

# Recursive Definitions

1 2 3 4 5 6 7 8 9 10 11



# Recursive Definitions

## Introduction

### Three step process

1. Define one or more elements to be in the set
2. Specify a rule to generate one or more elements in the set from one or more elements that are already in the set.
3. Specify that all objects not generated by the above rules are not in the set.

The third step above is usually assumed rather than stated explicitly.

1 2 3 4 5 6 7 8 9 10 11

# Recursive Definitions

## Introduction

**Recursive definitions** are ways to define sets based on a collection of rules such that one or more of the rules specifies what elements are in the set based on elements already in the set.

1 2 3 4 5 6 7 8 9 10 11

# Recursive Definitions

## Introduction

**Example.** Consider how we define the factorial operation:

**Rule 1:**  $1! = 1$

**Rule 2:**  $n! = n(n - 1)!$       $n$  is an integer greater than 1

This is a familiar example of a recursive definition. (In addition to the rules above, we usually also define  $0! = 1$ .)

1 2 3 4 5 6 7 8 9 10 11

## Recursive Definitions

[prev](#) | [slides](#) | [next](#)

### Introduction

**Example.** Let  $\Sigma = \{ x \}$ . We can define a language  $L$  as follows:

**Rule 1:**  $\Lambda$  is in  $L$

**Rule 2:** If  $w$  is a word in  $L$  then  $xw$  is also in  $L$ .

This generates the language

$$L = \{ \Lambda x xx xxx \dots \}$$

so that  $L = L^*$ .

[1 2 3 4 5 6 7 8 9 10 11](#)

## Recursive Definitions

[prev](#) | [slides](#) | [next](#)

### Arithmetic Expressions (an extended example)

Let  $\Sigma$  be the set of digits, arithmetic operators, and parentheses:

$$\Sigma = \{ 0 1 2 3 4 5 6 7 8 9 + - * / ( ) \}$$

and let the language  $AE$  be defined with

**Rule 1:** Any string of digits is in  $AE$

**Rule 2:** If  $x$  is a word in  $AE$  then so are  $(x)$  and  $-x$

**Rule 3:** If  $x$  and  $y$  are words in  $AE$  then so are

1.  $x + y$  (if  $y$  does not begin with a  $-$  sign)
2.  $x - y$  (if  $y$  does not begin with a  $-$  sign)
3.  $x * y$
4.  $x / y$
5.  $x ** y$  (exponentiation)

## Recursive Definitions

[prev](#) | [slides](#) | [next](#)

### Introduction

**Example.** Let  $\Sigma = \{ 0 1 \}$ . We can define a language  $L$  as follows:

**Rule 1:**  $0$  is in  $L$

**Rule 2:** If  $w$  is a word in  $L$  then  $w0$  is also in  $L$ .

**Rule 3:** If  $w$  is a word in  $L$  then  $1w$  is also in  $L$ .

This generates the language

$$L = \{ 0 00 10 000 100 110 0000 1000 1100 1110 00000 10000 \\ 11000 11100 11110 \dots \}$$

[1 2 3 4 5 6 7 8 9 10 11](#)

[1 2 3 4 5 6 7 8 9 10 11](#)

## Recursive Definitions

[prev](#) | [slides](#) | [next](#)

### Arithmetic Expressions (an extended example)

Note:

1. Strings such as " $2 * 3 + 5$ " are in  $AE$  but the definition for  $AE$  says nothing about how this string is to be interpreted -- the *meaning* of the arithmetic expression is not given by the definition.
2. According to this definition  $\Lambda$  is not in  $AE$ . If it were then Rule 3 would assert that  $\Lambda + \Lambda$ , which is just  $+$ , would be in  $AE$ .

[1](#) [2](#) [3](#) [4](#) [5](#) [6](#) [7](#) [8](#) [9](#) [10](#) [11](#)

## Recursive Definitions

[prev](#) | [slides](#) | [next](#)

### Arithmetic Expressions (an extended example)

**Theorem:** No arithmetic expression can begin or end with the symbol  $/$ .

**Proof:** Since no number begins or ends with  $/$  this symbol cannot enter  $AE$  via Rule 1. Rule 2 does not even mention this symbol so  $/$  cannot enter via Rule 2. According to Rule 3,  $/$  can only enter between two members of  $AE$ , so that the resulting member of  $AE$  does not begin or end with  $/$ .

[1](#) [2](#) [3](#) [4](#) [5](#) [6](#) [7](#) [8](#) [9](#) [10](#) [11](#)

## Recursive Definitions

[prev](#) | [slides](#) | [next](#)

### Arithmetic Expressions (an extended example)

**Theorem:** An arithmetic expression cannot contain the character  $\$$ .

**Proof:** According to Rule 1, every string of digits is in  $AE$ . Actually, according to Rule 1 **only** strings of digits are in  $AE$ ; however the other rules specify additional members. Rules 2 and 3 take elements that already in  $AE$  and combine them with symbols from  $\Sigma$ . Since it is not possible for a  $\$$  to enter  $AE$  from Rule 1, and neither Rule 2 or Rule 3 can bring in a  $\$$ , it is not possible for  $\$$  to be in  $AE$ .

[1](#) [2](#) [3](#) [4](#) [5](#) [6](#) [7](#) [8](#) [9](#) [10](#) [11](#)

## Recursive Definitions

[prev](#) | [slides](#) | [next](#)

### Arithmetic Expressions (an extended example)

**Theorem:** No arithmetic expression can contain  $//$ .

**Proof:** See text, page 27. A proof by contradiction is developed by assuming that there is a string in  $AE$  that contains  $//$  and further that we are looking at an example of such a string that is as short as any other. The proof continues by showing that a substring with  $//$  exists as a member of  $AE$ , contradicting our assumption.

[1](#) [2](#) [3](#) [4](#) [5](#) [6](#) [7](#) [8](#) [9](#) [10](#) [11](#)