



CSCI 2200
Foundations of Computer Science

Lecture 8:
Structural Induction



CSCI 2200
Foundations of Computer Science

Lecture 8: Structural Induction

“Bloody Tears” by Kenichi Matsubara,
from *Castlevania II: Simon’s Quest*



Today's tasks

- HW questions
- Structural induction

Two reminders

- Every recursive set definition has one or more *base cases* and one or more *recursive rules* – “constructors”, if you will.
 - Example: $0 \in S$; $x \in S \rightarrow (x + 4) \in S$ (What set is this?)
- Most of our proofs relating to sets revolve around the *subset* relation – is one set entirely contained in another?


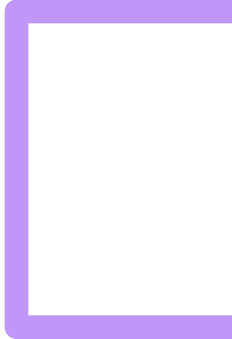
Two questions

$$0 \in S; \quad x \in S \rightarrow (x + 4) \in S$$

- Given the previous slide, if there is some property P we're interested in relating to a recursive set, we tend to ask one of two questions:
 - Does every member of our recursive set have property P ? (i.e. $S \subseteq S_P$?) For example: "Is every member of S even?"
 - Is every object with property P in our recursive set? (i.e. $S_P \subseteq S$?) For example: "Is every even number contained in S ?"
- The second question can be tricky. For the first one, we have a "standard" approach...



... more induction!

- To get the intuition:
 - The first two orks had blue eyes. Whenever two blue-eyed orks mate, the offspring always have blue eyes.
 - Can you conclude that all orks have blue eyes?
 - Sure – where would any other color come from?
 - Your textbook author refers to this sort of reasoning as “structural induction.”
 - Really, this is just a special case of strong induction.
- 
- 

Structural induction

- When S is a recursively defined set, to prove that some predicate $P(x)$ is true for every element $s_i \in S$:
 - Prove $P(s_1), \dots, P(s_b)$ for every base case $s_1 \dots s_b$. (There might only be one of these.)
 - For every recursive “constructor” rule, prove that if $P(x)$ holds for the “parent” (i.e. earlier, smaller, what have you) elements in that rule, then it must also be true for the “child” elements.
 - The idea here is that every rule maintains P .

Example structural induction proof

- A common task for compilers is to be able to match parentheses. $((()()))$ is valid, $((()()))$ is not.
- We can define M , the set of strings consisting of correctly matched parentheses, in a recursive manner:
 - $\varepsilon \in M$; $x, y \in M \rightarrow (x)y \in M$
- To prove that M is precisely the set of consisting of correctly matched parentheses, we need to show:
 - Every $m \in M$ is correctly matched. (str. induction proof)
 - Every correctly matched string of parentheses is in M . (Very much not a structural induction proof.)

Example S.I. proof

$$\varepsilon \in M; \quad x, y \in M \rightarrow (x)y \in M$$

Claim: Every string in M is correctly matched.

... Wait, what does "correctly matched" mean?

- As an obvious starter: The number of left parens and the number of right parens must be equal.
- But that's not enough. Consider:) () (
- We also need the condition that any *prefix* of a string in M must have at least as many (as it has).
 - That is, if we read from left to right and count each (as +1 and each) as -1, our running total must never be negative.

Example S.I. proof

$$\varepsilon \in M; \quad x, y \in M \rightarrow (x)y \in M$$

Claim: Every string in M is correctly matched.

Proof: By structural induction.

Base case: The empty string is trivially correctly matched.

Induction step: If x and y are correctly matched, then so is $(x)y$. (Note: only one rule, so our induction step is simple)

If x and y each have equal numbers of left & right parens, then adding one more of each keeps them equal.

The prefixes of $(x)y$ are:

- * Just $($, which clearly has more lefts than rights.
- * $($ followed by a prefix of x ; by the I.H., all prefixes of x have more lefts than rights, adding one more left will maintain that property.
- * (x) which has equal lefts & rights.
- * or (x) followed by a prefix of y . By the I.H., all prefixes of y have more lefts than rights, and (x) has equal numbers of both. ■

NOT structural induction

$$\varepsilon \in M; \quad x, y \in M \rightarrow (x)y \in M$$

- Claim: Every correctly matched set of parenthesis is in M .
- *NOTE*: Structural induction is not the tool here, because we have no rule (yet) that describes the production of every correctly matched set of parentheses.
 - Once we finish this proof, we can then prove OTHER things about the set of strings of matched parenthesis via structural induction. But since our definition of “matched” is not recursive, we can’t start from there.

