be extended to this case. All we need to do is generate an extra $b$. This can be done with a production $S \rightarrow A b$, with other productions chosen so that $A$ can derive the language in the previous example. Reasoning in this fashion, we get the grammar $G=(\{S, A\},\{a, b\}, S, P)$, with productions

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

Derive a few specific sentences to convince yourself that this works.

> The previous examples are fairly easy ones, so rigorous arguments may seem superfluous. But often it is not so easy to find a grammar for a language described in an informal way or to give an intuitive characterization of the language defined by a grammar. To show that a given language is indeed generated by a certain grammar $G$, we must be able to show (a) that every $w \in L$ can be derived from $S$ using $G$ and (b) that every string so derived is in

## Example 1.12

Find a grammar that generates

$$
L=\left\{a^{n} b^{n+1}: n \geq 0\right\}
$$

The idea behind the previous example can be extended to this case. All we need to do is generate an extra $b$. This can be done with a production $S \rightarrow A b$, with other productions chosen so that $A$ can derive the language in the previous example. Reasoning in this fashion, we get the grammar $G=(\{S, A\},\{a, b\}, S, P)$, with productions

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

Derive a few specific sentences to convince yourself that this works.

> The previous examples are fairly easy ones, so rigorous arguments may seem superfluous. But often it is not so easy to find a grammar for a language described in an informal way or to give an intuitive characterization of the language defined by a grammar. To show that a given language is indeed generated by a certain grammar $G$, we must be able to show (a) that every $w \in L$ can be derived from $S$ using $G$ and (b) that every string so derived is in

## Example 1.12

Find a grammar that generates

$$
L=\left\{a^{n} b^{n+1}: n \geq 0\right\}
$$

The idea behind the previous example can be extended to this case. All we need to do is generate an extra $b$. This can be done with a production $S \rightarrow A b$, with other productions chosen so that $A$ can derive the language in the previous example. Reasoning in this fashion, we get the grammar $G=(\{S, A\},\{a, b\}, S, P)$, with productions

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

Derive a few specific sentences to convince yourself that this works.
The previous examples are fairly easy ones,
superfluous. But often it is not so easy to find a grammar for a language described in an informal way or to give an intuitive characterization of the language defined by a grammar. To show that a given language is indeed generated by a certain grammar $G$, we must be able to show (a) that every $w \in L$ can be derived from $S$ using $G$ and (b) that every string so derived is in

## Example 1.12

Find a grammar that generates

$$
L=\left\{a^{n} b^{n+1}: n \geq 0\right\}
$$

The idea behind the previous example can be extended to this case. All we need to do is generate an extra $b$. This can be done with a production $S \rightarrow A b$, with other productions chosen so that $A$ can derive the language in the previous example. Reasoning in this fashion, we get the grammar $G=(\{S, A\},\{a, b\}, S, P)$, with productions

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

Derive a few specific sentences to convince yourself that this works.
The previous examples are fairly easy ones, so rigorous arguments may seem superfluous.
described in an informal way or to give an intuitive characterization of the language defined by a grammar. To show that a given language is indeed generated by a certain grammar $G$, we must be able to show (a) that every $w \in L$ can be derived from $S$ using $G$ and (b) that every string so derived is in

## Example 1.12

Find a grammar that generates

$$
L=\left\{a^{n} b^{n+1}: n \geq 0\right\}
$$

The idea behind the previous example can be extended to this case. All we need to do is generate an extra $b$. This can be done with a production $S \rightarrow A b$, with other productions chosen so that $A$ can derive the language in the previous example. Reasoning in this fashion, we get the grammar $G=(\{S, A\},\{a, b\}, S, P)$, with productions

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

Derive a few specific sentences to convince yourself that this works.
The previous examples are fairly easy ones, so rigorous arguments may seem superfluous. But often it is not so easy to find a grammar for a language described in an informal way or to give an intuitive characterization of the language defined by a grammar. To show that a given language is indeed generated by a certain grammar $G$, we must be able to show (a) that every $w \in L$ can be derived from $S$ using $G$ and (b) that every string so derived is in

## 2 THREE BASIC CONCEPTS: Grammars

## Example 1.12

Find a grammar that generates

$$
L=\left\{a^{n} b^{n+1}: n \geq 0\right\} .
$$

The idea behind the previous example can be extended to this case. All we need to do is generate an extra $b$. This can be done with a production $S \rightarrow A b$, with other productions chosen so that $A$ can derive the language in the previous example. Reasoning in this fashion, we get the grammar $G=(\{S, A\},\{a, b\}, S, P)$, with productions

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

Derive a few specific sentences to convince yourself that this works.
The previous examples are fairly easy ones, so rigorous arguments may seem superfluous. But often it is not so easy to find a grammar for a language described in an informal way or to give an intuitive characterization of the language defined by a grammar. To show that a given language is indeed generated by a certain grammar $G$, we must be able to show

## 2 THREE BASIC CONCEPTS: Grammars

## Example 1.12

Find a grammar that generates

$$
L=\left\{a^{n} b^{n+1}: n \geq 0\right\} .
$$

The idea behind the previous example can be extended to this case. All we need to do is generate an extra $b$. This can be done with a production $S \rightarrow A b$, with other productions chosen so that $A$ can derive the language in the previous example. Reasoning in this fashion, we get the grammar $G=(\{S, A\},\{a, b\}, S, P)$, with productions

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

Derive a few specific sentences to convince yourself that this works.
The previous examples are fairly easy ones, so rigorous arguments may seem superfluous. But often it is not so easy to find a grammar for a language described in an informal way or to give an intuitive characterization of the language defined by a grammar. To show that a given language is indeed generated by a certain grammar $G$, we must be able to show (a) that every $w \in L$ can be derived from $S$ using $G$ and (b) that every string so derived is in

## 2 THREE BASIC CONCEPTS: Grammars

## Example 1.12

Find a grammar that generates

$$
L=\left\{a^{n} b^{n+1}: n \geq 0\right\} .
$$

The idea behind the previous example can be extended to this case. All we need to do is generate an extra $b$. This can be done with a production $S \rightarrow A b$, with other productions chosen so that $A$ can derive the language in the previous example. Reasoning in this fashion, we get the grammar $G=(\{S, A\},\{a, b\}, S, P)$, with productions

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

Derive a few specific sentences to convince yourself that this works.
The previous examples are fairly easy ones, so rigorous arguments may seem superfluous. But often it is not so easy to find a grammar for a language described in an informal way or to give an intuitive characterization of the language defined by a grammar. To show that a given language is indeed generated by a certain grammar $G$, we must be able to show (a) that every $w \in L$ can be derived from $S$ using $G$ and (b) that every string so derived is in $L$.

## Example 1.13

## Example 1.13

Take $\Sigma=\{a, b\}$, and let $n_{a}(w)$ and $n_{b}(w)$ denote the number of $a$ 's and $b$ 's in the string $w$, respectively. Then the grammar $G$ with productions

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

generates the language

$$
\mathcal{I}=\left\{w: n_{a}(w)=n_{b}(w)\right\} .
$$

This claim is not so obvious, and we need to provide convincing arguments. First, it is clear that every sentential form of $G$ has an equal number of $a$ 's and $b$ 's, because the only productions that generate the string $a$, namely $S \rightarrow a S b$ and $S \rightarrow b S a$, simultaneously generate the string $b$. Therefore, every element of $L(G)$ is in $L$. It is a little harder to see that every string in $L$ can be derived with $G$.

## Example 1.13

Take $\Sigma=\{a, b\}$, and let $n_{a}(w)$ and $n_{b}(w)$ denote the number of $a$ 's and $b$ 's in the string $w$, respectively. Then the grammar $G$ with productions
generates the language

$$
L=\left\{w: n_{a}(w)=n_{b}(w)\right\} .
$$

This claim is not so obvious, and we need to provide convincing arguments. First, it is clear that every sentential form of $G$ has an equal number of $a$ 's and $b$ 's, because the only productions that generate the string $a$, namely $S \rightarrow a S b$ and $S \rightarrow b S a$, simultaneously generate the string $b$. Therefore, every element of $L(G)$ is in $L$. It is a little harder to see that every string in $L$ can be derived with $G$.

## Example 1.13

Take $\Sigma=\{a, b\}$, and let $n_{a}(w)$ and $n_{b}(w)$ denote the number of $a$ 's and $b$ 's in the string $w$, respectively. Then the grammar $G$ with productions


## generates the language

$$
L=\left\{w: n_{a}(w)=n_{b}(w)\right\}
$$

$\square$
This claim is not so obvious, and we need to provide convincing arguments. First, it is clear that every sentential form of $G$ has an equal number of $a$ 's and $b$ 's, because the only productions that generate the string $a$, namely $S \rightarrow a S b$ and $S \rightarrow b S a$, simultaneously generate the string $b$. Therefore, every element of $L(G)$ is in $L$. It is a little harder to see that every string in $L$ can be derived with $G$.

## Example 1.13

Take $\Sigma=\{a, b\}$, and let $n_{a}(w)$ and $n_{b}(w)$ denote the number of $a$ 's and $b$ 's in the string $w$, respectively. Then the grammar $G$ with productions

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

## generates the language

$$
L=\left\{w: n_{a}(w)=n_{b}(w)\right\}
$$

> This claim is not so obvious, and we need to provide convincing arguments. First, it is clear that every sentential form of $G$ has an equal number of $a$ 's and $b$ 's, because the only productions that generate the string $a$, namely $S \rightarrow a S b$ and $S \rightarrow b S a$, simultaneously generate the string $b$. Therefore, every element of $L(G)$ is in $L$. It is a little harder to see that every string in $L$ can be derived with $G$.

## Example 1.13

Take $\Sigma=\{a, b\}$, and let $n_{a}(w)$ and $n_{b}(w)$ denote the number of $a$ 's and $b$ 's in the string $w$, respectively. Then the grammar $G$ with productions

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

generates the language

> This claim is not so obvious, and we need to provide convincing arguments. First, it is clear that every sentential form of $G$ has an equal number of $a$ 's and $b$ 's, because the only productions that generate the string $a$, namely $S \rightarrow a S b$ and $S \rightarrow b S a$, simultaneously generate the string $b$. Therefore, every element of $L(G)$ is in $L$. It is a little harder to see that every string in $L$ can be derived with $G$.

## Example 1.13

Take $\Sigma=\{a, b\}$, and let $n_{a}(w)$ and $n_{b}(w)$ denote the number of $a$ 's and $b$ 's in the string $w$, respectively. Then the grammar $G$ with productions

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

generates the language

$$
L=\left\{w: n_{a}(w)=n_{b}(w)\right\} .
$$

This claim is not so obvious, and we need to provide convincing arguments. First, it is clear that every sentential form of $G$ has an equal number of $a$ 's and $b$ 's, because the only productions that generate the string $a$, namely $S \rightarrow a S b$ and $S \rightarrow b S a$, simultaneously generate the string $b$. Therefore, every element of $L(G)$ is in $L$. It is a little harder to see that every string in $L$ can be derived with $G$.

## Example 1.13

Take $\Sigma=\{a, b\}$, and let $n_{a}(w)$ and $n_{b}(w)$ denote the number of $a$ 's and $b$ 's in the string $w$, respectively. Then the grammar $G$ with productions

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

generates the language

$$
L=\left\{w: n_{a}(w)=n_{b}(w)\right\}
$$

This claim is not so obvious, and we need to provide convincing arguments. First, it is clear that every sentential form of $G$ has an equal number of $a$ 's and $b$ 's, because the only productions that generate the string $a$, namely $S \rightarrow a S b$ and $S \rightarrow b S a$, simultaneously generate the string $b$. Therefore, every element of $L(G)$ is in $L$. It is a little harder to see that every string in $L$ can be derived with $G$.

## Example 1.13

Take $\Sigma=\{a, b\}$, and let $n_{a}(w)$ and $n_{b}(w)$ denote the number of $a$ 's and $b$ 's in the string $w$, respectively. Then the grammar $G$ with productions

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

generates the language

$$
L=\left\{w: n_{a}(w)=n_{b}(w)\right\}
$$

This claim is not so obvious, and we need to provide convincing arguments.
First, it is clear that every sentential form of $G$ has an equal number of $a$ 's and $b$ 's, because the only productions that generate the string $a$, namely $S \rightarrow a S b$ and $S \rightarrow b S a$, simultaneously generate the string $b$. Therefore, every element of $L(G)$ is in $L$. It is a little harder to see that every string in $L$ can be derived with $G$.

## Example 1.13

Take $\Sigma=\{a, b\}$, and let $n_{a}(w)$ and $n_{b}(w)$ denote the number of $a$ 's and $b$ 's in the string $w$, respectively. Then the grammar $G$ with productions

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

generates the language

$$
L=\left\{w: n_{a}(w)=n_{b}(w)\right\}
$$

This claim is not so obvious, and we need to provide convincing arguments.
First, it is clear that every sentential form of $G$ has an equal number of $a$ 's and $b$ 's, and $S \rightarrow b S a$, simultaneously generate the string $b$. Therefore, every element of $L(G)$ is in $L$. It is a little harder to see that every string in $L$ can be derived

## Example 1.13

Take $\Sigma=\{a, b\}$, and let $n_{a}(w)$ and $n_{b}(w)$ denote the number of $a$ 's and $b$ 's in the string $w$, respectively. Then the grammar $G$ with productions

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

generates the language

$$
L=\left\{w: n_{a}(w)=n_{b}(w)\right\}
$$

This claim is not so obvious, and we need to provide convincing arguments.
First, it is clear that every sentential form of $G$ has an equal number of $a$ 's and $b$ 's, because the only productions that generate the string $a$,


## Example 1.13

Take $\Sigma=\{a, b\}$, and let $n_{a}(w)$ and $n_{b}(w)$ denote the number of $a$ 's and $b$ 's in the string $w$, respectively. Then the grammar $G$ with productions

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

generates the language

$$
L=\left\{w: n_{a}(w)=n_{b}(w)\right\}
$$

This claim is not so obvious, and we need to provide convincing arguments.
First, it is clear that every sentential form of $G$ has an equal number of $a$ 's and $b$ 's, because the only productions that generate the string $a$, namely $S \rightarrow a S b$ and $S \rightarrow b S a$, simultaneously generate the string $b$. Therefore, every element of $L(G)$ is in $L$. It is a little harder to see that every string in $L$ can be derived

## Example 1.13

Take $\Sigma=\{a, b\}$, and let $n_{a}(w)$ and $n_{b}(w)$ denote the number of $a$ 's and $b$ 's in the string $w$, respectively. Then the grammar $G$ with productions

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

generates the language

$$
L=\left\{w: n_{a}(w)=n_{b}(w)\right\}
$$

This claim is not so obvious, and we need to provide convincing arguments. First, it is clear that every sentential form of $G$ has an equal number of $a$ 's and $b$ 's, because the only productions that generate the string $a$, namely $S \rightarrow a S b$ and $S \rightarrow b S a$, simultaneously generate the string $b$.

## Example 1.13

Take $\Sigma=\{a, b\}$, and let $n_{a}(w)$ and $n_{b}(w)$ denote the number of $a$ 's and $b$ 's in the string $w$, respectively. Then the grammar $G$ with productions

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

generates the language

$$
L=\left\{w: n_{a}(w)=n_{b}(w)\right\}
$$

This claim is not so obvious, and we need to provide convincing arguments. First, it is clear that every sentential form of $G$ has an equal number of $a$ 's and $b$ 's, because the only productions that generate the string $a$, namely $S \rightarrow a S b$ and $S \rightarrow b S a$, simultaneously generate the string $b$. Therefore, every element of $L(G)$ is in $L$.

