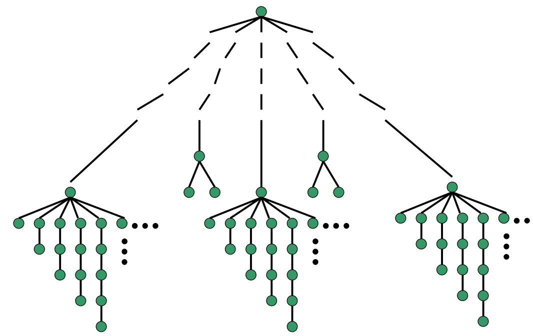
Mathematics for Computer Science

MIT 6.042J/18.062J

(*cont'd:*) Finite-Path Trees
Well-founded Orders

(*new:*) Recursive Definitions
Structural Induction

Finite-Path Trees



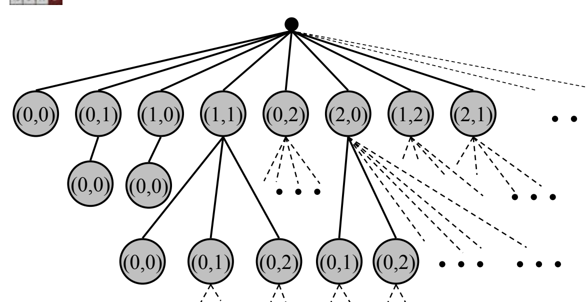
Finite-Path Trees

F-P tree:

Every directed path from the root is finite.

(Every "downward path" eventually stops.)

Finite-Path Trees



Game Tree for "Choose-a-pair" Game

Well-founded Partial Order

Every strictly decreasing chain is *finite*.

Same as:

Every nonempty subset has a *minimal element*.

Well-founded Partial Order

Example:

Lexicographic order on $\mathbb{N}^2$

$$(x,y) \prec_{\text{lex}} (x',y') \quad ::=$$

$$[\ x < x' \text{ or } (\ x=x' \text{ and } y<y'\ )\ ]$$

Well-founded Partial Order

Example:

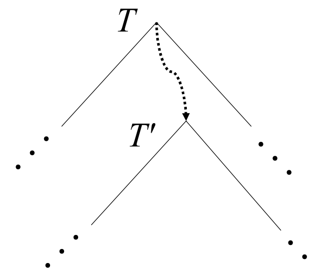
Strict subtree order on F-P trees

$$T' \prec_{ss} T \quad ::=$$

T' is a strict subtree of T

Well-founded Partial Order

$$T' \prec_{ss} T \quad ::=$$

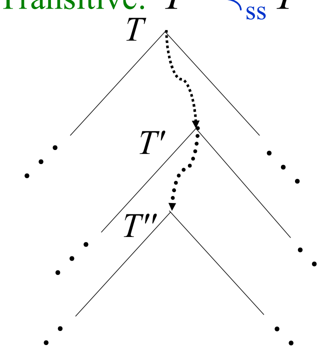


Well-founded Partial Order

Theorem: $\prec_{ss}$ is a Well-founded Partial Order on Finite-path Trees.

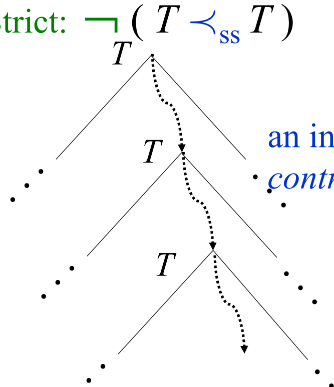
Well-founded Partial Order

Transitive: $T'' \prec_{ss} T' \prec_{ss} T$ ✔



Well-founded Partial Order

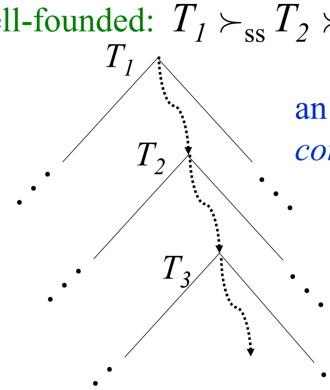
Strict: $\neg(T \prec_{ss} T)$ ✔



an infinite path *contradicts F-P*

Well-founded Partial Order

Well-founded: $T_1 \succ_{ss} T_2 \succ_{ss} T_2 \succ_{ss} \cdots$?



