CSCI 2200 FOUNDATIONS OF COMPUTER SCIENCE

David Goldschmidt
goldsd3@rpi.edu
Fall 2022

PROVING AN UNFOLDED RECURRENCE – EXAMPLE

Given $A_1 = 1$ and $A_n = 10A_{n-1} + 1$ for $n > 1$

If unfolding does not work, try tinkering with a few values to observe a pattern…

$$A_2 = 10A_1 + 1 = 10(1) + 1 = 11$$

$$A_3 = 10A_2 + 1 = 10(11) + 1 = 111$$

$$A_4 = 10A_3 + 1 = 10(111) + 1 = 1111$$

$$A_5 = 10A_4 + 1 = 10(1111) + 1 = 11111$$

Aha, observe that $A_n = 1 + 10 + 100 + 1000 + \ldots + 10^{n-1}$

…how can we write this more precisely?

$$A_n = \left\lfloor \frac{10^n}{9} \right\rfloor$$

PROVING AN UNFOLDED RECURRENCE — EXAMPLE

Given $A_1 = 1$ and $A_n = 10A_{n-1} + 1$ for $n > 1$ — our claim is that $A_n = \left\lfloor \frac{10^n}{9} \right\rfloor$

Proof. We use induction to prove the claim for $n \geq 1$.

1. **[Base case]** For $n = 1$, we have $A_1 = \left\lfloor \frac{10^1}{9} \right\rfloor = 1$, which is **T**.
2. **[Induction step]** Assume $A_n = \left\lfloor \frac{10^n}{9} \right\rfloor$; we must prove $A_{n+1} = \left\lfloor \frac{10^{n+1}}{9} \right\rfloor$.

   From the LHS, $A_{n+1} = 10A_n + 1$.

   Applying our induction hypothesis, we have $A_{n+1} = 10\left\lfloor \frac{10^n}{9} \right\rfloor + 1$.

Given $A_1 = 1$ and $A_n = 10A_{n-1} + 1$ for $n > 1$ — our claim is that $A_n = \left\lfloor \frac{10^n}{9} \right\rfloor$

Proof. We use induction to prove the claim for $n \geq 1$.

1. **[Base case]** For $n = 1$, we have $A_1 = \left\lfloor \frac{10^1}{9} \right\rfloor = 1$, which is **T**.
2. **[Induction step]** Assume $A_n = \left\lfloor \frac{10^n}{9} \right\rfloor$; we must prove $A_{n+1} = \left\lfloor \frac{10^{n+1}}{9} \right\rfloor$.

   From the LHS, $A_{n+1} = 10A_n + 1$.

   Applying our induction hypothesis, we have $A_{n+1} = 10\left\lfloor \frac{10^n}{9} \right\rfloor + 1$.

See Problem 7.12(a)…

   Rearranging, we obtain $A_{n+1} = 10\left(\left\lfloor \frac{10^n}{9} \right\rfloor + \frac{1}{10}\right)$.

Adding 0.1 and multiplying by 10 is equivalent to the floor here…

   And from this, we have $A_{n+1} = \left\lfloor \frac{10^{n+1}}{9} \right\rfloor$, as was to be shown. ■

Strings in M are *balanced*, meaning the number of opening and closing parentheses is equal…

RECURSIVE SETS – MATCHED PARENTHESES

Recursive definition of matched parentheses set M.

1. $\varepsilon \in M$ **[basis]**
2. $x, y \in M \rightarrow [x] \bullet y \in M$ **[constructor]**

Write *derivations* for [ ], [[ ]], and [ ] [ ] by showing each step taken from the base case…

$M = \{ \varepsilon, [\,], [\,[\,]\,], [\,]\,[\,], [\,[\,]\,]\,[\,], \dots \}$

$\varepsilon \rightarrow [\,]$ set $x = \varepsilon$ and $y = \varepsilon$ to get $[\,\varepsilon\,]\,\varepsilon = [\,]$

$\varepsilon \rightarrow [\,] \rightarrow [\,[\,]\,]$ set $x = [\,]$ and $y = \varepsilon$ to get $[\,[\,]\,]\,\varepsilon = [\,[\,]\,]$

$\varepsilon \rightarrow [\,] \rightarrow [\,]\,[\,]$ set $x = \varepsilon$ and $y = [\,]$ to get $[\,\varepsilon\,]\,[\,] = [\,]\,[\,]$

We *derive* element s_n by applying the constructor to two prior strings…

…and the two strings need not be distinct!