## Example 1.13

Take $\Sigma=\{a, b\}$, and let $n_{a}(w)$ and $n_{b}(w)$ denote the number of $a$ 's and $b$ 's in the string $w$, respectively. Then the grammar $G$ with productions

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

generates the language

$$
L=\left\{w: n_{a}(w)=n_{b}(w)\right\} .
$$

This claim is not so obvious, and we need to provide convincing arguments. First, it is clear that every sentential form of $G$ has an equal number of $a$ 's and $b$ 's, because the only productions that generate the string $a$, namely $S \rightarrow a S b$ and $S \rightarrow b S a$, simultaneously generate the string $b$. Therefore, every element of $L(G)$ is in $L$. It is a little harder to see that every string in $L$ can be derived with $G$.

## 2 THREE BASIC CONCEPTS: Grammars

## Example 1.13 (continuation)

Let us begin by looking at the problem in outline, considering the various forms $w \in L$ can have. Suppose $w$ starts with a and ends with $b$. Then it has the form

$$
w=a w_{1} b,
$$

where $w_{1}$ is also in $L$. We can think of this case as being derived starting with $S \Rightarrow a S b$
if $S$ does indeed derive any string in $L$. A similar argument can be made if $w$ starts with $b$ and ends with $a$. But this does not take care of all cases, because a string in $L$ can begin and end with the same symbol. If we write down a string of this type, say $a a b b b a$, we see that it can be considered as the concatenation of two shorter strings $a a b b$ and $b a$, both of which are in L. Is this true in general? To show that this is indeed so, we can use the following argument: Suppose that, starting at the left end of the string, we count +1 for the string $a$ and -1 for the string $b$. If a string $w$ starts and ends with $a$, then the count will be +1 after the leftmost symbol and -1 immediately before the rightmost one. Therefore, the count has to go through zero somewhere in the middle of the string, indicating that such a string must have the form

$$
w=w_{1} w_{2},
$$

where both $w_{1}$ and $w_{2}$ are in $L$. This case can be taken care of by the production $S \rightarrow S S$.

## 2 THREE BASIC CONCEPTS: Grammars

## Example 1.13 (continuation)

Let us begin by looking at the problem in outline, considering the various forms $w \in L$ can have. Suppose $w$ starts with a and ends with $b$. Then it has the form $w=a w_{1} b$,
where $w_{1}$ is also in $L$. We can think of this case as being derived starting with $S \Rightarrow a S b$
if $S$ does indeed derive any string in $L$. A similar argument can be made if $w$ starts with $b$ and ends with $a$. But this does not take care of all cases, because a string in $L$ can begin and end with the same symbol. If we write down a string of this type, say aabbba, we see that it can be considered as the concatenation of two shorter strings $a a b b$ and $b a$, both of which are in $L$. Is this true in general? To show that this is indeed so, we can use the following argument: Suppose that, starting at the left end of the string, we count +1 for the string $a$ and -1 for the string $b$. If a string $w$ starts and ends with $a$, then the count will be +1 after the leftmost symbol and -1 immediately before the rightmost one. Therefore, the count has to go through zero somewhere in the middle of the string, indicating that such a string must have the form

$$
w=w_{1} w_{2}
$$

where both $w_{1}$ and $w_{2}$ are in $L$. This case can be taken care of by the production

## 2 THREE BASIC CONCEPTS: Grammars

Example 1.13 (continuation)
Let us begin by looking at the problem in outline, considering the various forms $w \in L$ can have.
$w=a w_{1} b$,
where $w_{1}$ is also in $L$. We can think of this case as being derived starting with $S \Rightarrow a S b$
if $S$ does indeed derive any string in $L$. A similar argument can be made if $w$ starts with $b$ and ends with $a$. But this does not take care of all cases, because a string in $L$ can begin and end with the same symbol. If we write down a string of this type, say $a a b b b a$, we see that it can be considered as the concatenation of two shorter strings $a a b b$ and $b a$, both of which are in $L$. Is this true in general? To show that this is indeed so, we can use the following argument: Suppose that, starting at the left end of the string, we count +1 for the string $a$ and -1 for the string $b$. If a string $w$ starts and ends with $a$, then the count will be +1 after the leftmost symbol and -1 immediately before the rightmost one. Therefore, the count has to go through zero somewhere in the middle of the string, indicating that such a string must have the form

$$
w=w_{1} w_{2},
$$

where both $w_{1}$ and $w_{2}$ are in $L$. This case can be taken care of by the production

Example 1.13 (continuation)
Let us begin by looking at the problem in outline, considering the various forms $w \in L$ can have. Suppose $w$ starts with a and ends with $b$.
where $w_{1}$ is also in $L$. We can think of this case as being derived starting with $S \Rightarrow a S b$
if $S$ does indeed derive any string in $L$. A similar argument can be made if $w$ starts with $b$ and ends with $a$. But this does not take care of all cases, because a string in $L$ can begin and end with the same symbol. If we write down a string of this type, say $a a b b b a$, we see that it can be considered as the concatenation of two shorter strings $a a b b$ and $b a$, both of which are in L. Is this true in general? To show that this is indeed so, we can use the following argument: Suppose that, starting at the left end of the string, we count +1 for the string $a$ and -1 for the string $b$. If a string $w$ starts and ends with $a$, then the count will be +1 after the leftmost symbol and -1 immediately before the rightmost one. Therefore, the count has to go through zero somewhere in the middle of the string, indicating that such a string must have the form $w=w_{1} w_{2}$,
where both $w_{1}$ and $w_{2}$ are in $L$. This case can be taken care of by the production

Example 1.13 (continuation)
Let us begin by looking at the problem in outline, considering the various forms $w \in L$ can have. Suppose $w$ starts with a and ends with $b$. Then it has the form
where $w_{1}$ is also in $L$. We can think of this case as being derived starting with $S \Rightarrow a S b$
if $S$ does indeed derive any string in $L$. A similar argument can be made if $w$ starts with $b$ and ends with $a$. But this does not take care of all cases, because a string in $L$ can begin and end with the same symbol. If we write down a string of this type, say $a a b b b a$, we see that it can be considered as the concatenation of two shorter strings $a a b b$ and $b a$, both of which are in L. Is this true in general? To show that this is indeed so, we can use the following argument: Suppose that, starting at the left end of the string, we count +1 for the string $a$ and -1 for the string $b$. If a string $w$ starts and ends with $a$, then the count will be +1 after the leftmost symbol and -1 immediately before the rightmost one. Therefore, the count has to go through zero somewhere in the middle of the string, indicating that such a string must have the form $w=w_{1} w_{2}$,
where both $w_{1}$ and $w_{2}$ are in $L$. This case can be taken care of by the production

Example 1.13 (continuation)
Let us begin by looking at the problem in outline, considering the various forms $w \in L$ can have. Suppose $w$ starts with a and ends with $b$. Then it has the form

$$
w=a w_{1} b
$$

where $w_{1}$ is also in $L$. We can think of this case as being derived starting with $S \Rightarrow a S b$
if $S$ does indeed derive any string in $L$. A similar argument can be made if $w$ starts with $b$ and ends with $a$. But this does not take care of all cases, because a string in $L$ can begin and end with the same symbol. If we write down a string of this type, say $a a b b b a$, we see that it can be considered as the concatenation of two shorter strings $a a b b$ and $b a$, both of which are in $L$. Is this true in general? To show that this is indeed so, we can use the following argument: Suppose that, starting at the left end of the string, we count +1 for the string $a$ and -1 for the string $b$. If a string $w$ starts and ends with $a$, then the count will be +1 after the leftmost symbol and -1 immediately before the rightmost one. Therefore, the count has to go through zero somewhere in the middle of the string, indicating that such a string must have the form $w=w_{1} w_{2}$,
where both $w_{1}$ and $w_{2}$ are in $L$. This case can be taken care of by the production

Example 1.13 (continuation)
Let us begin by looking at the problem in outline, considering the various forms $w \in L$ can have. Suppose $w$ starts with a and ends with $b$. Then it has the form

$$
w=a w_{1} b
$$

where $w_{1}$ is also in $L$. We can think of this case as being derived starting with
if $S$ does indeed derive any string in $L$. A similar argument can be made if $w$ starts with $b$ and ends with $a$. But this does not take care of all cases, because a string in $L$ can begin and end with the same symbol. If we write down a string of this type, say $a a b b b a$, we see that it can be considered as the concatenation of two shorter strings $a a b b$ and $b a$, both of which are in $L$. Is this true in general? To show that this is indeed so, we can use the following argument: Suppose that, starting at the left end of the string, we count +1 for the string $a$ and -1 for the string $b$. If a string $w$ starts and ends with $a$, then the count will be +1 after the leftmost symbol and -1 immediately before the rightmost one. Therefore, the count has to go through zero somewhere in the middle of the string, indicating that such a string must have the form $w=w_{1} w_{2}$,

[^3] production

Example 1.13 (continuation)
Let us begin by looking at the problem in outline, considering the various forms $w \in L$ can have. Suppose $w$ starts with a and ends with $b$. Then it has the form

$$
w=a w_{1} b
$$

where $w_{1}$ is also in $L$. We can think of this case as being derived starting with
if $S$ does indeed derive any string in $L$. A similar argument can be made if $w$ starts with $b$ and ends with $a$. But this does not take care of all cases, because a string in $L$ can begin and end with the same symbol. If we write down a string of this type, say $a a b b b a$, we see that it can be considered as the concatenation of two shorter strings $a a b b$ and $b a$, both of which are in $L$. Is this true in general? To show that this is indeed so, we can use the following argument: Suppose that, starting at the left end of the string, we count +1 for the string $a$ and -1 for the string $b$. If a string $w$ starts and ends with $a$, then the count will be +1 after the leftmost symbol and -1 immediately before the rightmost one. Therefore, the count has to go through zero somewhere in the middle of the string, indicating that such a string must have the form

[^4] production

Example 1.13 (continuation)
Let us begin by looking at the problem in outline, considering the various forms $w \in L$ can have. Suppose $w$ starts with a and ends with $b$. Then it has the form

$$
w=a w_{1} b
$$

where $w_{1}$ is also in $L$. We can think of this case as being derived starting with

$$
S \Rightarrow a S b
$$

> if $S$ does indeed derive any string in $L$. A similar argument can be made if $u$ starts with $b$ and ends with $a$. But this does not take care of all cases, because a string in $L$ can begin and end with the same symbol. If we write down a string of this type, say $a a b b b a$, we see that it can be considered as the concatenation of two shorter strings $a a b b$ and $b a$, both of which are in $L$. Is this true in general? To show that this is indeed so, we can use the following argument: Suppose that, starting at the left end of the string, we count +1 for the string $a$ and -1 for the string $b$. If a string $w$ starts and ends with $a$, then the count will be +1 after the leftmost symbol and -1 immediately before the rightmost one. Therefore, the count has to go through zero somewhere in the middle of the string, indicating that such a string must have the form

[^5] production

Example 1.13 (continuation)
Let us begin by looking at the problem in outline, considering the various forms $w \in L$ can have. Suppose $w$ starts with a and ends with $b$. Then it has the form

$$
w=a w_{1} b
$$

where $w_{1}$ is also in $L$. We can think of this case as being derived starting with

$$
S \Rightarrow a S b
$$

if $S$ does indeed derive any string in $L$.


#### Abstract

starts with $b$ and ends with $a$. But this does not take care of all cases, because a string in $L$ can begin and end with the same symbol. If we write down a string of this type, say $a a b b b a$, we see that it can be considered as the concatenation of two shorter strings $a a b b$ and $b a$, both of which are in $L$. Is this true in general? To show that this is indeed so, we can use the following argument: Suppose that, starting at the left end of the string, we count +1 for the string $a$ and -1 for the string $b$. If a string $w$ starts and ends with $a$, then the count will be +1 after the leftmost symbol and -1 immediately before the rightmost one. Therefore, the count has to go through zero somewhere in the middle of the string, indicating that such a string must have the form


[^6] production

## Example 1.13 (continuation)

Let us begin by looking at the problem in outline, considering the various forms $w \in L$ can have. Suppose $w$ starts with a and ends with $b$. Then it has the form

$$
w=a w_{1} b
$$

where $w_{1}$ is also in $L$. We can think of this case as being derived starting with

$$
S \Rightarrow a S b
$$

if $S$ does indeed derive any string in $L$. A similar argument can be made if $w$ starts with $b$ and ends with $a$.
a string in $L$ can begin and end with the same symbol. If we write down a string of this type, say $a a b b b a$, we see that it can be considered as the concatenation of two shorter strings $a a b b$ and $b a$, both of which are in $L$. Is this true in general? To show that this is indeed so, we can use the following argument: Suppose that, starting at the left end of the string, we count +1 for the string $a$ and -1 for the string $b$. If a string $w$ starts and ends with $a$, then the count will be +1 after the leftmost symbol and -1 immediately before the rightmost one. Therefore, the count has to go through zero somewhere in the middle of the string, indicating that such a string must have the form

[^7] production

## Example 1.13 (continuation)

Let us begin by looking at the problem in outline, considering the various forms $w \in L$ can have. Suppose $w$ starts with a and ends with $b$. Then it has the form

$$
w=a w_{1} b
$$

where $w_{1}$ is also in $L$. We can think of this case as being derived starting with

$$
S \Rightarrow a S b
$$

if $S$ does indeed derive any string in $L$. A similar argument can be made if $w$ starts with $b$ and ends with $a$. But this does not take care of all cases,
a string in $L$ can begin and end with the same symbol. If we write down a string of this type, say $a a b b b a$, we see that it can be considered as the concatenation of two shorter strings $a a b b$ and $b a$, both of which are in $L$. Is this true in general? To show that this is indeed so, we can use the following argument: Suppose that, starting at the left end of the string, we count +1 for the string $a$ and -1 for the string $b$. If a string $w$ starts and ends with $a$, then the count will be +1 after the leftmost symbol and -1 immediately before the rightmost one. Therefore, the count has to go through zero somewhere in the middle of the string, indicating that such a string must have the form where both $w_{1}$ and $w_{2}$ are in $L$. This case can be taken care of by the production

## Example 1.13 (continuation)

Let us begin by looking at the problem in outline, considering the various forms $w \in L$ can have. Suppose $w$ starts with a and ends with $b$. Then it has the form

$$
w=a w_{1} b
$$

where $w_{1}$ is also in $L$. We can think of this case as being derived starting with

$$
S \Rightarrow a S b
$$

if $S$ does indeed derive any string in $L$. A similar argument can be made if $w$ starts with $b$ and ends with $a$. But this does not take care of all cases, because a string in $L$ can begin and end with the same symbol.
> string of this type, say aabbba, we see that it can be considered as the concatenation of two shorter strings $a a b b$ and $b a$, both of which are in $L$. Is this true in general? To show that this is indeed so, we can use the following argument: Suppose that, starting at the left end of the string, we count +1 for the string $a$ and -1 for the string $b$. If a string $w$ starts and ends with $a$, then the count will be +1 after the leftmost symbol and -1 immediately before the rightmost one. Therefore, the count has to go through zero somewhere in the middle of the string, indicating that such a string must have the form

