Recursively Defined Sets and Structures

Recursive definitions of sets have two parts:

- The *basis step* specifies an initial collection of elements.
- The *recursive step* gives the rules for forming new elements in the set from those already known to be in the set.

Recursively Defined Sets and Structures

- Sometimes the recursive definition has an *exclusion rule*, which specifies that the set contains nothing other than those elements specified in the basis step and generated by applications of the rules in the recursive step – this is usually indicated by the phrase: *the smallest set generated by these rules.*
- We will always assume that the exclusion rule holds, even if it is not explicitly mentioned.
- We will develop a form of induction, called *structural induction*, to prove results about recursively defined sets.

Recursively Defined Sets and Structures

Example: The natural numbers **N**.

BASIS STEP: $0 \in \mathbf{N}$.

RECURSIVE STEP: If n is in **N**, then $n + 1$ is in **N**.

- Initially 0 is in S, then $0 + 1 = 1$, then $1 + 1 = 2$, etc.

- Note: the natural numbers are a recursively defined set, later on we'll see that mathematical induction is just a special case of structural induction.

Recursively Defined Sets and Structures

Example : Subset of Integers S:

BASIS STEP: $3 \in S$.

RECURSIVE STEP: If $x \in S$ and $y \in S$, then $x + y$ is in S.

- Initially 3 is in S, then $3 + 3 = 6$, then $3 + 6 = 9$, etc.

Strings

Definition: The set Σ^* of *strings* over the alphabet Σ:

BASIS STEP: $\lambda \in \Sigma^*$ (λ is the empty string)

RECURSIVE STEP: If w is in Σ^* and x is in Σ,
then $wx \in \Sigma^*$.

Strings

Example: If $\Sigma = \{0,1\}$, the strings in in Σ^* are the set of all bit strings, λ,0,1, 00,01,10, 11, etc.

Strings

Example: If $\Sigma = \{a,b\}$, show that aab is in Σ^*.

- Since $\lambda \in \Sigma^*$ and $a \in \Sigma$, $a \in \Sigma^*$.
- Since $a \in \Sigma^*$ and $a \in \Sigma$, $aa \in \Sigma^*$.
- Since $aa \in \Sigma^*$ and $b \in \Sigma$, $aab \in \Sigma^*$.

String Length Operation

Definition: We can define a string length operator, say $l:\Sigma^* \rightarrow N$, recursively,

BASIS STEP: $l(\lambda)= 0$ with $\lambda \in \Sigma^*$.

RECURSIVE STEP: If $w \in \Sigma^*$ and $x \in \Sigma$, then

$$l(wx)=l(w)+1.$$

Balanced Parentheses

Example: Give a recursive definition of the set of balanced parentheses P.

Solution:

BASIS STEP: $() \in P$

RECURSIVE STEP: If $w \in P$, then $()w \in P$, $(w) \in P$ and $w() \in P$.

- Show that $((())())$ is in P.
- Why is $))((()$ not in P?

Well-Formed Formulae in Propositional Logic

Definition: The set of *well-formed formulae* (WFF) in propositional logic involving **T**, **F**, propositional variables, and operators from the set $\{\neg,\wedge,\vee,\rightarrow,\leftrightarrow\}$.

BASIS STEP: **T**,**F**, and V, where V is the set of propositional variables, are well-formed formulae. We write **T**,**F**,$\in$WFF and V$\subset$WFF.

RECURSIVE STEP: If $E,F\in$WFF, then $(\neg E)$, $(E \wedge F)$, $(E \vee F)$, $(E \rightarrow F)$, $(E \leftrightarrow F) \in$WFF.

Well-Formed Formulae in Propositional Logic

Examples:

$p \in V$ is a WFF

T is a WFF

$((p \vee q) \rightarrow (q \wedge \mathbf{F}))$ is a WFF

$pq \wedge$ is <u>not</u> a WFF.

Note: in general, the syntax of formal languages can be defined inductively! This include programming languages.

The Language of Algebraic Expressions

- We can define the set LA of algebraic expressions recursively:
  - BASIS: any number and any variable name is in LA.
  - RECURSIVE STEP: let A,B∈LA, then
    A+B, A-B, A/B, A*B, (A) ∈ LA