PROPERTIES OF RECURSIVE SETS

Recursive definition of the natural numbers $\mathbb{N}$.

1. $1 \in \mathbb{N}$. **[basis]**
2. $x \in \mathbb{N} \rightarrow x + 1 \in \mathbb{N}$. **[constructor]**

e.g., show $P(n)$: $5^n - 1$ is divisible by 4 for all natural numbers…

Consider any property $P(n)$ that we want to prove for all $n \in \mathbb{N}$.

Proof. We use structural induction to prove that property $P(n)$ holds for all $n \in \mathbb{N}$.

1. **[Base cases]** Prove $P(n)$ holds for all base cases.
2. **[Induction step]** Prove that each constructor rule preserves $P(n)$:

   IF $P(n)$ is **T** for parent element x, THEN $P(n)$ is **T** for all child element(s)

3. By structural induction, we have shown that $P(n)$ is **T** for all $n \in \mathbb{N}$. ■

RECURSIVE SETS – BINARY STRINGS OF ODD LENGTH

Recursive definition of set S_{odd} (binary strings of odd length).

1. $0 \in S_{odd}$; $1 \in S_{odd}$ **[basis]**
2. $x \in S_{odd} \rightarrow 0 \bullet x \bullet 0 \in S_{odd}$ **[constructor (i)]**
   $x \in S_{odd} \rightarrow 0 \bullet x \bullet 1 \in S_{odd}$ **[constructor (ii)]**
   $x \in S_{odd} \rightarrow 1 \bullet x \bullet 0 \in S_{odd}$ **[constructor (iii)]**
   $x \in S_{odd} \rightarrow 1 \bullet x \bullet 1 \in S_{odd}$ **[constructor (iv)]**

Prove by structural induction that every element in S_{odd} is of odd length…

(next slide)

Prove (often by contradiction) that every binary string of odd length is in S_{odd}…

(next next slide)

STRUCTURAL INDUCTION

Recursive definition of set S_{odd}.

1. $0 \in S_{odd}$; $1 \in S_{odd}$ **[basis]**
2. $x \in S_{odd} \rightarrow 0 \bullet x \bullet 0 \in S_{odd}$ **[constructor (i)]**
   $x \in S_{odd} \rightarrow 0 \bullet x \bullet 1 \in S_{odd}$ **[constructor (ii)]**
   $x \in S_{odd} \rightarrow 1 \bullet x \bullet 0 \in S_{odd}$ **[constructor (iii)]**
   $x \in S_{odd} \rightarrow 1 \bullet x \bullet 1 \in S_{odd}$ **[constructor (iv)]**

Proof. We use structural induction to prove that every element of S_{odd} has odd length.

1. **[Base cases]** Both strings 0 and 1 have odd length.
2. **[Induction step]** We must prove that each constructor preserves *oddness*.

   Assume $x \in S_{odd}$ has odd length, meaning $|x| = 2k + 1$ for $k \in \mathbb{N}_0$.

   For constructor (i), we must show $0 \bullet x \bullet 0$ has odd length.

   Here, $0 \bullet x \bullet 0$ increases the length by 2, so $|0x0| = 2k + 3$, which must be odd.

   For constructors (ii), (iii), and (iv), we repeat the above.
3. By structural induction, we conclude that every member of S_{odd} has odd length. ■

CONTRADICTION

Recursive definition of set S_{odd}.

1. $0 \in S_{odd}$; $1 \in S_{odd}$ **[basis]**
2. $x \in S_{odd} \rightarrow 0 \bullet x \bullet 0 \in S_{odd}$ **[constructor (i)]**
   $x \in S_{odd} \rightarrow 0 \bullet x \bullet 1 \in S_{odd}$ **[constructor (ii)]**
   $x \in S_{odd} \rightarrow 1 \bullet x \bullet 0 \in S_{odd}$ **[constructor (iii)]**
   $x \in S_{odd} \rightarrow 1 \bullet x \bullet 1 \in S_{odd}$ **[constructor (iv)]**

Proof. We use contradiction to prove that every binary string of odd length is in S_{odd}.

Consider string s, the shortest string of odd length <u>not</u> in S_{odd}.
Then $|s| \geq 3$ or else s would be a base case in S_{odd}. We have two cases.