[^8] production

## Example 1.13 (continuation)

Let us begin by looking at the problem in outline, considering the various forms $w \in L$ can have. Suppose $w$ starts with a and ends with $b$. Then it has the form

$$
w=a w_{1} b
$$

where $w_{1}$ is also in $L$. We can think of this case as being derived starting with

$$
S \Rightarrow a S b
$$

if $S$ does indeed derive any string in $L$. A similar argument can be made if $w$ starts with $b$ and ends with $a$. But this does not take care of all cases, because a string in $L$ can begin and end with the same symbol. If we write down a string of this type, say $a a b b b a$,
we see that it can be considered as the concatenation of two shorter strings $a a b b$ and $b a$, both of which are in $L$. Is
this true in general? To show that this is indeed so, we can use the following
argument: Suppose that, starting at the left end of the string, we count +1 for
the string $a$ and -1 for the string $b$. If a string $w$ starts and ends with $a$, then
the count will be +1 after the leftmost symbol and -1 immediately before the
rightmost one. Therefore, the count has to go through zero somewhere in the
middle of the string, indicating that such a string must have the form where both $w_{1}$ and $w_{2}$ are in $L$. This case can be taken care of by the production

## Example 1.13 (continuation)

Let us begin by looking at the problem in outline, considering the various forms $w \in L$ can have. Suppose $w$ starts with a and ends with $b$. Then it has the form

$$
w=a w_{1} b
$$

where $w_{1}$ is also in $L$. We can think of this case as being derived starting with

$$
S \Rightarrow a S b
$$

if $S$ does indeed derive any string in $L$. A similar argument can be made if $w$ starts with $b$ and ends with $a$. But this does not take care of all cases, because a string in $L$ can begin and end with the same symbol. If we write down a string of this type, say $a a b b b a$, we see that it can be considered as the concatenation of two shorter strings $a a b b$ and $b a$,
this true in general? To show that this is indeed so, we can use the following argument: Suppose that, starting at the left end of the string, we count +1 for the string $a$ and -1 for the string $b$. If a string $w$ starts and ends with $a$, then the count will be +1 after the leftmost symbol and -1 immediately before the rightmost one. Therefore, the count has to go through zero somewhere in the middle of the string, indicating that such a string must have the form

[^9] production

## Example 1.13 (continuation)

Let us begin by looking at the problem in outline, considering the various forms $w \in L$ can have. Suppose $w$ starts with a and ends with $b$. Then it has the form

$$
w=a w_{1} b
$$

where $w_{1}$ is also in $L$. We can think of this case as being derived starting with

$$
S \Rightarrow a S b
$$

if $S$ does indeed derive any string in $L$. A similar argument can be made if $w$ starts with $b$ and ends with $a$. But this does not take care of all cases, because a string in $L$ can begin and end with the same symbol. If we write down a string of this type, say $a a b b b a$, we see that it can be considered as the concatenation of two shorter strings $a a b b$ and $b a$, both of which are in $L$.
argument: Suppose that, starting at the left end of the string, we count +1 for the string $a$ and -1 for the string $b$. If a string $w$ starts and ends with $a$, then the count will be +1 after the leftmost symbol and -1 immediately before the rightmost one. Therefore, the count has to go through zero somewhere in the middle of the string, indicating that such a string must have the form

[^10] production

## Example 1.13 (continuation)

Let us begin by looking at the problem in outline, considering the various forms $w \in L$ can have. Suppose $w$ starts with a and ends with $b$. Then it has the form

$$
w=a w_{1} b
$$

where $w_{1}$ is also in $L$. We can think of this case as being derived starting with

$$
S \Rightarrow a S b
$$

if $S$ does indeed derive any string in $L$. A similar argument can be made if $w$ starts with $b$ and ends with $a$. But this does not take care of all cases, because a string in $L$ can begin and end with the same symbol. If we write down a string of this type, say $a a b b b a$, we see that it can be considered as the concatenation of two shorter strings $a a b b$ and $b a$, both of which are in $L$. Is this true in general?
argument: Suppose that, starting at the left end of the string, we count +1 for the string $a$ and -1 for the string $b$. If a string $w$ starts and ends with $a$, then the count will be +1 after the leftmost symbol and -1 immediately before the rightmost one. Therefore, the count has to go through zero somewhere in the middle of the string, indicating that such a string must have the form

[^11] production

## Example 1.13 (continuation)

Let us begin by looking at the problem in outline, considering the various forms $w \in L$ can have. Suppose $w$ starts with a and ends with $b$. Then it has the form

$$
w=a w_{1} b
$$

where $w_{1}$ is also in $L$. We can think of this case as being derived starting with

$$
S \Rightarrow a S b
$$

if $S$ does indeed derive any string in $L$. A similar argument can be made if $w$ starts with $b$ and ends with $a$. But this does not take care of all cases, because a string in $L$ can begin and end with the same symbol. If we write down a string of this type, say $a a b b b a$, we see that it can be considered as the concatenation of two shorter strings $a a b b$ and $b a$, both of which are in $L$. Is this true in general? To show that this is indeed so, we can use the following argument:
the string $a$ and -1 for the string $b$. If a string $w$ starts and ends with $a$, then the count will be +1 after the leftmost symbol and -1 immediately before the rightmost one. Therefore, the count has to go through zero somewhere in the middle of the string, indicating that such a string must have the form

[^12] production

## Example 1.13 (continuation)

Let us begin by looking at the problem in outline, considering the various forms $w \in L$ can have. Suppose $w$ starts with a and ends with $b$. Then it has the form

$$
w=a w_{1} b
$$

where $w_{1}$ is also in $L$. We can think of this case as being derived starting with

$$
S \Rightarrow a S b
$$

if $S$ does indeed derive any string in $L$. A similar argument can be made if $w$ starts with $b$ and ends with $a$. But this does not take care of all cases, because a string in $L$ can begin and end with the same symbol. If we write down a string of this type, say $a a b b b a$, we see that it can be considered as the concatenation of two shorter strings $a a b b$ and $b a$, both of which are in $L$. Is this true in general? To show that this is indeed so, we can use the following argument: Suppose that, starting at the left end of the string,
> the count will be +1 after the leftmost symbol and -1 immediately before the rightmost one. Therefore, the count has to go through zero somewhere in the middle of the string, indicating that such a string must have the form

[^13] production

## Example 1.13 (continuation)

Let us begin by looking at the problem in outline, considering the various forms $w \in L$ can have. Suppose $w$ starts with a and ends with $b$. Then it has the form

$$
w=a w_{1} b
$$

where $w_{1}$ is also in $L$. We can think of this case as being derived starting with

$$
S \Rightarrow a S b
$$

if $S$ does indeed derive any string in $L$. A similar argument can be made if $w$ starts with $b$ and ends with $a$. But this does not take care of all cases, because a string in $L$ can begin and end with the same symbol. If we write down a string of this type, say $a a b b b a$, we see that it can be considered as the concatenation of two shorter strings $a a b b$ and $b a$, both of which are in $L$. Is this true in general? To show that this is indeed so, we can use the following argument: Suppose that, starting at the left end of the string, we count +1 for the string $a$ and -1 for the string $b$.
the count will be +1 after the leftmost symbol and -1 immediately before the rightmost one. Therefore, the count has to go through zero somewhere in the middle of the string, indicating that such a string must have the form

[^14] production

## Example 1.13 (continuation)

Let us begin by looking at the problem in outline, considering the various forms $w \in L$ can have. Suppose $w$ starts with a and ends with $b$. Then it has the form

$$
w=a w_{1} b
$$

where $w_{1}$ is also in $L$. We can think of this case as being derived starting with

$$
S \Rightarrow a S b
$$

if $S$ does indeed derive any string in $L$. A similar argument can be made if $w$ starts with $b$ and ends with $a$. But this does not take care of all cases, because a string in $L$ can begin and end with the same symbol. If we write down a string of this type, say $a a b b b a$, we see that it can be considered as the concatenation of two shorter strings $a a b b$ and $b a$, both of which are in $L$. Is this true in general? To show that this is indeed so, we can use the following argument: Suppose that, starting at the left end of the string, we count +1 for the string $a$ and -1 for the string $b$. If a string $w$ starts and ends with $a$, then the count will be +1 after the leftmost symbol
rightmost one. Therefore, the count has to go through zero somewhere in the middle of the string, indicating that such a string must have the form

[^15] production

## Example 1.13 (continuation)

Let us begin by looking at the problem in outline, considering the various forms $w \in L$ can have. Suppose $w$ starts with a and ends with $b$. Then it has the form

$$
w=a w_{1} b
$$

where $w_{1}$ is also in $L$. We can think of this case as being derived starting with

$$
S \Rightarrow a S b
$$

if $S$ does indeed derive any string in $L$. A similar argument can be made if $w$ starts with $b$ and ends with $a$. But this does not take care of all cases, because a string in $L$ can begin and end with the same symbol. If we write down a string of this type, say $a a b b b a$, we see that it can be considered as the concatenation of two shorter strings $a a b b$ and $b a$, both of which are in $L$. Is this true in general? To show that this is indeed so, we can use the following argument: Suppose that, starting at the left end of the string, we count +1 for the string $a$ and -1 for the string $b$. If a string $w$ starts and ends with $a$, then the count will be +1 after the leftmost symbol and -1 immediately before the rightmost one.
middle of the string, indicating that such a string must have the form
where both $w_{1}$ and $w_{2}$ are in $L$. This case can be taken care of by the production

## Example 1.13 (continuation)

Let us begin by looking at the problem in outline, considering the various forms $w \in L$ can have. Suppose $w$ starts with a and ends with $b$. Then it has the form

$$
w=a w_{1} b
$$

where $w_{1}$ is also in $L$. We can think of this case as being derived starting with

$$
S \Rightarrow a S b
$$

if $S$ does indeed derive any string in $L$. A similar argument can be made if $w$ starts with $b$ and ends with $a$. But this does not take care of all cases, because a string in $L$ can begin and end with the same symbol. If we write down a string of this type, say $a a b b b a$, we see that it can be considered as the concatenation of two shorter strings $a a b b$ and $b a$, both of which are in $L$. Is this true in general? To show that this is indeed so, we can use the following argument: Suppose that, starting at the left end of the string, we count +1 for the string $a$ and -1 for the string $b$. If a string $w$ starts and ends with $a$, then the count will be +1 after the leftmost symbol and -1 immediately before the rightmost one. Therefore, the count has to go through zero somewhere in the middle of the string,

[^16]
## production

## Example 1.13 (continuation)

Let us begin by looking at the problem in outline, considering the various forms $w \in L$ can have. Suppose $w$ starts with a and ends with $b$. Then it has the form

$$
w=a w_{1} b
$$

where $w_{1}$ is also in $L$. We can think of this case as being derived starting with

$$
S \Rightarrow a S b
$$

if $S$ does indeed derive any string in $L$. A similar argument can be made if $w$ starts with $b$ and ends with $a$. But this does not take care of all cases, because a string in $L$ can begin and end with the same symbol. If we write down a string of this type, say $a a b b b a$, we see that it can be considered as the concatenation of two shorter strings $a a b b$ and $b a$, both of which are in $L$. Is this true in general? To show that this is indeed so, we can use the following argument: Suppose that, starting at the left end of the string, we count +1 for the string $a$ and -1 for the string $b$. If a string $w$ starts and ends with $a$, then the count will be +1 after the leftmost symbol and -1 immediately before the rightmost one. Therefore, the count has to go through zero somewhere in the middle of the string, indicating that such a string must have the form

[^17] production

## Example 1.13 (continuation)

Let us begin by looking at the problem in outline, considering the various forms $w \in L$ can have. Suppose $w$ starts with a and ends with $b$. Then it has the form

$$
w=a w_{1} b
$$

where $w_{1}$ is also in $L$. We can think of this case as being derived starting with

$$
S \Rightarrow a S b
$$

if $S$ does indeed derive any string in $L$. A similar argument can be made if $w$ starts with $b$ and ends with $a$. But this does not take care of all cases, because a string in $L$ can begin and end with the same symbol. If we write down a string of this type, say $a a b b b a$, we see that it can be considered as the concatenation of two shorter strings $a a b b$ and $b a$, both of which are in $L$. Is this true in general? To show that this is indeed so, we can use the following argument: Suppose that, starting at the left end of the string, we count +1 for the string $a$ and -1 for the string $b$. If a string $w$ starts and ends with $a$, then the count will be +1 after the leftmost symbol and -1 immediately before the rightmost one. Therefore, the count has to go through zero somewhere in the middle of the string, indicating that such a string must have the form

$$
w=w_{1} w_{2}
$$

where both $w_{1}$ and $w_{2}$ are in $L$. This case can be taken care of by the production

## Example 1.13 (continuation)

Let us begin by looking at the problem in outline, considering the various forms $w \in L$ can have. Suppose $w$ starts with a and ends with $b$. Then it has the form

$$
w=a w_{1} b
$$

where $w_{1}$ is also in $L$. We can think of this case as being derived starting with

$$
S \Rightarrow a S b
$$

if $S$ does indeed derive any string in $L$. A similar argument can be made if $w$ starts with $b$ and ends with $a$. But this does not take care of all cases, because a string in $L$ can begin and end with the same symbol. If we write down a string of this type, say $a a b b b a$, we see that it can be considered as the concatenation of two shorter strings $a a b b$ and $b a$, both of which are in $L$. Is this true in general? To show that this is indeed so, we can use the following argument: Suppose that, starting at the left end of the string, we count +1 for the string $a$ and -1 for the string $b$. If a string $w$ starts and ends with $a$, then the count will be +1 after the leftmost symbol and -1 immediately before the rightmost one. Therefore, the count has to go through zero somewhere in the middle of the string, indicating that such a string must have the form

$$
w=w_{1} w_{2}
$$

where both $w_{1}$ and $w_{2}$ are in $L$. This case can be taken care of by the
production

## Example 1.13 (continuation)

Let us begin by looking at the problem in outline, considering the various forms $w \in L$ can have. Suppose $w$ starts with a and ends with $b$. Then it has the form

$$
w=a w_{1} b
$$

where $w_{1}$ is also in $L$. We can think of this case as being derived starting with

$$
S \Rightarrow a S b
$$

if $S$ does indeed derive any string in $L$. A similar argument can be made if $w$ starts with $b$ and ends with $a$. But this does not take care of all cases, because a string in $L$ can begin and end with the same symbol. If we write down a string of this type, say $a a b b b a$, we see that it can be considered as the concatenation of two shorter strings $a a b b$ and $b a$, both of which are in $L$. Is this true in general? To show that this is indeed so, we can use the following argument: Suppose that, starting at the left end of the string, we count +1 for the string $a$ and -1 for the string $b$. If a string $w$ starts and ends with $a$, then the count will be +1 after the leftmost symbol and -1 immediately before the rightmost one. Therefore, the count has to go through zero somewhere in the middle of the string, indicating that such a string must have the form

$$
w=w_{1} w_{2}
$$

where both $w_{1}$ and $w_{2}$ are in $L$. This case can be taken care of by the production $S \rightarrow S S$.

## Example 1.13 (continuation)

## 2 THREE BASIC CONCEPTS: Grammars

## Example 1.13 (continuation)

Once we see the argument intuitively, we are ready to proceed more rigorously. Again we use induction. Assume that all $w \in L$ with $|w| \leq 2 n$ can be derived with $G$. Take any $w \in L$ of length $2 n+2$. If $w=a w_{1} b$, then $w_{1}$ is in $L$, and $\left|w_{1}\right|=2 n$. Therefore, by assumption we have that

$$
S \stackrel{*}{\Rightarrow} w_{1} .
$$

Then

$$
S \Rightarrow a S b \stackrel{*}{\Rightarrow} a w_{1} b=w
$$

is possible, and $w$ can be derived with $G$. Obviously, similar arguments can be made if $w=b w_{1} a$.
If $w$ is not of this form, that is, if it starts and ends with the same symbol, then the counting argument tells us that it must have the form $w=w_{1} w_{2}$, with $w_{1}$ and $w_{2}$ both in $L$ and of length less than or equal to $2 n$. Hence again we see that

$$
S \Rightarrow S S \Rightarrow w_{1} S \Rightarrow w_{1} w_{2}=w
$$

is possible.
Since the inductive assumption is clearly satisfied for $n=1$, we have a basis, and the claim is true for all $n$, completing our argument.

