



Recursive Data Types

Ordered Recursive Binary Trees



Ordered Recursive Binary Trees

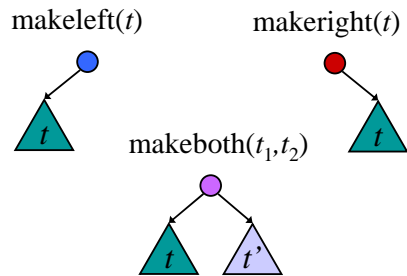
Defined recursively as follows:

- A single node ● is an RBT

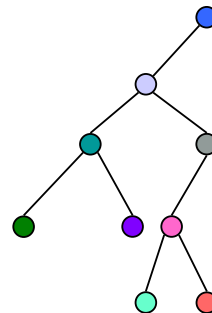


Recursive Binary Trees

If t, t' are RBTs, then so are:



Recursive Binary Tree



Recursive Functions on RBT

Recursive def of set of nodes

- $\text{nodes}(\bullet) ::= \{ \bullet \}$,
- $\text{nodes}(\text{makeleft}(t)) ::= \text{nodes}(t) \cup \{ \bullet \}$
- $\text{nodes}(\text{makeright}(t)) ::= \text{nodes}(t) \cup \{ \bullet \}$
- $\text{nodes}(\text{makeboth}(t_1, t_2)) ::= \text{nodes}(t_1) \cup \text{nodes}(t_2) \cup \{ \bullet \}$

where \bullet is the "new" root node.



Recursive Functions on RBT

Def. of depth

- $\text{depth}(\bullet) ::= 0$
- $\text{depth}(\text{makeleft}(t)) ::= 1 + \text{depth}(t)$
- $\text{depth}(\text{makeright}(t)) ::= 1 + \text{depth}(t)$
- $\text{depth}(\text{makeboth}(t_1, t_2)) ::= 1 + \max\{\text{depth}(t_1), \text{depth}(t_2)\}$

Tree Depth

depth 4

Copyright © Albert R. Meyer, 2002. All rights reserved. October 11, 2002 L6-2.7

Nodes versus Depth

Lemma: $|\text{nodes}(t)| + 1 \leq 2^{\text{depth}(t)+1}$

Proof by **Structural Induction**

Base Case: $t = \bullet$

$$|\text{nodes}(t)| + 1 = 1 + 1 = 2 = 2^{0+1} = 2^{\text{depth}(t)+1} \quad \text{OK!}$$

Copyright © Albert R. Meyer, 2002. All rights reserved. October 11, 2002 L6-2.8

Nodes versus Depth

Induction Case: makeleft(t)

Assume: $|\text{nodes}(t)| + 1 \leq 2^{\text{depth}(t)+1}$

To Prove:

$$|\text{nodes}(\text{makeleft}(t))| + 1 \leq 2^{\text{depth}(\text{makeleft}(t))+1}$$

Copyright © Albert R. Meyer, 2002. All rights reserved. October 11, 2002 L6-2.9

Nodes versus Depth

$$\begin{aligned} &|\text{nodes}(\text{makeleft}(t))| + 1 \\ &= (|\text{nodes}(t)| + 1) + 1 \text{ by def. of nodes} \\ &\leq (2^{\text{depth}(t)+1} + 1) + 1 \text{ by ind. hyp.} \\ &= 2^{\text{depth}(t)+1} + 2 \leq 2^{\text{depth}(t)+1} + 2^{\text{depth}(t)+1} \\ &= 2 \cdot 2^{\text{depth}(t)+1} = 2^{(\text{depth}(t)+1)+1} \\ &= 2^{\text{depth}(\text{makeleft}(t))+1} \text{ by def. of depth} \end{aligned}$$

Copyright © Albert R. Meyer, 2002. All rights reserved. October 11, 2002 L6-2.10

Nodes versus Depth

Induction Case: makeright(t)

Same

Copyright © Albert R. Meyer, 2002. All rights reserved. October 11, 2002 L6-2.11

Nodes versus Depth

Induction Case: makeboth(t_1, t_2)

Assume: $|\text{nodes}(t_1)| + 1 \leq 2^{\text{depth}(t_1)+1}$
 $|\text{nodes}(t_2)| + 1 \leq 2^{\text{depth}(t_2)+1}$

