Ch 5.3: Recursive Definitions

ICS 141: Discrete Mathematics for Computer Science I

Kyle Berney
Department of ICS, University of Hawaii at Manoa

Recursive Definitions

- Recall from Chapter 2.4, that we can define sequences using a recurrence relation:
  1. Provide one or more initial terms of the sequence
  2. Rule for determining subsequent terms from terms that preceed it
- Generalize this approach to recursively define functions, sets, and other structures

Recursively Defined Functions

- Recursively define a function with the set of non-negative integers as its domain:
  1. Basis Step:
     - Specify the value of the function at 0
  2. Recursive Step:
     - Give a rule for finding its value at an integer from its values at smaller integers

Recursively Defined Functions

- Ex:
  - Basis Step: $f(0) = 3$
  - Recursive Step: $f(n+1) = 2f(n) + 3$

$$f(1) = 2f(0) + 3 = 2 \cdot 3 + 3 = 9$$
$$f(2) = 2f(1) + 3 = 2 \cdot 9 + 3 = 21$$
$$f(3) = 2f(2) + 3 = 2 \cdot 21 + 3 = 45$$
$$f(4) = 2f(3) + 3 = 2 \cdot 45 + 3 = 93$$
$$\vdots$$

Recursively Defined Sets

- Recursively define a set:

1. Basis Step:
   - Specify an initial collection of elements
2. Recursive Step:
   - Give a rule for forming new elements in the set from those already known to be in the set

Recursively Defined Sets

- Ex: Let S be the set of all positive multiples of 3
  - Basis Step: $3 \in S$
  - Recursive Step: If $x \in S$ and $y \in S$ then $x + y \in S$

- Apply recursive step:

1. $3 + 3 = 6$
   $\Rightarrow \{3, 6\}$
2. $3 + 6 = 9$ and $6 + 6 = 12$
   $\Rightarrow \{3, 6, 9, 12\}$
3. $9 + 3 = 12$, $9 + 6 = 12 + 3 = 15$, $9 + 9 = 12 + 6 = 18$, $12 + 9 = 21$, and $12 + 12 = 24$
   $\Rightarrow \{3, 6, 9, 12, 15, 18, 21, 24\}$

$\vdots$

Recursively Defined Sets

- The set Σ^* of strings over the alphabet Σ is defined as
  - Basis Step: $\lambda \in \Sigma^*$ (recall that λ is the empty string)
  - Recursive Step: If $w \in \Sigma^*$ and $x \in \Sigma$ then $wx \in \Sigma^*$

Recursively Defined Sets

- The set Σ^* of strings over the alphabet Σ is defined as
  - Basis Step: $\lambda \in \Sigma^*$ (recall that λ is the empty string)
  - Recursive Step: If $w \in \Sigma^*$ and $x \in \Sigma$ then $wx \in \Sigma^*$

- Ex: Let $\Sigma = \{0, 1\}$

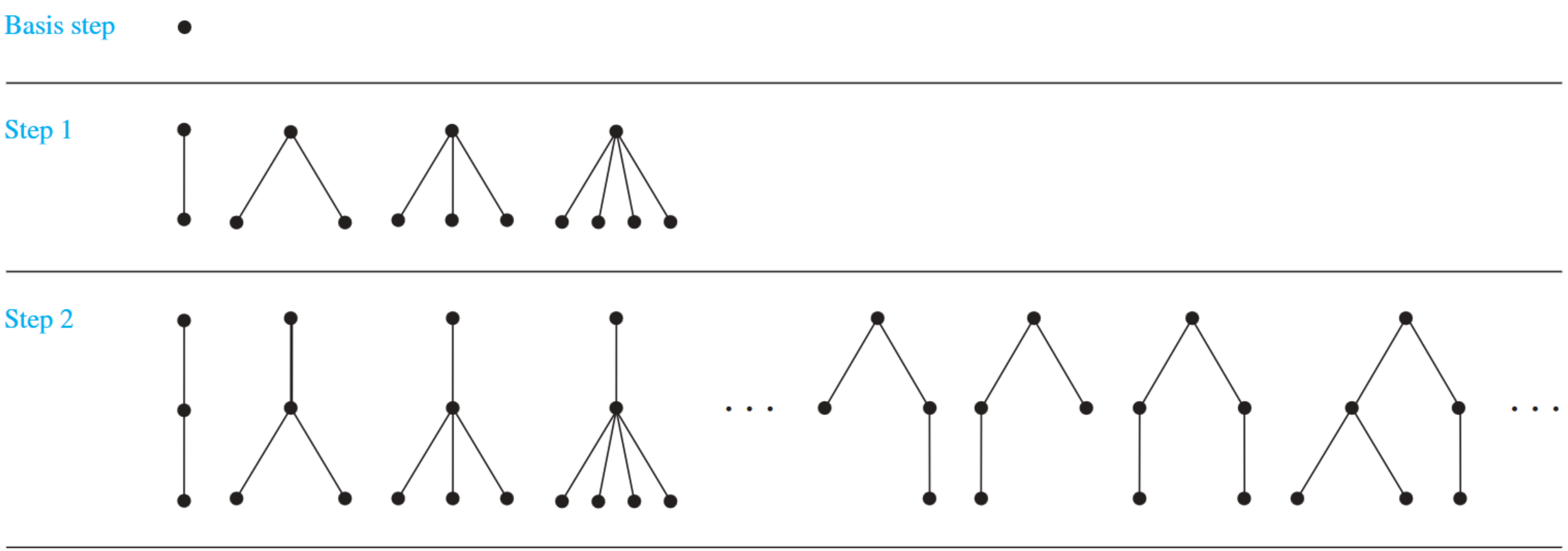
1. $\Sigma^* = \{0, 1\}$
2. $\Sigma^* = \{0, 1, 00, 01, 10, 11\}$
3. $\Sigma^* =$ $\{0, 1, 00, 01, 10, 11, 000, 001, 010, 011, 100, 101, 110, 111\}$