## Example 1.13 (continuation)

Once we see the argument intuitively, we are ready to proceed more rigorously. Again we use induction. Assume that all $w \in L$ with $|w| \leq 2 n$ can be derived with $G$. Take any $w \in L$ of length $2 n+2$. If $w=a w_{1} b$, then $w_{1}$ is in $L$, and $\left|w_{1}\right|=2 n$. Therefore, by assumption we have that
$\mathrm{S} \stackrel{*}{-}$

## Then

$$
S \Rightarrow a S b \stackrel{*}{\Rightarrow} a w_{1} b=w
$$

is possible, and $w$ can be derived with $G$. Obviously, similar arguments can be made if $w=b w_{1} a$.
If $w$ is not of this form, that is, if it starts and ends with the same symbol, then the counting argument tells us that it must have the form $w=w_{1} w_{2}$, with $w_{1}$ and $w_{2}$ both in $L$ and of length less than or equal to $2 n$. Hence again we see that

$$
S \Rightarrow S S \stackrel{*}{\Rightarrow} w_{1} S \stackrel{*}{\Rightarrow} w_{1} w_{2}=w
$$

is possible.
Since the inductive assumption is clearly satisfied for $n=1$, we have a basis, and the claim is true for all $n$, completing our argument.

## Example 1.13 (continuation)

Once we see the argument intuitively, we are ready to proceed more rigorously. Again we use induction. Assume that all $w \in L$ with $|w| \leq 2 n$ can be derived with $G$. Take any $w \in L$ of length $2 n+2$. If $w=a w_{1} b$, then $w_{1}$ is in $L$, and $\left|w_{1}\right|=2 n$. Therefore, by assumption we have that

## Then

$$
S \Rightarrow a S b \stackrel{*}{\Rightarrow} a w_{1} b=w
$$

is possible, and $w$ can be derived with $G$. Obviously, similar arguments can be made if $w=b w_{1} a$.
If $w$ is not of this form, that is, if it starts and ends with the same symbol, then the counting argument tells us that it must have the form $w=w_{1} w_{2}$, with $w_{1}$ and $w_{2}$ both in $L$ and of length less than or equal to $2 n$. Hence again we see that
is possible.
Since the inductive assumption is clearly satisfied for $n=1$, we have a basis, and the claim is true for all $n$, completing our argument.

## Example 1.13 (continuation)

Once we see the argument intuitively, we are ready to proceed more rigorously. Again we use induction. with $G$. Take any $w \in L$ of length $2 n+2$. If $w=a w_{1} b$, then $w_{1}$ is in $L$, and $\left|w_{1}\right|=2 n$. Therefore, by assumption we have that

## Then

$$
S \Rightarrow a S b \stackrel{*}{\Rightarrow} a w_{1} b=w
$$

is possible, and $w$ can be derived with $G$. Obviously, similar arguments can be made if $w=b w_{1} a$.
If $w$ is not of this form, that is, if it starts and ends with the same symbol, then the counting argument tells us that it must have the form $w=w_{1} w_{2}$, with $w_{1}$ and $w_{2}$ both in $L$ and of length less than or equal to $2 n$. Hence again we see that
is possible.
Since the inductive assumption is clearly satisfied for $n=1$, we have a basis, and the claim is true for all $n$, completing our argument.

## Example 1.13 (continuation)

Once we see the argument intuitively, we are ready to proceed more rigorously. Again we use induction. Assume that all $w \in L$ with $|w| \leq 2 n$ can be derived with $G$.
$\left|w_{1}\right|=2 n$. Therefore, by assumption we have that

Then
$S \Rightarrow a S b \stackrel{*}{\Rightarrow} a w_{1} b=w$
is possible, and $w$ can be derived with $G$. Obviously, similar arguments can be made if $w=b w_{1} a$.
If $w$ is not of this form, that is, if it starts and ends with the same symbol, then the counting argument tells us that it must have the form $w=w_{1} w_{2}$, with $w_{1}$ and $w_{2}$ both in $L$ and of length less than or equal to $2 n$. Hence again we see that
$S \Rightarrow S S \stackrel{*}{\Rightarrow} w_{1} S \stackrel{*}{\Rightarrow} w_{1} w_{2}=w$
is possible.
Since the inductive assumption is clearly satisfied for $n=1$, we have a basis, and the claim is true for all $n$, completing our argument.

## Example 1.13 (continuation)

Once we see the argument intuitively, we are ready to proceed more rigorously. Again we use induction. Assume that all $w \in L$ with $|w| \leq 2 n$ can be derived with $G$. Take any $w \in L$ of length $2 n+2$.
$w_{1} \mid=2 n$. Therefore, by assumption we have that

Then
is possible, and $w$ can be derived with $G$. Obviously, similar arguments can be made if $w=b w_{1} a$.
If $w$ is not of this form, that is, if it starts and ends with the same symbol, then the counting argument tells us that it must have the form $w=w_{1} w_{2}$, with $w_{1}$ and $w_{2}$ both in $L$ and of length less than or equal to $2 n$. Hence again we see that
$S \Rightarrow S S \stackrel{*}{\Rightarrow} w_{1} S \stackrel{*}{\Rightarrow} w_{1} w_{2}=w$
is possible.
Since the inductive assumption is clearly satisfied for $n=1$, we have a basis, and the claim is true for all $n$, completing our argument.

## Example 1.13 (continuation)

Once we see the argument intuitively, we are ready to proceed more rigorously. Again we use induction. Assume that all $w \in L$ with $|w| \leq 2 n$ can be derived with $G$. Take any $w \in L$ of length $2 n+2$. If $w=a w_{1} b$, then $w_{1}$ is in $L$, and $\left|w_{1}\right|=2 n$. Therefore, by assumption we have that
is possible, and $w$ can be derived with $G$. Obviously, similar arguments can be made if $w=b w_{1} a$.
If $w$ is not of this form, that is, if it starts and ends with the same symbol, then the counting argument tells us that it must have the form $w=w_{1} w_{2}$, with $w_{1}$ and $w_{2}$ both in $L$ and of length less than or equal to $2 n$. Hence again we see that
$S \Rightarrow S S \stackrel{*}{\Rightarrow} w_{1} S^{*} w_{1} w_{2}=w$
is possible.
Since the inductive assumption is clearly satisfied for $n=1$, we have a basis, and the claim is true for all $n$, completing our argument.

## Example 1.13 (continuation)

Once we see the argument intuitively, we are ready to proceed more rigorously. Again we use induction. Assume that all $w \in L$ with $|w| \leq 2 n$ can be derived with $G$. Take any $w \in L$ of length $2 n+2$. If $w=a w_{1} b$, then $w_{1}$ is in $L$, and $\left|w_{1}\right|=2 n$. Therefore, by assumption we have that

Then
is possible, and $w$ can be derived with $G$. Obviously, similar arguments can be made if $w=b w_{1} a$.
If $w$ is not of this form, that is, if it starts and ends with the same symbol, then the counting argument tells us that it must have the form $w=w_{1} w_{2}$, with $w_{1}$ and $w_{2}$ both in $L$ and of length less than or equal to $2 n$. Hence again we see that
$S \Rightarrow S S \stackrel{*}{\Rightarrow} w_{1} S \stackrel{*}{\Rightarrow} w_{1} w_{2}=w$
is possible.
Since the inductive assumption is clearly satisfied for $n=1$, we have a basis, and the claim is true for all $n$, completing our argument.

## Example 1.13 (continuation)

Once we see the argument intuitively, we are ready to proceed more rigorously. Again we use induction. Assume that all $w \in L$ with $|w| \leq 2 n$ can be derived with $G$. Take any $w \in L$ of length $2 n+2$. If $w=a w_{1} b$, then $w_{1}$ is in $L$, and $\left|w_{1}\right|=2 n$. Therefore, by assumption we have that
is possible, and $w$ can be derived with $G$. Obviously, similar arguments can be made if $w=b w_{1} a$.
If $w$ is not of this form, that is, if it starts and ends with the same symbol, then the counting argument tells us that it must have the form $w=w_{1} w_{2}$, with $w_{1}$ and $w_{2}$ both in $L$ and of length less than or equal to $2 n$. Hence again we see that
$S \Rightarrow S S \stackrel{*}{\Rightarrow} w_{1} S \stackrel{*}{\Rightarrow} w_{1} w_{2}=w$
is possible
Since the inductive assumption is clearly satisfied for $n=1$, we have a basis, and the claim is true for all $n$, completing our argument.

## Example 1.13 (continuation)

Once we see the argument intuitively, we are ready to proceed more rigorously. Again we use induction. Assume that all $w \in L$ with $|w| \leq 2 n$ can be derived with $G$. Take any $w \in L$ of length $2 n+2$. If $w=a w_{1} b$, then $w_{1}$ is in $L$, and $\left|w_{1}\right|=2 n$. Therefore, by assumption we have that

$$
S \stackrel{*}{\Rightarrow} w_{1} .
$$

Then
is possible, and $w$ can be derived with $G$. Obviously, similar arguments can be made if $w=b w_{1} a$.
If $w$ is not of this form, that is, if it starts and ends with the same symbol, then the counting argument tells us that it must have the form $w=w_{1} w_{2}$, with $w_{1}$ and $w_{2}$ both in $L$ and of length less than or equal to $2 n$. Hence again we see that
$S \Rightarrow S S \stackrel{*}{\Rightarrow} w_{1} S \stackrel{*}{\Rightarrow} w_{1} w_{2}=w$
is possible.
Since the inductive assumption is clearly satisfied for $n=1$, we have a basis, and the claim is true for all $n$, completing our argument.

## Example 1.13 (continuation)

Once we see the argument intuitively, we are ready to proceed more rigorously. Again we use induction. Assume that all $w \in L$ with $|w| \leq 2 n$ can be derived with $G$. Take any $w \in L$ of length $2 n+2$. If $w=a w_{1} b$, then $w_{1}$ is in $L$, and $\left|w_{1}\right|=2 n$. Therefore, by assumption we have that

$$
S \stackrel{*}{\Rightarrow} w_{1} .
$$

Then
is possible, and $w$ can be derived with $G$. Obviously, similar arguments can be made if $w=b w_{1} a$.
If $w$ is not of this form, that is, if it starts and ends with the same symbol, then the counting argument tells us that it must have the form $w=w_{1} w_{2}$, with $w_{1}$ and $w_{2}$ both in $L$ and of length less than or equal to $2 n$. Hence again we see that
$S \Rightarrow S S \stackrel{*}{\Rightarrow} w_{1} S \stackrel{*}{\Rightarrow} w_{1} w_{2}=w$
is possible
Since the inductive assumption is clearly satisfied for $n=1$, we have a basis, and the claim is true for all $n$, completing our argument.

## Example 1.13 (continuation)

Once we see the argument intuitively, we are ready to proceed more rigorously. Again we use induction. Assume that all $w \in L$ with $|w| \leq 2 n$ can be derived with $G$. Take any $w \in L$ of length $2 n+2$. If $w=a w_{1} b$, then $w_{1}$ is in $L$, and $\left|w_{1}\right|=2 n$. Therefore, by assumption we have that

$$
S \stackrel{*}{\Rightarrow} w_{1} .
$$

Then

$$
S \Rightarrow a S b \stackrel{*}{\Rightarrow} a w_{1} b=w
$$

is possible, and $w$ can be derived with $G$. Obviously, similar arguments can be made if $w=b w_{1} a$.
If $w$ is not of this form, that is, if it starts and ends with the same symbol, then the counting argument tells us that it must have the form $w=w_{1} w_{2}$, with $w_{1}$ and $w_{2}$ both in $L$ and of length less than or equal to $2 n$. Hence again we see that
$S \Rightarrow S S \stackrel{*}{\Rightarrow} w_{1} S \stackrel{*}{\Rightarrow} w_{1} w_{2}=w$
is possible.
Since the inductive assumption is clearly satisfied for $n=1$, we have a basis, and the claim is true for all $n$, completing our argument.

## Example 1.13 (continuation)

Once we see the argument intuitively, we are ready to proceed more rigorously. Again we use induction. Assume that all $w \in L$ with $|w| \leq 2 n$ can be derived with $G$. Take any $w \in L$ of length $2 n+2$. If $w=a w_{1} b$, then $w_{1}$ is in $L$, and $\left|w_{1}\right|=2 n$. Therefore, by assumption we have that

$$
S \stackrel{*}{\Rightarrow} w_{1} .
$$

Then

$$
S \Rightarrow a S b \stackrel{*}{\Rightarrow} a w_{1} b=w
$$

is possible, and $w$ can be derived with $G$. Obviously, similar arguments can be made if $w=b w_{1} a$.
If $w$ is not of this form, that is, if it starts and ends with the same symbol, then the counting argument tells us that it must have the form $w=w_{1} w_{2}$, with $w_{1}$ and $w_{2}$ both in $L$ and of length less than or equal to $2 n$. Hence again we see that
$\qquad$
is possible
Since the inductive assumption is clearly satisfied for $n=1$, we have a basis, and the claim is true for all $n$, completing our argument.

## Example 1.13 (continuation)

Once we see the argument intuitively, we are ready to proceed more rigorously. Again we use induction. Assume that all $w \in L$ with $|w| \leq 2 n$ can be derived with $G$. Take any $w \in L$ of length $2 n+2$. If $w=a w_{1} b$, then $w_{1}$ is in $L$, and $\left|w_{1}\right|=2 n$. Therefore, by assumption we have that

$$
S \stackrel{*}{\Rightarrow} w_{1} .
$$

Then

$$
S \Rightarrow a S b \stackrel{*}{\Rightarrow} a w_{1} b=w
$$

is possible, and $w$ can be derived with $G$.
Obviously, similar arguments can be
If $w$ is not of this form, that is, if it starts and ends with the same symbol, then the counting argument tells us that it must have the form $w=w_{1} w_{2}$, with $w_{1}$ and $w_{2}$ both in $L$ and of length less than or equal to $2 n$. Hence again we see that
$S \Rightarrow S S \stackrel{*}{\Rightarrow} w_{1} S \stackrel{*}{\Rightarrow} w_{1} w_{2}=w$
is possible.
Since the inductive assumption is clearly satisfied for $n=1$, we have a basis, and the claim is true for all $n$, completing our argument.

## Example 1.13 (continuation)

Once we see the argument intuitively, we are ready to proceed more rigorously. Again we use induction. Assume that all $w \in L$ with $|w| \leq 2 n$ can be derived with $G$. Take any $w \in L$ of length $2 n+2$. If $w=a w_{1} b$, then $w_{1}$ is in $L$, and $\left|w_{1}\right|=2 n$. Therefore, by assumption we have that

$$
S \stackrel{*}{\Rightarrow} w_{1} .
$$

Then

$$
S \Rightarrow a S b \stackrel{*}{\Rightarrow} a w_{1} b=w
$$

is possible, and $w$ can be derived with $G$. Obviously, similar arguments can be made if $w=b w_{1} a$.
If $w$ is not of this form, that is, if it starts and ends with the same symbol, then the counting argument tells us that it must have the form $w=w_{1} w_{2}$, with $w_{1}$ and $w_{2}$ both in $L$ and of length less than or equal to $2 n$. Hence again we see that
is possible
Since the inductive assumption is clearly satisfied for $n=1$, we have a basis, and the claim is true for all $n$, completing our argument.