an infinite path *contradicts F-P*

✔

Terminating Games

Application:
In every 2-person terminating win-lose game, one player has a
Winning Strategy

Terminating Games

Proof: Suppose not.

Choose a *minimal* game-tree w/o a winning strategy for either player.
So all its subtrees do have winning strategies.

Terminating Games

If *any* subtree has a winning strategy for Player 1, then he has a winning strategy:
``*go to that subtree*.''

Terminating Games

If *no* subtree has a winning strategy for Player 1, then Player 2 wins in every subtree, so wins the whole game.
Either way, the whole game has a winning strategy!

Contradiction.

In-Class Problem

Problem 1

Recursive Definitions

Recursive Definitions

Recursive Definitions

Define something in terms of a simpler version of the same thing:

- Base case(s) that don't depend on anything else.
- Induction ("Construction:) case(s) that depend on simpler cases.

Example Definition: set E

Define set $E \subseteq \mathbb{Z}$, recursively:

- $0 \in E$ Base Case
- If $n \in$ E, then $n + 2 \in E$ Induction case 1
- If $n \in E$, then $-n \in E$ Induction case 2

E : 0, 0+2, (0+2)+2, ((0+2)+2) +2

0, 2, 4, 6, . . . all even numbers

Also, –2, –4, –6, …

Recursive Definition: Extremal Clause

So, E *contains* the even integers

Anything Else? No!

- $0 \in E$
- If $n \in E$, then $n + 2 \in E$
- If $n \in E$, then $-n \in E$
- That's All!

Extremal Clause

Implicit part of definition

Example Definition: set E

So E is exactly:
The Even Integers

Set of Strings Example

Define set of strings, $S \subseteq \{a, b\}^*$

If $x, y \in S$, then the following are in S:

- $\lambda \in S$, (the *empty string*)
- bxa
- xy
- axb

Set of Strings Example

Lemma: Every x in S has an equal number of a's and b's.

Proof by Structural Induction on the definition of S

Structural Induction on S

Proof: Let

EQ ::= {strings with equal # of a's and b's}

$P(x)$ ::= $x \in S \rightarrow x \in EQ$

Base Case: $x = \lambda$.

0 a's and 0 b's. OK

Structural Induction on S

Inductive Step

Assume: $P(x)$ and $P(y)$

To Prove: $P(\text{a}x\text{b})$, $P(\text{b}x\text{a})$, and $P(xy)$

Case 1: axb

has $k + 1$ a's and b's if x has k of each. OK

Structural Induction on S

Inductive Step

Case 2: bxa SAME

Case 3: xy

has $k_x + k_y$ of each. OK

So $S \subseteq$ EQ

Structural Induction on S

Rest of Story:

Theorem: S = EQ

Show by strong induction on length of string in EQ. Proof in Notes 6.

Recursive Data Types

Ordered Recursive Binary Trees

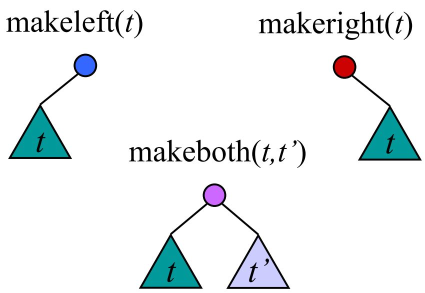
Recursive Binary Trees

Defined recursively as follows:

- A single node r is an RBT: 

Recursive Binary Trees

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



Recursive Binary Tree Example