To Prove:

$$\begin{aligned} &|\text{nodes}(\text{makeboth}(t_1, t_2))| + 1 \\ &\leq 2^{\text{depth}(\text{makeboth}(t_1, t_2))+1} \end{aligned}$$

Copyright © Albert R. Meyer, 2002. All rights reserved. October 11, 2002 L6-2.12



Nodes versus Depth

$$\begin{aligned}
& |\text{nodes}(\text{makeboth}(t_1, t_2))| + 1 \\
&= (|\text{nodes}(t_1)| + |\text{nodes}(t_2)| + 1) + 1 \quad \text{def. of nodes} \\
&= (|\text{nodes}(t_1)| + 1) + (|\text{nodes}(t_2)| + 1) \\
&\leq 2^{\text{depth}(t_1)+1} + 2^{\text{depth}(t_2)+1} \quad \text{induction hyp.} \\
&\leq 2^{\max(\text{depth}(t_1), \text{depth}(t_2))+1} + 2^{\max(\text{depth}(t_1), \text{depth}(t_2))+1} \\
&= 2^{(\max(\text{depth}(t_1), \text{depth}(t_2))+1)+1} \\
&= 2^{\text{depth}(\text{makeboth}(t_1, t_2))+1} \quad \text{def. of depth}
\end{aligned}$$



Class Problems

Problems 1&2



Structural vs Ordinary Induction

Could **replace Structural Induction** on RBT's by **Strong Induction on size** of an RBT. (Not recommended.)

What about 18.01 Functions (18F)?
What is the "size" of the COSINE function?



Structural vs Ordinary Induction

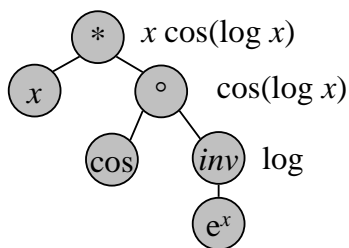
Could **replace Structural Induction** on F18's by **Strong Induction on size** of *derivation trees*.

(Still not recommended.)



Structural vs Ordinary Induction

derivation tree for $x \cdot \cos(\log(x))$



Structural vs Ordinary Induction

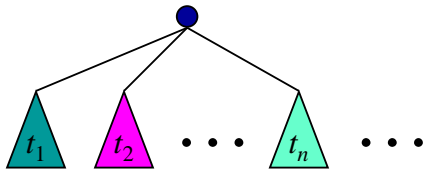
But Structural Induction is **also good on infinite trees**, and **can't be replaced** by induction on size. (... a *Meta-Theorem* of Formal Logic.)



Recursive Ordered Countable Trees

Single node is an RecCT

If $t_1, t_2, \dots, t_n \dots$ are RecCT's, so is:



May have **infinitely** many subtrees



Recursive Ordered Countable Trees

Theorem. **Every RecCT is Finite-path.**

Proof: By **structural induction** on $t \in \text{RecCT}$.

Base Case (t is one node):

- Has only a 0 length path.



RecCT \subseteq Finite-Path Trees

Induction Step (t has subtrees):

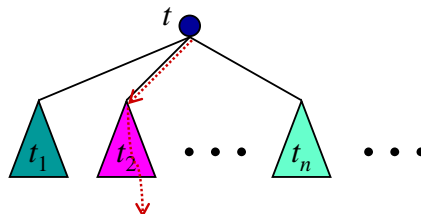
Suppose was **infinite path** from root of t .

Then would have infinite path in subtree.



RecCT \subseteq Finite-Path Trees

Induction Step (t has subtrees):



RecCT \subseteq Finite-Path Trees

Induction Step (t has subtrees):

Suppose was infinite path from root of t .

Then would have infinite path in subtree.

By hypothesis subtrees are Finite-Path, a contradiction. So t is F-P. **QED.**



Recursive Ordered Countable Trees

Converse: **Every Finite-path tree with Subtrees labelled 1,2,... is RecCT.**

Proof: By **Well-foundedness** of F-P under the **strict-subtree** partial order.

Consider a minimal F-P tree that is not a RecCT. Its subtrees are RecCT's by minimality. So by def of RecCT, it is too, a **contradiction.**

\therefore no 1,2,... labelled F-P tree is not a RecCT.



Class Problem

Problem 3

Copyright © Albert R. Meyer, 2002. All rights reserved. October 11, 2002

L6-2.25