## Example 1.13 (continuation)

Once we see the argument intuitively, we are ready to proceed more rigorously. Again we use induction. Assume that all $w \in L$ with $|w| \leq 2 n$ can be derived with $G$. Take any $w \in L$ of length $2 n+2$. If $w=a w_{1} b$, then $w_{1}$ is in $L$, and $\left|w_{1}\right|=2 n$. Therefore, by assumption we have that

$$
S \stackrel{*}{\Rightarrow} w_{1} .
$$

Then

$$
S \Rightarrow a S b \stackrel{*}{\Rightarrow} a w_{1} b=w
$$

is possible, and $w$ can be derived with $G$. Obviously, similar arguments can be made if $w=b w_{1} a$.
If $w$ is not of this form,
the counting argument tells us that it must have the form $w=w_{1} w_{2}$, with $w_{1}$ and $w_{2}$ both in $L$ and of length less than or equal to $2 n$. Hence again we see that
is possible
Since the inductive assumption is clearly satisfied for $n=1$, we have a basis, and the claim is true for all $n$, completing our argument.

## Example 1.13 (continuation)

Once we see the argument intuitively, we are ready to proceed more rigorously. Again we use induction. Assume that all $w \in L$ with $|w| \leq 2 n$ can be derived with $G$. Take any $w \in L$ of length $2 n+2$. If $w=a w_{1} b$, then $w_{1}$ is in $L$, and $\left|w_{1}\right|=2 n$. Therefore, by assumption we have that

$$
S \stackrel{*}{\Rightarrow} w_{1} .
$$

Then

$$
S \Rightarrow a S b \stackrel{*}{\Rightarrow} a w_{1} b=w
$$

is possible, and $w$ can be derived with $G$. Obviously, similar arguments can be made if $w=b w_{1} a$.
If $w$ is not of this form, that is, if it starts and ends with the same symbol, and $w_{2}$ both in $L$ and of length less than or equal to $2 n$. Hence again we see that
is possible
Since the inductive assumption is clearly satisfied for $n=1$, we have a basis, and the claim is true for all $n$, completing our argument.

## Example 1.13 (continuation)

Once we see the argument intuitively, we are ready to proceed more rigorously. Again we use induction. Assume that all $w \in L$ with $|w| \leq 2 n$ can be derived with $G$. Take any $w \in L$ of length $2 n+2$. If $w=a w_{1} b$, then $w_{1}$ is in $L$, and $\left|w_{1}\right|=2 n$. Therefore, by assumption we have that

$$
S \stackrel{*}{\Rightarrow} w_{1} .
$$

Then

$$
S \Rightarrow a S b \stackrel{*}{\Rightarrow} a w_{1} b=w
$$

is possible, and $w$ can be derived with $G$. Obviously, similar arguments can be made if $w=b w_{1} a$.
If $w$ is not of this form, that is, if it starts and ends with the same symbol, then the counting argument tells us that it must have the form $w=w_{1} w_{2}$,
and $w_{2}$ both in $L$ and of length less than or equal to $2 n$. Hence again we see that
is possible
Since the inductive assumption is clearly satisfied for $n=1$, we have a basis, and the claim is true for all $n$, completing our argument.

## Example 1.13 (continuation)

Once we see the argument intuitively, we are ready to proceed more rigorously. Again we use induction. Assume that all $w \in L$ with $|w| \leq 2 n$ can be derived with $G$. Take any $w \in L$ of length $2 n+2$. If $w=a w_{1} b$, then $w_{1}$ is in $L$, and $\left|w_{1}\right|=2 n$. Therefore, by assumption we have that

$$
S \stackrel{*}{\Rightarrow} w_{1} .
$$

Then

$$
S \Rightarrow a S b \stackrel{*}{\Rightarrow} a w_{1} b=w
$$

is possible, and $w$ can be derived with $G$. Obviously, similar arguments can be made if $w=b w_{1} a$.
If $w$ is not of this form, that is, if it starts and ends with the same symbol, then the counting argument tells us that it must have the form $w=w_{1} w_{2}$, with $w_{1}$ and $w_{2}$ both in $L$ and of length less than or equal to $2 n$. that
is possible
Since the inductive assumption is clearly satisfied for $n=1$, we have a basis, and the claim is true for all $n$, completing our argument.

## Example 1.13 (continuation)

Once we see the argument intuitively, we are ready to proceed more rigorously. Again we use induction. Assume that all $w \in L$ with $|w| \leq 2 n$ can be derived with $G$. Take any $w \in L$ of length $2 n+2$. If $w=a w_{1} b$, then $w_{1}$ is in $L$, and $\left|w_{1}\right|=2 n$. Therefore, by assumption we have that

$$
S \stackrel{*}{\Rightarrow} w_{1} .
$$

Then

$$
S \Rightarrow a S b \stackrel{*}{\Rightarrow} a w_{1} b=w
$$

is possible, and $w$ can be derived with $G$. Obviously, similar arguments can be made if $w=b w_{1} a$.
If $w$ is not of this form, that is, if it starts and ends with the same symbol, then the counting argument tells us that it must have the form $w=w_{1} w_{2}$, with $w_{1}$ and $w_{2}$ both in $L$ and of length less than or equal to $2 n$. Hence again we see that
is possible
Since the inductive assumption is clearly satisfied for $n=1$, we have a basis, and the claim is true for all $n$, completing our argument.

## Example 1.13 (continuation)

Once we see the argument intuitively, we are ready to proceed more rigorously. Again we use induction. Assume that all $w \in L$ with $|w| \leq 2 n$ can be derived with $G$. Take any $w \in L$ of length $2 n+2$. If $w=a w_{1} b$, then $w_{1}$ is in $L$, and $\left|w_{1}\right|=2 n$. Therefore, by assumption we have that

$$
S \stackrel{*}{\Rightarrow} w_{1} .
$$

Then

$$
S \Rightarrow a S b \stackrel{*}{\Rightarrow} a w_{1} b=w
$$

is possible, and $w$ can be derived with $G$. Obviously, similar arguments can be made if $w=b w_{1} a$.
If $w$ is not of this form, that is, if it starts and ends with the same symbol, then the counting argument tells us that it must have the form $w=w_{1} w_{2}$, with $w_{1}$ and $w_{2}$ both in $L$ and of length less than or equal to $2 n$. Hence again we see that

$$
S \Rightarrow S S \stackrel{*}{\Rightarrow} w_{1} S \stackrel{*}{\Rightarrow} w_{1} w_{2}=w
$$

is possible.
Since the inductive assumption is clearly satisfied for $n=1$, we have a basis, and the claim is true for all $n$, completing our argument.

## Example 1.13 (continuation)

Once we see the argument intuitively, we are ready to proceed more rigorously. Again we use induction. Assume that all $w \in L$ with $|w| \leq 2 n$ can be derived with $G$. Take any $w \in L$ of length $2 n+2$. If $w=a w_{1} b$, then $w_{1}$ is in $L$, and $\left|w_{1}\right|=2 n$. Therefore, by assumption we have that

$$
S \stackrel{*}{\Rightarrow} w_{1} .
$$

Then

$$
S \Rightarrow a S b \stackrel{*}{\Rightarrow} a w_{1} b=w
$$

is possible, and $w$ can be derived with $G$. Obviously, similar arguments can be made if $w=b w_{1} a$.
If $w$ is not of this form, that is, if it starts and ends with the same symbol, then the counting argument tells us that it must have the form $w=w_{1} w_{2}$, with $w_{1}$ and $w_{2}$ both in $L$ and of length less than or equal to $2 n$. Hence again we see that

$$
S \Rightarrow S S \stackrel{*}{\Rightarrow} w_{1} S \stackrel{*}{\Rightarrow} w_{1} w_{2}=w
$$

is possible.
Since the inductive assumption is clearly satisfied for $n=1$, we have a basis, and the claim is true for all $n$, completing our argument.

## Example 1.13 (continuation)

Once we see the argument intuitively, we are ready to proceed more rigorously. Again we use induction. Assume that all $w \in L$ with $|w| \leq 2 n$ can be derived with $G$. Take any $w \in L$ of length $2 n+2$. If $w=a w_{1} b$, then $w_{1}$ is in $L$, and $\left|w_{1}\right|=2 n$. Therefore, by assumption we have that

$$
S \stackrel{*}{\Rightarrow} w_{1} .
$$

Then

$$
S \Rightarrow a S b \stackrel{*}{\Rightarrow} a w_{1} b=w
$$

is possible, and $w$ can be derived with $G$. Obviously, similar arguments can be made if $w=b w_{1} a$.
If $w$ is not of this form, that is, if it starts and ends with the same symbol, then the counting argument tells us that it must have the form $w=w_{1} w_{2}$, with $w_{1}$ and $w_{2}$ both in $L$ and of length less than or equal to $2 n$. Hence again we see that

$$
S \Rightarrow S S \stackrel{*}{\Rightarrow} w_{1} S \stackrel{*}{\Rightarrow} w_{1} w_{2}=w
$$

is possible.
Since the inductive assumption is clearly satisfied for $n=1$, we have a basis,

## Example 1.13 (continuation)

Once we see the argument intuitively, we are ready to proceed more rigorously. Again we use induction. Assume that all $w \in L$ with $|w| \leq 2 n$ can be derived with $G$. Take any $w \in L$ of length $2 n+2$. If $w=a w_{1} b$, then $w_{1}$ is in $L$, and $\left|w_{1}\right|=2 n$. Therefore, by assumption we have that

$$
S \stackrel{*}{\Rightarrow} w_{1} .
$$

Then

$$
S \Rightarrow a S b \stackrel{*}{\Rightarrow} a w_{1} b=w
$$

is possible, and $w$ can be derived with $G$. Obviously, similar arguments can be made if $w=b w_{1} a$.
If $w$ is not of this form, that is, if it starts and ends with the same symbol, then the counting argument tells us that it must have the form $w=w_{1} w_{2}$, with $w_{1}$ and $w_{2}$ both in $L$ and of length less than or equal to $2 n$. Hence again we see that

$$
S \Rightarrow S S \stackrel{*}{\Rightarrow} w_{1} S \stackrel{*}{\Rightarrow} w_{1} w_{2}=w
$$

is possible.
Since the inductive assumption is clearly satisfied for $n=1$, we have a basis,

## Example 1.13 (continuation)

Once we see the argument intuitively, we are ready to proceed more rigorously. Again we use induction. Assume that all $w \in L$ with $|w| \leq 2 n$ can be derived with $G$. Take any $w \in L$ of length $2 n+2$. If $w=a w_{1} b$, then $w_{1}$ is in $L$, and $\left|w_{1}\right|=2 n$. Therefore, by assumption we have that

$$
S \stackrel{*}{\Rightarrow} w_{1} .
$$

Then

$$
S \Rightarrow a S b \stackrel{*}{\Rightarrow} a w_{1} b=w
$$

is possible, and $w$ can be derived with $G$. Obviously, similar arguments can be made if $w=b w_{1} a$.
If $w$ is not of this form, that is, if it starts and ends with the same symbol, then the counting argument tells us that it must have the form $w=w_{1} w_{2}$, with $w_{1}$ and $w_{2}$ both in $L$ and of length less than or equal to $2 n$. Hence again we see that

$$
S \Rightarrow S S \stackrel{*}{\Rightarrow} w_{1} S \stackrel{*}{\Rightarrow} w_{1} w_{2}=w
$$

is possible.
Since the inductive assumption is clearly satisfied for $n=1$, we have a basis, and the claim is true for all $n$, completing our argument.

## Example 1.13 (continuation)

Once we see the argument intuitively, we are ready to proceed more rigorously. Again we use induction. Assume that all $w \in L$ with $|w| \leq 2 n$ can be derived with $G$. Take any $w \in L$ of length $2 n+2$. If $w=a w_{1} b$, then $w_{1}$ is in $L$, and $\left|w_{1}\right|=2 n$. Therefore, by assumption we have that

$$
S \stackrel{*}{\Rightarrow} w_{1} .
$$

Then

$$
S \Rightarrow a S b \stackrel{*}{\Rightarrow} a w_{1} b=w
$$

is possible, and $w$ can be derived with $G$. Obviously, similar arguments can be made if $w=b w_{1} a$.
If $w$ is not of this form, that is, if it starts and ends with the same symbol, then the counting argument tells us that it must have the form $w=w_{1} w_{2}$, with $w_{1}$ and $w_{2}$ both in $L$ and of length less than or equal to $2 n$. Hence again we see that

$$
S \Rightarrow S S \stackrel{*}{\Rightarrow} w_{1} S \stackrel{*}{\Rightarrow} w_{1} w_{2}=w
$$

is possible.
Since the inductive assumption is clearly satisfied for $n=1$, we have a basis, and the claim is true for all $n$, completing our argument.

## Example 1.13 (continuation)

Once we see the argument intuitively, we are ready to proceed more rigorously. Again we use induction. Assume that all $w \in L$ with $|w| \leq 2 n$ can be derived with $G$. Take any $w \in L$ of length $2 n+2$. If $w=a w_{1} b$, then $w_{1}$ is in $L$, and $\left|w_{1}\right|=2 n$. Therefore, by assumption we have that

$$
S \stackrel{*}{\Rightarrow} w_{1} .
$$

Then

$$
S \Rightarrow a S b \stackrel{*}{\Rightarrow} a w_{1} b=w
$$

is possible, and $w$ can be derived with $G$. Obviously, similar arguments can be made if $w=b w_{1} a$.
If $w$ is not of this form, that is, if it starts and ends with the same symbol, then the counting argument tells us that it must have the form $w=w_{1} w_{2}$, with $w_{1}$ and $w_{2}$ both in $L$ and of length less than or equal to $2 n$. Hence again we see that

$$
S \Rightarrow S S \stackrel{*}{\Rightarrow} w_{1} S \stackrel{*}{\Rightarrow} w_{1} w_{2}=w
$$

is possible.
Since the inductive assumption is clearly satisfied for $n=1$, we have a basis, and the claim is true for all $n$, completing our argument.

## 2 THREE BASIC CONCEPTS: Grammars

```
Normally, a given language has many grammars that generate it. Even though these
grammars are different, they are equivalent in some sense. We say that two grammars
\(G_{1}\) and \(G_{2}\) are equivalent if they generate the same language, that is, if
\(L\left(G_{1}\right)=L\left(G_{2}\right)\).
As we will see later, it is not always easy to see if two grammars are equivalent.
```

Example 1.14

## 2 THREE BASIC CONCEPTS: Grammars

Normally, a given language has many grammars that generate it. Even though these grammars are different, they are equivalent in some sense. We say that two grammars $G_{1}$ and $G_{2}$ are equivalent if they generate the same language, that is, if

```
    L(G1) = L(G2).
```

As we will see later, it is not always easy to see if two grammars are equivalent.

## Example 1.14

## 2 THREE BASIC CONCEPTS: Grammars

Normally, a given language has many grammars that generate it. Even though these grammars are different, they are equivalent in some sense. We say that two grammars $G_{1}$ and $G_{2}$ are equivalent if they generate the same language, that is, if $L\left(G_{1}\right)=L\left(G_{2}\right)$ As we will see later, it is not always easy to see if two grammars are equivalent.

```
Example 1.14
```


## 2 THREE BASIC CONCEPTS: Grammars

Normally, a given language has many grammars that generate it. Even though these grammars are different, they are equivalent in some sense.

We say that two grammars $G_{1}$ and $G_{2}$ are equivalent if they generate the same language, that is, if $L\left(G_{1}\right)=L\left(G_{2}\right)$. As we will see later, it is not always easy to see if two grammars are equivalent.

```
Examole 1.14
```


## 2 THREE BASIC CONCEPTS: Grammars

Normally, a given language has many grammars that generate it. Even though these grammars are different, they are equivalent in some sense. We say that two grammars $G_{1}$ and $G_{2}$ are equivalent if they generate the same language,

As we will see later, it is not always easy to see if two grammars are equivalent.