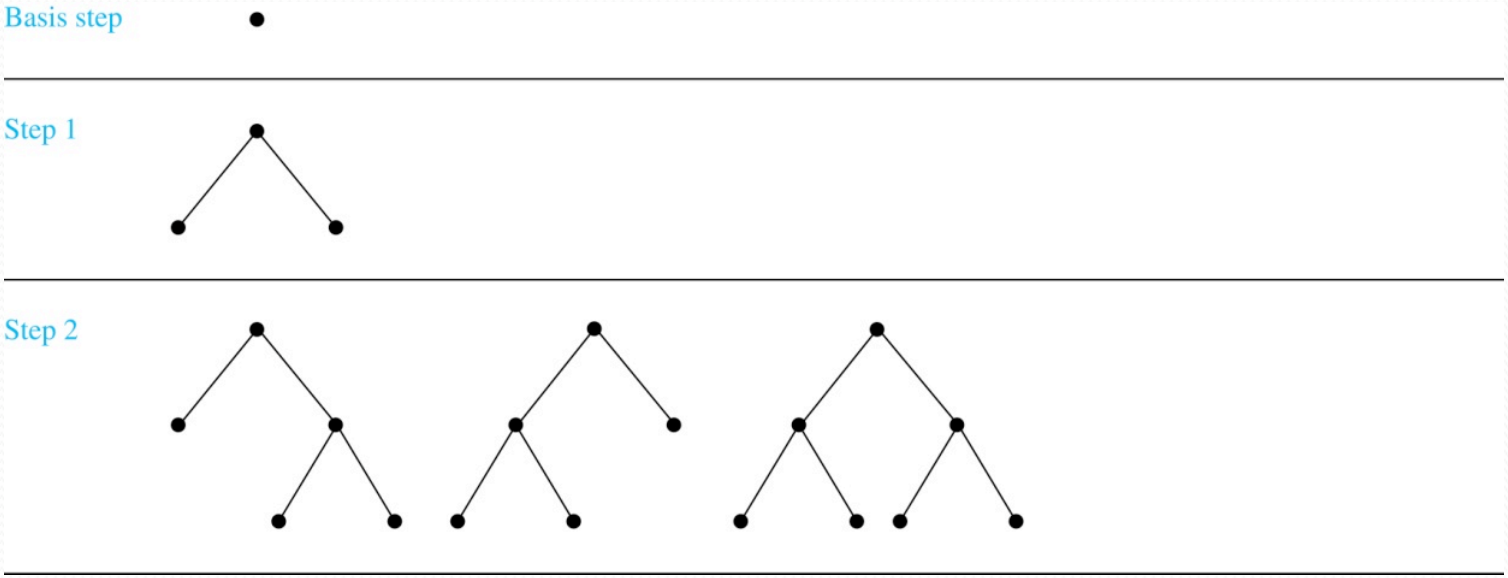
Full Binary Trees

Definition: The set of *full binary trees* can be defined recursively by these steps.

BASIS STEP: There is a full binary tree consisting of only a single vertex r.

RECURSIVE STEP: If T_1 and T_2 are disjoint full binary trees, there is a full binary tree consisting of a root r together with edges connecting the root to each of the roots of the left subtree T_1 and the right subtree T_2.

Building Up Full Binary Trees



Structural Induction

Definition: To prove a property of the elements of a recursively defined set, we use *structural induction*.

BASIS STEP: Show that the result holds for all elements specified in the basis step of the recursive definition.

INDUCTIVE STEP: Assume that the property holds for the elements currently in the recursively defined set. Show that it is true for each of the rules used to construct new elements in the recursive step of the definition.

Example

- Observe, if n is divisible by 3, then we can write $n=3k$ for some integer k.
- Now, show that every element in the set S defined recursively as
  - basis: $3 \in S$
  - recursive step: if $p \in S$ then $p+3 \in S$

  is divisible by 3.
- Cannot do it by exhaustively listing elements, infinite set.
- Proof by structural induction.

Example

- Proof by structural induction
  - basis: 3 is divisible by 3, therefore basis case holds.
  - inductive step: Assume q∈S, that is, q=3k for some integer k. We now show that q+3∈S. Using our inductive hypothesis we have

$$q + 3 = 3k + 3$$

$$= 3(k + 1)$$

Which shows that q+3 is divisible by 3. (QED)

Another Example

Definition: The *height* $h(T)$ of a full binary tree T is defined recursively as follows:

- BASIS STEP: The height of a full binary tree T consisting of only a root r is $h(T) = 0$.
- RECURSIVE STEP: If T_1 and T_2 are full binary trees, then the full binary tree $T = T_1 \cdot T_2$ has height $h(T) = 1 + \max(h(T_1), h(T_2))$.

Another Example

- **Definition**: The number of vertices $n(T)$ of a full binary tree T satisfies the following recursive formula:
  - BASIS STEP: The number of vertices of a full binary tree T consisting of only a root r is $n(T) = 1$.
  - RECURSIVE STEP: If T_1 and T_2 are full binary trees, then the full binary tree $T = T_1 \cdot T_2$ has the number of vertices

    $n(T) = 1 + n(T_1) + n(T_2)$.

Another Example

Theorem: If T is a full binary tree, then $n(T) \leq 2^{h(T)+1} - 1$.

Proof: Use structural induction.

- BASIS STEP: The result holds for a full binary tree consisting only of a root, $n(T) = 1$ and $h(T) = 0$. Hence, $n(T) = 1 \leq 2^{0+1} - 1 = 1$.
- RECURSIVE STEP: Assume $n(T_1) \leq 2^{h(T1)+1} - 1$ and also $n(T_2) \leq 2^{h(T2)+1} - 1$ whenever T_1 and T_2 are full binary trees.

$$
\begin{aligned}
n(T) &= 1 + n(T_1) + n(T_2) && (\textit{by recursive formula of } n(T)) \\
&\leq 1 + (2^{h(T_1)+1} - 1) + (2^{h(T_2)+1} - 1) && (\textit{by inductive hypothesis}) \\
&\leq 2 \cdot \max(2^{h(T_1)+1}, 2^{h(T_2)+1}) - 1 \\
&= 2 \cdot 2^{\max(h(T_1),h(T_2))+1} - 1 && (\max(2^x, 2^y) = 2^{\max(x,y)}) \\
&= 2 \cdot 2^{h(t)} - 1 && (\textit{by recursive definition of } h(T)) \\
&= 2^{h(t)+1} - 1
\end{aligned}
$$