Case 1. If s starts with 0, we can define $s = 0 \bullet x \bullet 0$ or $s = 0 \bullet x \bullet 1$.
Then, x must have odd length $|x| \geq 1$, meaning that $x \in S_{odd}$.

But if $x \in S_{odd}$, then by constructor (i) or (ii), we have $s \in S_{odd}$, a contradiction!

Case . If s starts with 1, we can define $s = 1 \bullet x \bullet 0$ or $s = 1 \bullet x \bullet 1$.
Then, x must have odd length $|x| \geq 1$, meaning that $x \in S_{odd}$.

But if $x \in S_{odd}$, then by constructor (iii) or (iv), we have $s \in S_{odd}$, a contradiction!

Given both contradictions, we conclude that there is no shortest string of odd length that is not in S_{odd}, i.e., every binary string of odd length is in S_{odd}. ■

ROOTED BINARY TREE

Write a recursive definition that generates RBTs of arbitrary height…

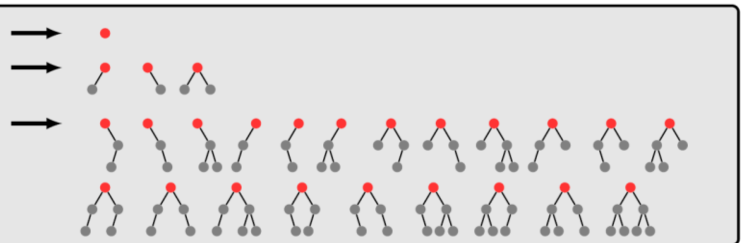
A *rooted binary tree* (*RBT*) is a graph with $|V| \geq 0$ vertices and the following properties:

1. One vertex is identified as the *root* of the tree (indicated in red)
2. There is exactly one path from the root to any other vertex in the tree

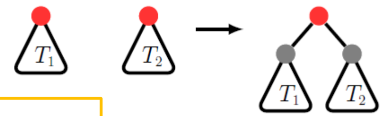
   (Here, a *path* is defined as a sequence of edge traversals…)

3. **Each vertex has at most two children!**

RBTs of height 0, 1, and 2

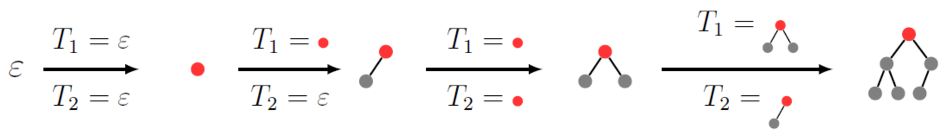


ROOTED BINARY TREE



Recursive definition of Rooted Binary Tree (RBT).

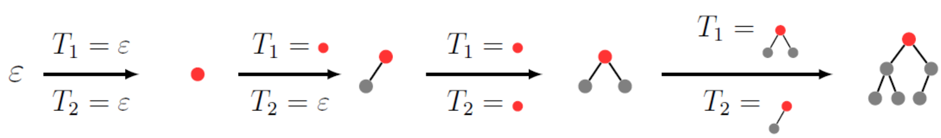
1. The empty tree ε is an RBT (no root). **[basis]**
2. If T_1 and T_2 are disjoint RBTs with roots r_1 and r_2, then linking r_1 and r_2 to a new root r produces a new RBT with root r. **[constructor]**



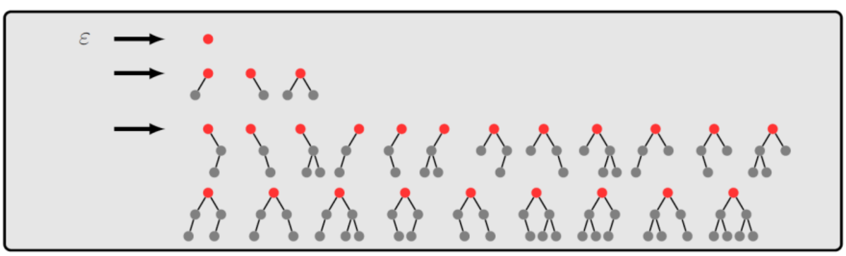
ROOTED BINARY TREE

Recursive definition of Rooted Binary Tree (RBT).

1. The empty tree ε is an RBT (no root).
2. If T_1 and T_2 are disjoint RBTs with roots r_1 and r_2, then linking r_1 and r_2 to a new root r produces a new RBT with root r.