## Example 1.14

## 2 THREE BASIC CONCEPTS: Grammars

Normally, a given language has many grammars that generate it. Even though these grammars are different, they are equivalent in some sense. We say that two grammars $G_{1}$ and $G_{2}$ are equivalent if they generate the same language, that is, if

As we will see later, it is not always easy to see if two grammars are equivalent.

## Example 1.14

## 2 THREE BASIC CONCEPTS: Grammars

Normally, a given language has many grammars that generate it. Even though these grammars are different, they are equivalent in some sense. We say that two grammars $G_{1}$ and $G_{2}$ are equivalent if they generate the same language, that is, if

$$
L\left(G_{1}\right)=L\left(G_{2}\right) .
$$

As we will see later, it is not always easy to see if two grammars are equivalent.

## Example 1.14

## 2 THREE BASIC CONCEPTS: Grammars

Normally, a given language has many grammars that generate it. Even though these grammars are different, they are equivalent in some sense. We say that two grammars $G_{1}$ and $G_{2}$ are equivalent if they generate the same language, that is, if

$$
L\left(G_{1}\right)=L\left(G_{2}\right) .
$$

As we will see later, it is not always easy to see if two grammars are equivalent.

## 2 THREE BASIC CONCEPTS: Grammars

Normally, a given language has many grammars that generate it. Even though these grammars are different, they are equivalent in some sense. We say that two grammars $G_{1}$ and $G_{2}$ are equivalent if they generate the same language, that is, if

$$
L\left(G_{1}\right)=L\left(G_{2}\right) .
$$

As we will see later, it is not always easy to see if two grammars are equivalent.

## Example 1.14

Consider the grammar $G_{1}=\left(\{A, S\},\{a, b\}, S, P_{1}\right)$, with $P_{1}$ consisting of the productions

Here we introduce a convenient shorthand notation in which several production rules with the same left-hand sides are written on the same line, with alternative right-hand sides separated by the symbol |. In this notation $S \rightarrow a A b \mid \lambda$ stands for the two productions $S \rightarrow a A b$ and $S \rightarrow \lambda$. This grammar is equivalent to the grammar $G$ in Example 1.11

```
with P given by
```

The equivalence is easy to prove by showing that

## 2 THREE BASIC CONCEPTS: Grammars

Normally, a given language has many grammars that generate it. Even though these grammars are different, they are equivalent in some sense. We say that two grammars $G_{1}$ and $G_{2}$ are equivalent if they generate the same language, that is, if

$$
L\left(G_{1}\right)=L\left(G_{2}\right) .
$$

As we will see later, it is not always easy to see if two grammars are equivalent.

## Example 1.14

Consider the grammar $G_{1}=\left(\{A, S\},\{a, b\}, S, P_{1}\right)$, with $P_{1}$ consisting of the productions

Here we introduce a convenient shorthand notation in which several production rules with the same left-hand sides are written on the same line, with alternative right-hand sides separated by the symbol |. In this notation $S \rightarrow a A b \mid \lambda$ stands for the two productions $S \rightarrow a A b$ and $S \rightarrow \lambda$. This grammar is equivalent to the grammar $G$ in Example 1.11 with $P$ given by

The equivalence is easy to prove by showing that

## 2 THREE BASIC CONCEPTS: Grammars

Normally, a given language has many grammars that generate it. Even though these grammars are different, they are equivalent in some sense. We say that two grammars $G_{1}$ and $G_{2}$ are equivalent if they generate the same language, that is, if

$$
L\left(G_{1}\right)=L\left(G_{2}\right) .
$$

As we will see later, it is not always easy to see if two grammars are equivalent.

## Example 1.14

Consider the grammar $G_{1}=\left(\{A, S\},\{a, b\}, S, P_{1}\right)$, with $P_{1}$ consisting of the productions

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

Here we introduce a convenient shorthand notation in which several production rules with the same left-hand sides are written on the same line, with alternative right-hand sides separated by the symbol $\mid$. In this notation $S \rightarrow a A b \mid \lambda$ stands for the two productions $S \rightarrow a A b$ and $S \rightarrow \lambda$. This grammar is equivalent to the grammar $G$ in Example 1.11
$\qquad$

The equivalence is easy to prove by showing that

## 2 THREE BASIC CONCEPTS: Grammars

Normally, a given language has many grammars that generate it. Even though these grammars are different, they are equivalent in some sense. We say that two grammars $G_{1}$ and $G_{2}$ are equivalent if they generate the same language, that is, if

$$
L\left(G_{1}\right)=L\left(G_{2}\right) .
$$

As we will see later, it is not always easy to see if two grammars are equivalent.

## Example 1.14

Consider the grammar $G_{1}=\left(\{A, S\},\{a, b\}, S, P_{1}\right)$, with $P_{1}$ consisting of the productions

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

Here we introduce a convenient shorthand notation in which several production rules with the same left-hand sides are written on the same line,

## 2 THREE BASIC CONCEPTS: Grammars

Normally, a given language has many grammars that generate it. Even though these grammars are different, they are equivalent in some sense. We say that two grammars $G_{1}$ and $G_{2}$ are equivalent if they generate the same language, that is, if

$$
L\left(G_{1}\right)=L\left(G_{2}\right) .
$$

As we will see later, it is not always easy to see if two grammars are equivalent.

## Example 1.14

Consider the grammar $G_{1}=\left(\{A, S\},\{a, b\}, S, P_{1}\right)$, with $P_{1}$ consisting of the productions

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

Here we introduce a convenient shorthand notation in which several production rules with the same left-hand sides are written on the same line, with alternative right-hand sides separated by the symbol |.
productions $S \rightarrow a A b$ and $S \rightarrow \lambda$. This grammar is equivalent to the grammar $G$ in Example 1.11
$\qquad$

The equivalence is easy to prove by showing that

## 2 THREE BASIC CONCEPTS: Grammars

Normally, a given language has many grammars that generate it. Even though these grammars are different, they are equivalent in some sense. We say that two grammars $G_{1}$ and $G_{2}$ are equivalent if they generate the same language, that is, if

$$
L\left(G_{1}\right)=L\left(G_{2}\right)
$$

As we will see later, it is not always easy to see if two grammars are equivalent.

## Example 1.14

Consider the grammar $G_{1}=\left(\{A, S\},\{a, b\}, S, P_{1}\right)$, with $P_{1}$ consisting of the productions

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

Here we introduce a convenient shorthand notation in which several production rules with the same left-hand sides are written on the same line, with alternative right-hand sides separated by the symbol |. In this notation $S \rightarrow a A b \mid \lambda$ stands for the two productions $S \rightarrow a A b$ and $S \rightarrow \lambda$. Example 1.11: with $P$ given by

The equivalence is easy to prove by showing that

## 2 THREE BASIC CONCEPTS: Grammars

Normally, a given language has many grammars that generate it. Even though these grammars are different, they are equivalent in some sense. We say that two grammars $G_{1}$ and $G_{2}$ are equivalent if they generate the same language, that is, if

$$
L\left(G_{1}\right)=L\left(G_{2}\right)
$$

As we will see later, it is not always easy to see if two grammars are equivalent.

## Example 1.14

Consider the grammar $G_{1}=\left(\{A, S\},\{a, b\}, S, P_{1}\right)$, with $P_{1}$ consisting of the productions

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

Here we introduce a convenient shorthand notation in which several production rules with the same left-hand sides are written on the same line, with alternative right-hand sides separated by the symbol $\mid$. In this notation $S \rightarrow a A b \mid \lambda$ stands for the two productions $S \rightarrow a A b$ and $S \rightarrow \lambda$. This grammar is equivalent to the grammar $G$ in Example 1.11:
with $P$ given by

The equivalence is easy to prove by showing that

We leave this as an exercise.

## 2 THREE BASIC CONCEPTS: Grammars

Normally, a given language has many grammars that generate it. Even though these grammars are different, they are equivalent in some sense. We say that two grammars $G_{1}$ and $G_{2}$ are equivalent if they generate the same language, that is, if

$$
L\left(G_{1}\right)=L\left(G_{2}\right)
$$

As we will see later, it is not always easy to see if two grammars are equivalent.

## Example 1.14

Consider the grammar $G_{1}=\left(\{A, S\},\{a, b\}, S, P_{1}\right)$, with $P_{1}$ consisting of the productions

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

Here we introduce a convenient shorthand notation in which several production rules with the same left-hand sides are written on the same line, with alternative right-hand sides separated by the symbol $\mid$. In this notation $S \rightarrow a A b \mid \lambda$ stands for the two productions $S \rightarrow a A b$ and $S \rightarrow \lambda$. This grammar is equivalent to the grammar $G$ in Example 1.11:

$$
G=(\{S\},\{a, b\}, S, P),
$$

The equivalence is easy to prove by showing that

## 2 THREE BASIC CONCEPTS: Grammars

Normally, a given language has many grammars that generate it. Even though these grammars are different, they are equivalent in some sense. We say that two grammars $G_{1}$ and $G_{2}$ are equivalent if they generate the same language, that is, if

$$
L\left(G_{1}\right)=L\left(G_{2}\right)
$$

As we will see later, it is not always easy to see if two grammars are equivalent.

## Example 1.14

Consider the grammar $G_{1}=\left(\{A, S\},\{a, b\}, S, P_{1}\right)$, with $P_{1}$ consisting of the productions

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

Here we introduce a convenient shorthand notation in which several production rules with the same left-hand sides are written on the same line, with alternative right-hand sides separated by the symbol $\mid$. In this notation $S \rightarrow a A b \mid \lambda$ stands for the two productions $S \rightarrow a A b$ and $S \rightarrow \lambda$. This grammar is equivalent to the grammar $G$ in Example 1.11:

$$
G=(\{S\},\{a, b\}, S, P),
$$

with $P$ given by

The equivalence is easy to prove by showing that

## 2 THREE BASIC CONCEPTS: Grammars

Normally, a given language has many grammars that generate it. Even though these grammars are different, they are equivalent in some sense. We say that two grammars $G_{1}$ and $G_{2}$ are equivalent if they generate the same language, that is, if

$$
L\left(G_{1}\right)=L\left(G_{2}\right)
$$

As we will see later, it is not always easy to see if two grammars are equivalent.

## Example 1.14

Consider the grammar $G_{1}=\left(\{A, S\},\{a, b\}, S, P_{1}\right)$, with $P_{1}$ consisting of the productions

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

Here we introduce a convenient shorthand notation in which several production rules with the same left-hand sides are written on the same line, with alternative right-hand sides separated by the symbol $\mid$. In this notation $S \rightarrow a A b \mid \lambda$ stands for the two productions $S \rightarrow a A b$ and $S \rightarrow \lambda$. This grammar is equivalent to the grammar $G$ in Example 1.11:

$$
G=(\{S\},\{a, b\}, S, P),
$$

with $P$ given by

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

The equivalence is easy to prove by showing that

We leave this as an exercise.

## 2 THREE BASIC CONCEPTS: Grammars

Normally, a given language has many grammars that generate it. Even though these grammars are different, they are equivalent in some sense. We say that two grammars $G_{1}$ and $G_{2}$ are equivalent if they generate the same language, that is, if

$$
L\left(G_{1}\right)=L\left(G_{2}\right)
$$

As we will see later, it is not always easy to see if two grammars are equivalent.

## Example 1.14

Consider the grammar $G_{1}=\left(\{A, S\},\{a, b\}, S, P_{1}\right)$, with $P_{1}$ consisting of the productions

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

Here we introduce a convenient shorthand notation in which several production rules with the same left-hand sides are written on the same line, with alternative right-hand sides separated by the symbol $\mid$. In this notation $S \rightarrow a A b \mid \lambda$ stands for the two productions $S \rightarrow a A b$ and $S \rightarrow \lambda$. This grammar is equivalent to the grammar $G$ in Example 1.11:

$$
G=(\{S\},\{a, b\}, S, P),
$$

with $P$ given by

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

The equivalence is easy to prove by showing that

[^18]
## 2 THREE BASIC CONCEPTS: Grammars

Normally, a given language has many grammars that generate it. Even though these grammars are different, they are equivalent in some sense. We say that two grammars $G_{1}$ and $G_{2}$ are equivalent if they generate the same language, that is, if

$$
L\left(G_{1}\right)=L\left(G_{2}\right)
$$

As we will see later, it is not always easy to see if two grammars are equivalent.

## Example 1.14

Consider the grammar $G_{1}=\left(\{A, S\},\{a, b\}, S, P_{1}\right)$, with $P_{1}$ consisting of the productions

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

Here we introduce a convenient shorthand notation in which several production rules with the same left-hand sides are written on the same line, with alternative right-hand sides separated by the symbol $\mid$. In this notation $S \rightarrow a A b \mid \lambda$ stands for the two productions $S \rightarrow a A b$ and $S \rightarrow \lambda$. This grammar is equivalent to the grammar $G$ in Example 1.11:

$$
G=(\{S\},\{a, b\}, S, P),
$$

with $P$ given by

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

The equivalence is easy to prove by showing that

$$
L\left(G_{1}\right)=\left\{a^{n} b^{n}: n \geq 0\right\} .
$$

[^19]
## 2 THREE BASIC CONCEPTS: Grammars

Normally, a given language has many grammars that generate it. Even though these grammars are different, they are equivalent in some sense. We say that two grammars $G_{1}$ and $G_{2}$ are equivalent if they generate the same language, that is, if

$$
L\left(G_{1}\right)=L\left(G_{2}\right)
$$

As we will see later, it is not always easy to see if two grammars are equivalent.

## Example 1.14

Consider the grammar $G_{1}=\left(\{A, S\},\{a, b\}, S, P_{1}\right)$, with $P_{1}$ consisting of the productions

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

Here we introduce a convenient shorthand notation in which several production rules with the same left-hand sides are written on the same line, with alternative right-hand sides separated by the symbol $\mid$. In this notation $S \rightarrow a A b \mid \lambda$ stands for the two productions $S \rightarrow a A b$ and $S \rightarrow \lambda$. This grammar is equivalent to the grammar $G$ in Example 1.11:

$$
G=(\{S\},\{a, b\}, S, P),
$$

with $P$ given by

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

The equivalence is easy to prove by showing that

$$
L\left(G_{1}\right)=\left\{a^{n} b^{n}: n \geq 0\right\} .
$$

We leave this as an exercise.

## 2 THREE BASIC CONCEPTS: Grammars

Normally, a given language has many grammars that generate it. Even though these grammars are different, they are equivalent in some sense. We say that two grammars $G_{1}$ and $G_{2}$ are equivalent if they generate the same language, that is, if

$$
L\left(G_{1}\right)=L\left(G_{2}\right)
$$

As we will see later, it is not always easy to see if two grammars are equivalent.

## Example 1.14

Consider the grammar $G_{1}=\left(\{A, S\},\{a, b\}, S, P_{1}\right)$, with $P_{1}$ consisting of the productions

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

Here we introduce a convenient shorthand notation in which several production rules with the same left-hand sides are written on the same line, with alternative right-hand sides separated by the symbol $\mid$. In this notation $S \rightarrow a A b \mid \lambda$ stands for the two productions $S \rightarrow a A b$ and $S \rightarrow \lambda$. This grammar is equivalent to the grammar $G$ in Example 1.11:

$$
G=(\{S\},\{a, b\}, S, P),
$$

with $P$ given by

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

The equivalence is easy to prove by showing that

$$
L\left(G_{1}\right)=\left\{a^{n} b^{n}: n \geq 0\right\} .
$$

We leave this as an exercise.