$\vdots$

Recursively Defined Rooted Trees

- Formally discuss rooted trees in Chapter 10 and 11 (ICS 241)
- Recursively define a rooted tree:

1. Basis Step: A single vertex is a rooted tree.
2. Recursive Step: Given n rooted trees $T_1, T_2, \ldots, T_n$ we can form another rooted tree starting with a single root and connecting the root to all other n rooted trees.



Recursively Defined Binary Trees

- Binary trees are rooted trees where every vertex has at most two children (i.e., connected to at most two other vertices)
- Recursively define a binary tree:

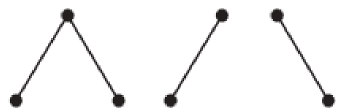
1. Basis Step: The empty set is a binary tree
2. Recursive Step: Given 2 binary trees T_1 and T_2 we can form another binary tree starting with a single root and connecting the root to T_1 and T_2

Recursively Defined Binary Trees

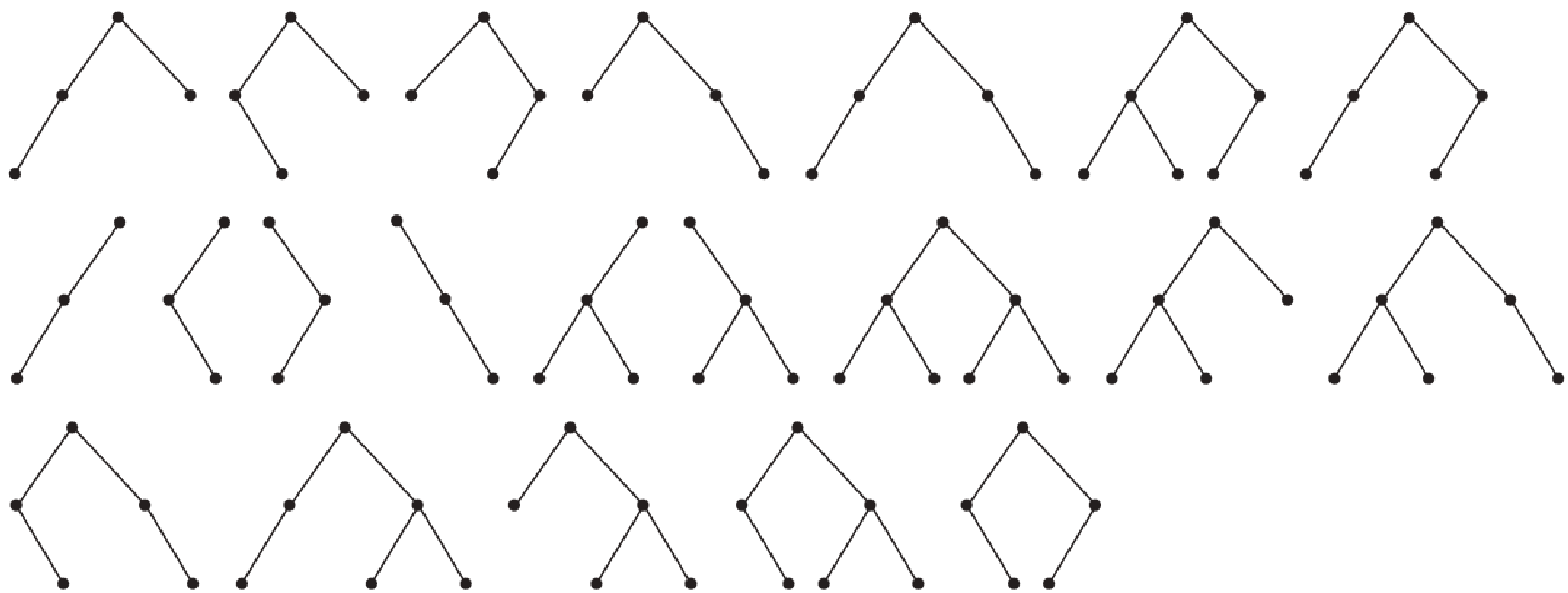
Basis step $\emptyset$

Step 1

Step 2



Step 3



Recursively Defined Full Binary Trees

- Full binary trees are binary trees where every vertex has either 0 or two children (i.e., connected to either 0 or two other vertices)
- Recursively define a full binary tree:

1. Basis Step: A single vertex is a full binary tee
2. Recursive Step: Given 2 full binary trees T_1 and T_2 we can form another binary tree starting with a single root and connecting the root to T_1 and T_2

Recursively Defined Full Binary Trees