RBTs of height 0, 1, and 2



Can you prove that an RBT with $n \geq 1$ vertices must have $n - 1$ edges…?

Recursive definition of Rooted Binary Tree (RBT).

1. The empty tree ε is an RBT (no root).
2. If T_1 and T_2 are disjoint RBTs with roots r_1 and r_2, then linking r_1 and r_2 to a new root r produces a new RBT with root r.

STRUCTURAL INDUCTION

Our claim is $P(T)$: if T is an RBT with $n \geq 1$ vertices, then T has $n - 1$ edges.

Proof. We use structural induction to prove claim $P(T)$.

1. **[Base case]** For $P(\varepsilon)$, the claim is **T** since ε is not an RBT with $n \geq 1$…
2. **[Induction step]** We must prove that the constructor preserves P. For our constructor, let parent T_1 have v_1 vertices and e_1 edges, and let parent T_2 have v_2 vertices and e_2 edges. Assume $P(T_1) \wedge P(T_2)$ is **T** (i.e., our induction hypothesis). Let the constructed RBT T_{new} have v_{new} vertices and e_{new} edges. We must show $e_{new} = v_{new} - 1$.

   Case 1. $T_1 = T_2 = \varepsilon$. Here, tree T_{new} has $v_{new} = 1$, $e_{new} = 0$, and $e_{new} = v_{new} - 1$.

   Case 2. $T_1 = \varepsilon$; $T_2 \neq \varepsilon$. Here, $v_{new} = v_2 + 1$ and $e_{new} = e_2 + 1$. Applying the induction hypothesis for e_2, we have $e_{new} = e_2 + 1 = (v_2 - 1) + 1 = v_2 = v_{new} - 1$.

⋮

Our claim is $P(T)$: if T is an RBT with $n \geq 1$ vertices, then T has $n - 1$ edges.

Proof. We use structural induction to prove claim $P(T)$.

1. **[Base case]** For $P(\varepsilon)$, the claim is **T** since ε is not an RBT with $n \geq 1$…
2. **[Induction step]** We must prove that the constructor preserves P. For our constructor, let parent T_1 have v_1 vertices and e_1 edges, and let parent T_2 have v_2 vertices and e_2 edges. Assume $P(T_1) \wedge P(T_2)$ is **T** (i.e., our induction hypothesis). Let the constructed RBT T_{new} have v_{new} vertices and e_{new} edges. We must show $e_{new} = v_{new} - 1$.

   here, the constructor preserves property $P(T)$, i.e., $e_{new} = v_{new} - 1$

   Case 1. $T_1 = T_2 = \varepsilon$. Here, tree T_{new} has $v_{new} = 1$, $e_{new} = 0$, and $e_{new} = v_{new} - 1$.

   Case 2. $T_1 = \varepsilon$; $T_2 \neq \varepsilon$. Here, $v_{new} = v_2 + 1$ and $e_{new} = e_2 + 1$. Applying the induction hypothesis, we have $e_{new} = e_2 + 1 = (v_2 - 1) + 1 = v_2 = v_{new} - 1$.

   Case 3. $T_1 \neq \varepsilon$; $T_2 = \varepsilon$. Here, $v_{new} = v_1 + 1$ and $e_{new} = e_1 + 1$. Applying the induction hypothesis, we have $e_{new} = e_1 + 1 = (v_1 - 1) + 1 = v_1 = v_{new} - 1$.

   Case 4. $T_1 \neq \varepsilon$; $T_2 \neq \varepsilon$. Here, $v_{new} = v_1 + v_2 + 1$ and $e_{new} = e_1 + e_2 + 2$. Applying the induction hypothesis, we have $e_{new} = e_1 + e_2 + 2 = (v_1 - 1) + (v_2 - 1) + 2$. Thus, $e_{new} = v_1 + v_2 = v_{new} - 1$.
3. By structural induction, $P(T)$ is **T** for all $T \in$ RBT. ■

WHAT NEXT...?

Grade inquiries for Exam 1 and Homeworks 1 and 2 due by 11:59PM on October 21

Homework 3 is due by 11:59PM on October 20

- This week's October 19 recitations will be Q&A sessions focused on Homework 3...

Practice! Tinker! Practice! Tinker! Practice! Tinker! Practice! Tinker! Practice!