## 2 THREE BASIC CONCEPTS: Automata

An automaton is an abstract model of a digital computer. As such, every automaton includes some essential features. It has a mechanism for reading input. It will be assumed that the input is a string over a given alphabet, written on an input file, which the automaton can read but not change. The input file is divided into cells, each of which can hold one symbol. The input mechanism can read the input file from left to right, one symbol at a time.


## 2 THREE BASIC CONCEPTS: Automata

An automaton is an abstract model of a digital computer. As such, every
automaton includes some essential features. It has a mechanism for reading input. It will be assumed that the input is a string over a given alphabet, written on an input file, which the automaton can read but not change. The input file is divided into cells, each of which can hold one symbol. The input mechanism can read the input file from left to right, one symbol at a time.


## Storage

## 2 THREE BASIC CONCEPTS: Automata

An automaton is an abstract model of a digital computer. As such, every automaton includes some essential features. It has a mechanism for reading input. It will be assumed that the input is a string over a given alphabet,
written on an input file, which the automaton can read but not change. The
input file is divided into cells, each of which can hold one symbol. The input
mechanism can read the input file from left to right, one symbol at a time.


## 2 THREE BASIC CONCEPTS: Automata

An automaton is an abstract model of a digital computer. As such, every automaton includes some essential features. It has a mechanism for reading input. It will be assumed that the input is a string over a given alphabet, written on an input file, which the automaton can read but not change. The input file is divided into cells, each of which can hold one symbol. The input mechanism can read the input file from left to right, one symbol at a time.

Control unit

Output


## 2 THREE BASIC CONCEPTS: Automata

An automaton is an abstract model of a digital computer. As such, every automaton includes some essential features. It has a mechanism for reading input. It will be assumed that the input is a string over a given alphabet, written on an input file, which the automaton can read but not change. The
input file is divided into cells, each of which can hold one symbol. The input
mechanism can read the input file from left to right, one symbol at a time.


## 2 THREE BASIC CONCEPTS: Automata

An automaton is an abstract model of a digital computer. As such, every automaton includes some essential features. It has a mechanism for reading input. It will be assumed that the input is a string over a given alphabet, written on an input file, which the automaton can read but not change.
mechanism can read the input file from left to right, one symbol at a time.


## 2 THREE BASIC CONCEPTS: Automata

An automaton is an abstract model of a digital computer. As such, every automaton includes some essential features. It has a mechanism for reading input. It will be assumed that the input is a string over a given alphabet, written on an input file, which the automaton can read but not change. The input file is divided into cells, each of which can hold one symbol.

Input file


Control unit

Output

## 2 THREE BASIC CONCEPTS: Automata

An automaton is an abstract model of a digital computer. As such, every automaton includes some essential features. It has a mechanism for reading input. It will be assumed that the input is a string over a given alphabet, written on an input file, which the automaton can read but not change. The input file is divided into cells, each of which can hold one symbol. The input mechanism can read the input file from left to right, one symbol at a time.

Input file


## 2 THREE BASIC CONCEPTS: Automata

The input mechanism can also detect the end of the input string (by sensing an end-of-file condition). The automaton can produce output of some form. It may have a temporary storage device, consisting of an unlimited number of cells, each capable of holding a single symbol from an alphabet (not necessarily the same one as the input alphabet).


## 2 THREE BASIC CONCEPTS: Automata

The input mechanism can also detect the end of the input string
end-of-file condition). The automaton can produce output of some form. It
may have a temporary storage device, consisting of an unlimited number of cells, each capable of holding a single symbol from an alphabet (not necessarily the same one as the input alphabet).


## 2 THREE BASIC CONCEPTS: Automata

The input mechanism can also detect the end of the input string (by sensing an end-of-file condition).
may have a temporary storage device, consisting of an unlimited number of cells, each capable of holding a single symbol from an alphabet (not necessarily the same one as the input alphabet).

Input file


## 2 THREE BASIC CONCEPTS: Automata

The input mechanism can also detect the end of the input string (by sensing an end-of-file condition). The automaton can produce output of some form.
may have a temporary storage device, consisting of an unlimited number of cells, each capable of holding a single symbol from an alphabet (not necessarily the same one as the input alphabet).

Input file


## 2 THREE BASIC CONCEPTS: Automata

The input mechanism can also detect the end of the input string (by sensing an end-of-file condition). The automaton can produce output of some form. It may have a temporary storage device,
cells, each capable of holding a single symbol from an alphabet (not necessarily the same one as the input alphabet).

Input file


## 2 THREE BASIC CONCEPTS: Automata

The input mechanism can also detect the end of the input string (by sensing an end-of-file condition). The automaton can produce output of some form. It may have a temporary storage device, consisting of an unlimited number of cells,

Input file


Storage

## 2 THREE BASIC CONCEPTS: Automata

The input mechanism can also detect the end of the input string (by sensing an end-of-file condition). The automaton can produce output of some form. It may have a temporary storage device, consisting of an unlimited number of cells, each capable of holding a single symbol from an alphabet (not necessarily


## 2 THREE BASIC CONCEPTS: Automata

The input mechanism can also detect the end of the input string (by sensing an end-of-file condition). The automaton can produce output of some form. It may have a temporary storage device, consisting of an unlimited number of cells, each capable of holding a single symbol from an alphabet (not necessarily the same one as the input alphabet).


## 2 THREE BASIC CONCEPTS: Automata

The automaton can read and change the contents of the storage cells. Finally, the automaton has a control unit, which can be in any one of a finite number of internal states, and which can change state in some defined manner. The following Figure shows a schematic representation of a general automaton.


## 2 THREE BASIC CONCEPTS: Automata

The automaton can read and change the contents of the storage cells. the automaton has a control unit, which can be in any one of a finite number of internal states, and which can change state in some defined manner. The following Figure shows a schematic representation of a general automaton.


## 2 THREE BASIC CONCEPTS: Automata

The automaton can read and change the contents of the storage cells. Finally, the automaton has a control unit, which can be in any one of a finite number of internal states, and which can change state in some defined manner. The following Figure shows a schematic representation of a general automaton.


## 2 THREE BASIC CONCEPTS: Automata

The automaton can read and change the contents of the storage cells. Finally, the automaton has a control unit, which can be in any one of a finite number of internal states, and which can change state in some defined manner. The following Figure shows a schematic representation of a general automaton.


Storage

## 2 THREE BASIC CONCEPTS: Automata

The automaton can read and change the contents of the storage cells. Finally, the automaton has a control unit, which can be in any one of a finite number of internal states, and which can change state in some defined manner.


The automaton can read and change the contents of the storage cells. Finally, the automaton has a control unit, which can be in any one of a finite number of internal states, and which can change state in some defined manner. The following Figure shows a schematic representation of a general automaton.


## 2 THREE BASIC CONCEPTS: Automata

An automaton is assumed to operate in a discrete timeframe. At any given time, the control unit is in some internal state, and the input mechanism is scanning a particular symbol on the input file. The internal state of the control unit at the next time step is determined by the next-state or transition function. This transition function gives the next state in terms of the current state, the current input symbol, and the information currently in the temporary storage. During the transition from one time interval to the next, output may be produced or the information in the temporary storage changed. The term configuration will be used to refer to a particular state of the control unit, input file, and temporary storage. The transition of the automaton from one configuration to the next will be called a move.
This general model covers all the automata we will discuss in this lectures. A finite-state control will be common to all specific cases, but differences will arise from the way in which the output can be produced and the nature of the temporary storage. As we will see, the nature of the temporary storage governs the power of different types of automata.

## 2 THREE BASIC CONCEPTS: Automata

An automaton is assumed to operate in a discrete timeframe. $\qquad$
time, the control unit is in some internal state, and the input mechanism is scanning a particular symbol on the input file. The internal state of the control unit at the next time step is determined by the next-state or transition
function. This transition function gives the next state in terms of the current state, the current input symbol, and the information currently in the temporary storage. During the transition from one time interval to the next, output may be produced or the information in the temporary storage changed. The term configuration will be used to refer to a particular state of the control unit, input file, and temporary storage. The transition of the automaton from one configuration to the next will be called a move.
This general model covers all the automata we will discuss in this lectures. A finite-state control will be common to all specific cases, but differences will arise from the way in which the output can be produced and the nature of the temporary storage. As we will see, the nature of the temporary storage governs the power of different types of automata.

An automaton is assumed to operate in a discrete timeframe. At any given time, the control unit is in some internal state, and the input mechanism is scanning a particular symbol on the input file. The internal state of the control unit at the next time step is determined by the next-state or transition
function. This transition function gives the next state in terms of the current state, the current input symbol, and the information currently in the temporary storage. During the transition from one time interval to the next, output may be produced or the information in the temporary storage changed. The term configuration will be used to refer to a particular state of the control unit, input file, and temporary storage. The transition of the automaton from one configuration to the next will be called a move.
This general model covers all the automata we will discuss in this lectures. A finite-state control will be common to all specific cases, but differences will arise from the way in which the output can be produced and the nature of the temporary storage. As we will see, the nature of the temporary storage governs the power of different types of automata.

## 2 THREE BASIC CONCEPTS: Automata

An automaton is assumed to operate in a discrete timeframe. At any given time, the control unit is in some internal state, and the input mechanism is scanning a particular symbol on the input file.

```
unit at the next time step is determined by the next-state or transition
function. This transition function gives the next state in terms of the current
state, the current input symbol, and the information currently in the temporary
storage. During the transition from one time interval to the next, output may
be produced or the information in the temporary storage changed. The term
configuration will be used to refer to a particular state of the control unit, input
file, and temporary storage. The transition of the automaton from one
configuration to the next will be called a move.
This general model covers all the automata we will discuss in this lectures. A
finite-state control will be common to all specific cases, but differences will
arise from the way in which the output can be produced and the nature of the
temporary storage. As we will see, the nature of the temporary storage governs
the power of different types of automata.
```

An automaton is assumed to operate in a discrete timeframe. At any given time, the control unit is in some internal state, and the input mechanism is scanning a particular symbol on the input file. The internal state of the control unit at the next time step is determined by the next-state or transition function.
state, the current input symbol, and the information currently in the temporary
storage. During the transition from one time interval to the next, output may be produced or the information in the temporary storage changed. The term configuration will be used to refer to a particular state of the control unit, input file, and temporary storage. The transition of the automaton from one configuration to the next will be called a move.
This general model covers all the automata we will discuss in this lectures. A finite-state control will be common to all specific cases, but differences will arise from the way in which the output can be produced and the nature of the temporary storage. As we will see, the nature of the temporary storage governs the power of different types of automata.

An automaton is assumed to operate in a discrete timeframe. At any given time, the control unit is in some internal state, and the input mechanism is scanning a particular symbol on the input file. The internal state of the control unit at the next time step is determined by the next-state or transition function. This transition function gives the next state in terms of the current state,
storage. During the transition from one time interval to the next, output may be produced or the information in the temporary storage changed. The term configuration will be used to refer to a particular state of the control unit, input file, and temporary storage. The transition of the automaton from one configuration to the next will be called a move.
This general model covers all the automata we will discuss in this lectures. A finite-state control will be common to all specific cases, but differences will arise from the way in which the output can be produced and the nature of the temporary storage. As we will see, the nature of the temporary storage governs the power of different types of automata.

An automaton is assumed to operate in a discrete timeframe. At any given time, the control unit is in some internal state, and the input mechanism is scanning a particular symbol on the input file. The internal state of the control unit at the next time step is determined by the next-state or transition function. This transition function gives the next state in terms of the current state, the current input symbol,
storage. During the transition from one time interval to the next, output may be produced or the information in the temporary storage changed. The term configuration will be used to refer to a particular state of the control unit, input file, and temporary storage. The transition of the automaton from one configuration to the next will be called a move.
This general model covers all the automata we will discuss in this lectures. A finite-state control will be common to all specific cases, but differences will arise from the way in which the output can be produced and the nature of the temporary storage. As we will see, the nature of the temporary storage governs the power of different types of automata.

An automaton is assumed to operate in a discrete timeframe. At any given time, the control unit is in some internal state, and the input mechanism is scanning a particular symbol on the input file. The internal state of the control unit at the next time step is determined by the next-state or transition function. This transition function gives the next state in terms of the current state, the current input symbol, and the information currently in the temporary storage.
be produced or the information in the temporary storage changed. The term configuration will be used to refer to a particular state of the control unit, input file, and temporary storage. The transition of the automaton from one configuration to the next will be called a move.
This general model covers all the automata we will discuss in this lectures. A finite-state control will be common to all specific cases, but differences will arise from the way in which the output can be produced and the nature of the temporary storage. As we will see, the nature of the temporary storage governs the power of different types of automata.

An automaton is assumed to operate in a discrete timeframe. At any given time, the control unit is in some internal state, and the input mechanism is scanning a particular symbol on the input file. The internal state of the control unit at the next time step is determined by the next-state or transition function. This transition function gives the next state in terms of the current state, the current input symbol, and the information currently in the temporary storage. During the transition from one time interval to the next, be produced or the information in the temporary storage changed. The term configuration will be used to refer to a particular state of the control unit, input file, and temporary storage. The transition of the automaton from one configuration to the next will be called a move.
This general model covers all the automata we will discuss in this lectures. A finite-state control will be common to all specific cases, but differences will arise from the way in which the output can be produced and the nature of the temporary storage. As we will see, the nature of the temporary storage governs the power of different types of automata.

An automaton is assumed to operate in a discrete timeframe. At any given time, the control unit is in some internal state, and the input mechanism is scanning a particular symbol on the input file. The internal state of the control unit at the next time step is determined by the next-state or transition function. This transition function gives the next state in terms of the current state, the current input symbol, and the information currently in the temporary storage. During the transition from one time interval to the next, output may be produced or the information in the temporary storage changed.
file and
file, and temporary storage. The transition of the automaton from one
configuration to the next will be called a move.
This general model covers all the automata we will discuss in this lectures. A finite-state control will be common to all specific cases, but differences will arise from the way in which the output can be produced and the nature of the temporary storage. As we will see, the nature of the temporary storage governs the power of different types of automata.

An automaton is assumed to operate in a discrete timeframe. At any given time, the control unit is in some internal state, and the input mechanism is scanning a particular symbol on the input file. The internal state of the control unit at the next time step is determined by the next-state or transition function. This transition function gives the next state in terms of the current state, the current input symbol, and the information currently in the temporary storage. During the transition from one time interval to the next, output may be produced or the information in the temporary storage changed. The term configuration will be used to refer to a particular state of the control unit, input
configuration to the next will be called a move.
This general model covers all the automata we will discuss in this lectures. A finite-state control will be common to all specific cases, but differences will arise from the way in which the output can be produced and the nature of the temporary storage. As we will see, the nature of the temporary storage governs the power of different types of automata.

An automaton is assumed to operate in a discrete timeframe. At any given time, the control unit is in some internal state, and the input mechanism is scanning a particular symbol on the input file. The internal state of the control unit at the next time step is determined by the next-state or transition function. This transition function gives the next state in terms of the current state, the current input symbol, and the information currently in the temporary storage. During the transition from one time interval to the next, output may be produced or the information in the temporary storage changed. The term configuration will be used to refer to a particular state of the control unit, input file,
configuration to the next will be called a move.
This general model covers all the automata we will discuss in this lectures. A finite-state control will be common to all specific cases, but differences will arise from the way in which the output can be produced and the nature of the temporary storage. As we will see, the nature of the temporary storage governs the power of different types of automata.

An automaton is assumed to operate in a discrete timeframe. At any given time, the control unit is in some internal state, and the input mechanism is scanning a particular symbol on the input file. The internal state of the control unit at the next time step is determined by the next-state or transition function. This transition function gives the next state in terms of the current state, the current input symbol, and the information currently in the temporary storage. During the transition from one time interval to the next, output may be produced or the information in the temporary storage changed. The term configuration will be used to refer to a particular state of the control unit, input file, and temporary storage.

[^20]An automaton is assumed to operate in a discrete timeframe. At any given time, the control unit is in some internal state, and the input mechanism is scanning a particular symbol on the input file. The internal state of the control unit at the next time step is determined by the next-state or transition function. This transition function gives the next state in terms of the current state, the current input symbol, and the information currently in the temporary storage. During the transition from one time interval to the next, output may be produced or the information in the temporary storage changed. The term configuration will be used to refer to a particular state of the control unit, input file, and temporary storage. The transition of the automaton from one configuration to the next will be called a move.
finite-state control will be common to all specific cases, but differences will arise from the way in which the output can be produced and the nature of the temporary storage. As we will see, the nature of the temporary storage governs the power of different types of automata.

An automaton is assumed to operate in a discrete timeframe. At any given time, the control unit is in some internal state, and the input mechanism is scanning a particular symbol on the input file. The internal state of the control unit at the next time step is determined by the next-state or transition function. This transition function gives the next state in terms of the current state, the current input symbol, and the information currently in the temporary storage. During the transition from one time interval to the next, output may be produced or the information in the temporary storage changed. The term configuration will be used to refer to a particular state of the control unit, input file, and temporary storage. The transition of the automaton from one configuration to the next will be called a move.
This general model covers all the automata we will discuss in this lectures.
arise from the way in which the output can be produced and the nature of the temporary storage. As we will see, the nature of the temporary storage governs the power of different types of automata.

An automaton is assumed to operate in a discrete timeframe. At any given time, the control unit is in some internal state, and the input mechanism is scanning a particular symbol on the input file. The internal state of the control unit at the next time step is determined by the next-state or transition function. This transition function gives the next state in terms of the current state, the current input symbol, and the information currently in the temporary storage. During the transition from one time interval to the next, output may be produced or the information in the temporary storage changed. The term configuration will be used to refer to a particular state of the control unit, input file, and temporary storage. The transition of the automaton from one configuration to the next will be called a move.
This general model covers all the automata we will discuss in this lectures. A finite-state control will be common to all specific cases,
> arise from the way in which the output can be produced and the nature of the temporary storage. As we will see, the nature of the temporary storage governs the power of different types of automata.

An automaton is assumed to operate in a discrete timeframe. At any given time, the control unit is in some internal state, and the input mechanism is scanning a particular symbol on the input file. The internal state of the control unit at the next time step is determined by the next-state or transition function. This transition function gives the next state in terms of the current state, the current input symbol, and the information currently in the temporary storage. During the transition from one time interval to the next, output may be produced or the information in the temporary storage changed. The term configuration will be used to refer to a particular state of the control unit, input file, and temporary storage. The transition of the automaton from one configuration to the next will be called a move.
This general model covers all the automata we will discuss in this lectures. A finite-state control will be common to all specific cases, but differences will arise from the way in which the output can be produced and the nature of the temporary storage.

An automaton is assumed to operate in a discrete timeframe. At any given time, the control unit is in some internal state, and the input mechanism is scanning a particular symbol on the input file. The internal state of the control unit at the next time step is determined by the next-state or transition function. This transition function gives the next state in terms of the current state, the current input symbol, and the information currently in the temporary storage. During the transition from one time interval to the next, output may be produced or the information in the temporary storage changed. The term configuration will be used to refer to a particular state of the control unit, input file, and temporary storage. The transition of the automaton from one configuration to the next will be called a move.
This general model covers all the automata we will discuss in this lectures. A finite-state control will be common to all specific cases, but differences will arise from the way in which the output can be produced and the nature of the temporary storage. As we will see,

An automaton is assumed to operate in a discrete timeframe. At any given time, the control unit is in some internal state, and the input mechanism is scanning a particular symbol on the input file. The internal state of the control unit at the next time step is determined by the next-state or transition function. This transition function gives the next state in terms of the current state, the current input symbol, and the information currently in the temporary storage. During the transition from one time interval to the next, output may be produced or the information in the temporary storage changed. The term configuration will be used to refer to a particular state of the control unit, input file, and temporary storage. The transition of the automaton from one configuration to the next will be called a move.
This general model covers all the automata we will discuss in this lectures. A finite-state control will be common to all specific cases, but differences will arise from the way in which the output can be produced and the nature of the temporary storage. As we will see, the nature of the temporary storage governs the power of different types of automata.

## 2 THREE BASIC CONCEPTS: Automata

For subsequent discussions, it will be necessary to distinguish between deterministic automata and nondeterministic automata. A deterministic automaton is one in which each move is uniquely determined by the current configuration. If we know the internal state, the input, and the contents of the temporary storage, we can predict the future behavior of the automaton exactly. In a nondeterministic automaton, this is not so. At each point, a nondeterministic automaton may have several possible moves, so we can only predict a set of possible actions. The relation between deterministic and nondeterministic automata of various types will play a significant role in our study.
An automaton whose output response is limited to a simple "yes" or "no" is called an accepter. Presented with an input string, an accepter either accepts the string or rejects it. A more general automaton, capable of producing strings of symbols as output, is called a transducer.

## 2 THREE BASIC CONCEPTS: Automata

For subsequent discussions, it will be necessary to distinguish between
deterministic automata and nondeterministic automata. A deterministic automaton is one in which each move is uniquely determined by the current configuration. If we know the internal state, the input, and the contents of the temporary storage, we can predict the future behavior of the automaton exactly. In a nondeterministic automaton, this is not so. At each point, a nondeterministic automaton may have several possible moves, so we can only predict a set of possible actions. The relation between deterministic and nondeterministic automata of various types will play a significant role in our study.
An autonaton whose output response is limited to a simple "yes" or "no" is called an accepter. Presented with an input string, an accepter either accepts the string or rejects it. A more general automaton, capable of producing strings of symbols as output, is called a transducer.

## 2 THREE BASIC CONCEPTS: Automata

For subsequent discussions, it will be necessary to distinguish between deterministic automata and nondeterministic automata. A deterministic automaton is one in which each move is uniquely determined by the current configuration. If we know the internal state, the input, and the contents of the temporary storage, we can predict the future behavior of the automaton exactly. In a nondeterministic automaton, this is not so. At each point, a nondeterministic automaton may have several possible moves, so we can only predict a set of possible actions. The relation between deterministic and nondeterministic automata of various types will play a significant role in our study.
An autonaton whose output response is limited to a simple "yes" or "no" is called an accepter. Presented with an input string, an accepter either accepts the string or rejects it. A more general automaton, capable of producing strings of symbols as output, is called a transducer.

## 2 THREE BASIC CONCEPTS: Automata

For subsequent discussions, it will be necessary to distinguish between deterministic automata and nondeterministic automata. A deterministic automaton is one in which each move is uniquely determined by the current configuration. If we know the internal state, the input, and the contents of the
temporary storage, we can predict the future behavior of the automaton exactly. In a nondeterministic automaton, this is not so. At each point, a nondeterministic automaton may have several possible moves, so we can only predict a set of possible actions. The relation between deterministic and nondeterministic automata of various types will play a significant role in our study.
An automaton whose output response is limited to a simple "yes" or "no" is called an accepter. Presented with an input string, an accepter either accepts the string or rejects it. A more general automaton, capable of producing strings of symbols as output, is called a transducer.

## 2 THREE BASIC CONCEPTS: Automata

For subsequent discussions, it will be necessary to distinguish between deterministic automata and nondeterministic automata. A deterministic automaton is one in which each move is uniquely determined by the current configuration. If we know the internal state, the input, and the contents of the temporary storage,
exactly. In a nondeterministic automaton, this is not so. At each point, a
nondeterministic automaton may have several possible moves, so we can only predict a set of possible actions. The relation between deterministic and nondeterministic automata of various types will play a significant role in our study.
An autonaton whose output response is limited to a simple "yes" or "no" is called an accepter. Presented with an input string, an accepter either accepts the string or rejects it. A more general automaton, capable of producing strings of symbols as output, is called a transducer.

For subsequent discussions, it will be necessary to distinguish between deterministic automata and nondeterministic automata. A deterministic automaton is one in which each move is uniquely determined by the current configuration. If we know the internal state, the input, and the contents of the temporary storage, we can predict the future behavior of the automaton exactly.
nondeterministic automaton may have several possible moves, so we can only predict a set of possible actions. The relation between deterministic and nondeterministic automata of various types will play a significant role in our study.

An automaton whose output response is limited to a simple "yes" or "no" is called an accepter. Presented with an input string, an accepter either accepts the string or rejects it. A more general automaton, capable of producing strings of symbols as output, is called a transducer.

For subsequent discussions, it will be necessary to distinguish between deterministic automata and nondeterministic automata. A deterministic automaton is one in which each move is uniquely determined by the current configuration. If we know the internal state, the input, and the contents of the temporary storage, we can predict the future behavior of the automaton exactly. In a nondeterministic automaton, this is not so.
nondeterministic automaton may have several possible moves, so we can only predict a set of possible actions. The relation between deterministic and nondeterministic automata of various types will play a significant role in our study.

An automaton whose output response is limited to a simple "yes" or "no" is called an accepter. Presented with an input string, an accepter either accepts the string or rejects it. A more general automaton, capable of producing strings of symbols as output, is called a transducer.

For subsequent discussions, it will be necessary to distinguish between deterministic automata and nondeterministic automata. A deterministic automaton is one in which each move is uniquely determined by the current configuration. If we know the internal state, the input, and the contents of the temporary storage, we can predict the future behavior of the automaton exactly. In a nondeterministic automaton, this is not so. At each point, a nondeterministic automaton may have several possible moves,
predict a set of possible actions. The relation between deterministic and
nondeterministic automata of various types will play a significant role in our
study.
An automaton whose output response is limited to a simple "yes" or "no" is
called an accepter. Presented with an input string, an accepter either accepts the string or rejects it. A more general automaton, capable of producing strings of symbols as output, is called a transducer.

For subsequent discussions, it will be necessary to distinguish between deterministic automata and nondeterministic automata. A deterministic automaton is one in which each move is uniquely determined by the current configuration. If we know the internal state, the input, and the contents of the temporary storage, we can predict the future behavior of the automaton exactly. In a nondeterministic automaton, this is not so. At each point, a nondeterministic automaton may have several possible moves, so we can only predict a set of possible actions.
nondeterministic automata of various types will play a significant role in our
study.
An automaton whose output response is limited to a simple "yes" or "no" is called an accepter. Presented with an input string, an accepter either accepts the string or rejects it. A more general automaton, capable of producing strings of symbols as output, is called a transducer.

For subsequent discussions, it will be necessary to distinguish between deterministic automata and nondeterministic automata. A deterministic automaton is one in which each move is uniquely determined by the current configuration. If we know the internal state, the input, and the contents of the temporary storage, we can predict the future behavior of the automaton exactly. In a nondeterministic automaton, this is not so. At each point, a nondeterministic automaton may have several possible moves, so we can only predict a set of possible actions. The relation between deterministic and nondeterministic automata of various types will play a significant role in our study.
An automaton whose output response is limited to a simple "yes" or "no" is called an accepter. Presented with an input string, an accepter either accepts the string or rejects it. A more general automaton, capable of producing strings of symbols as output, is called a transducer.

For subsequent discussions, it will be necessary to distinguish between deterministic automata and nondeterministic automata. A deterministic automaton is one in which each move is uniquely determined by the current configuration. If we know the internal state, the input, and the contents of the temporary storage, we can predict the future behavior of the automaton exactly. In a nondeterministic automaton, this is not so. At each point, a nondeterministic automaton may have several possible moves, so we can only predict a set of possible actions. The relation between deterministic and nondeterministic automata of various types will play a significant role in our study.

An automaton whose output response is limited to a simple "yes" or "no" is called an accepter.
the string or rejects it. A more general automaton, capable of producing strings
of symbols as output, is called a transducer.

For subsequent discussions, it will be necessary to distinguish between deterministic automata and nondeterministic automata. A deterministic automaton is one in which each move is uniquely determined by the current configuration. If we know the internal state, the input, and the contents of the temporary storage, we can predict the future behavior of the automaton exactly. In a nondeterministic automaton, this is not so. At each point, a nondeterministic automaton may have several possible moves, so we can only predict a set of possible actions. The relation between deterministic and nondeterministic automata of various types will play a significant role in our study.
An automaton whose output response is limited to a simple "yes" or "no" is called an accepter. Presented with an input string, an accepter either accepts the string or rejects it.

For subsequent discussions, it will be necessary to distinguish between deterministic automata and nondeterministic automata. A deterministic automaton is one in which each move is uniquely determined by the current configuration. If we know the internal state, the input, and the contents of the temporary storage, we can predict the future behavior of the automaton exactly. In a nondeterministic automaton, this is not so. At each point, a nondeterministic automaton may have several possible moves, so we can only predict a set of possible actions. The relation between deterministic and nondeterministic automata of various types will play a significant role in our study.
An automaton whose output response is limited to a simple "yes" or "no" is called an accepter. Presented with an input string, an accepter either accepts the string or rejects it. A more general automaton, capable of producing strings of symbols as output, is called a transducer.

## Thank You for attention!


[^0]:    But although this relationship is obvious, it is useful to be able to make it precise and prove it. The techniques for doing so are important in more complicated situations.

[^1]:    and if we associate the actual words＂a＂and＂the＂with 〈article〉，＂boy＂and us that the sentences＂a boy runs＂and＂the dog walks＂are properly formed．If we were to give a complete grammar，then in theory，every proper sentence could be explained this way．

[^2]:    where

    It will be assumed without further mention that the sets $V$ and $T$ are
    nonempty and disjoint.

[^3]:    where both $w_{1}$ and $w_{2}$ are in $L$. This case can be taken care of by the

[^4]:    where both $w_{1}$ and $w_{2}$ are in $L$. This case can be taken care of by the

[^5]:    where both $w_{1}$ and $w_{2}$ are in $L$. This case can be taken care of by the

[^6]:    where both $w_{1}$ and $w_{2}$ are in $L$. This case can be taken care of by the

[^7]:    where both $w_{1}$ and $w_{2}$ are in $L$. This case can be taken care of by the

[^8]:    where both $w_{1}$ and $w_{2}$ are in $L$. This case can be taken care of by the

[^9]:    where both $w_{1}$ and $w_{2}$ are in $L$. This case can be taken care of by the

[^10]:    where both $w_{1}$ and $w_{2}$ are in $L$. This case can be taken care of by the

[^11]:    where both $w_{1}$ and $w_{2}$ are in $L$. This case can be taken care of by the

[^12]:    where both $w_{1}$ and $w_{2}$ are in $L$. This case can be taken care of by the

[^13]:    where both $w_{1}$ and $w_{2}$ are in $L$. This case can be taken care of by the

[^14]:    where both $w_{1}$ and $w_{2}$ are in $L$. This case can be taken care of by the

[^15]:    where both $w_{1}$ and $w_{2}$ are in $L$. This case can be taken care of by the

[^16]:    where both $w_{1}$ and $w_{2}$ are in $L$. This case can be taken care of by the

[^17]:    where both $w_{1}$ and $w_{2}$ are in $L$. This case can be taken care of by the

[^18]:    We leave this as an exercise.

[^19]:    We leave this as an exercise.

[^20]:    This general model covers all the automata we will discuss in this lectures. A finite-state control will be common to all specific cases, but differences will arise from the way in which the output can be produced and the nature of the temporary storage. As we will see, the nature of the temporary storage governs the power of different types of automata.