NOT structural induction

$$\varepsilon \in M; \quad x, y \in M \rightarrow (x)y \in M$$

- Claim: Every correctly matched set of parenthesis is in M .
- Proof by contradiction.
 - Assume that there are correctly matched sets of parentheses that are not in M . Let s be the shortest such string.
 - Since s is correctly matched, it must start with (– otherwise it would have a prefix with more lefts than rights. Now, scan to the right until the first point in the string where (and) are equal.
 - We can now write s as $(x)y$, where y is everything after the designated point in the string.
 - x and y must both be matched 🙌 and are smaller than s . If both are in M , then so is s . If one of them is not, then we have a smaller matched string than s that is also not in M . Either way, contradiction.

Palindromes

- Recall that the definition of a palindrome is a string that reads the same forwards and backwards, i.e. $w = w^R$.
- Also recall from last class that we proposed the following recursive construction of binary palindromes:
 - $\varepsilon \in P$; $0 \in P$; $1 \in P$
 - $x \in P \rightarrow 0x0 \in P$
 - $x \in P \rightarrow 1x1 \in P$
- Claim: Every element of P is a palindrome.

Palindromes – structural induction proof

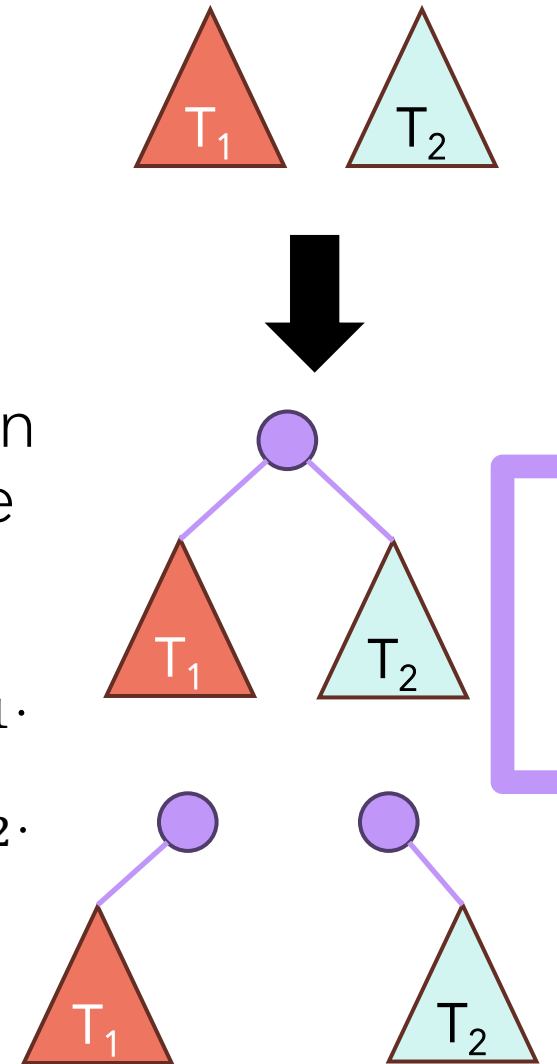
- $\varepsilon \in P; 0 \in P; 1 \in P$
- $x \in P \rightarrow 0x0 \in P$
- $x \in P \rightarrow 1x1 \in P$
- Claim: Every element of P is a palindrome. Proof by S.I.
- Base cases: $\varepsilon^R = \varepsilon; 0^R = 0; 1^R = 1$
- Induction: Observe that $(0x0)^R = 0x^R0$ and $(1x1)^R = 1x^R1$. If $x^R = x$, then these are equal to $0x0$ and $1x1$ respectively, and we're done – “palindromeness” is preserved by both of our construction rules. ■

An observation

- $\varepsilon \in P$; $0 \in P$; $1 \in P$
- $x \in P \rightarrow 0x0 \in P$
- $x \in P \rightarrow 1x1 \in P$
- How would you prove that **01010** was in P ?
 - A derivation – show how to produce the string.
 - $0 \in P \rightarrow 101 \in P \rightarrow 01010 \in P$
- How would you prove **110010** was not in P ?
 - Contraposition: Our S.I. proved $x \in P \rightarrow x^R = x$
 - But if we reverse **110010** we get **010011**. $x^R \neq x \rightarrow x \notin P$

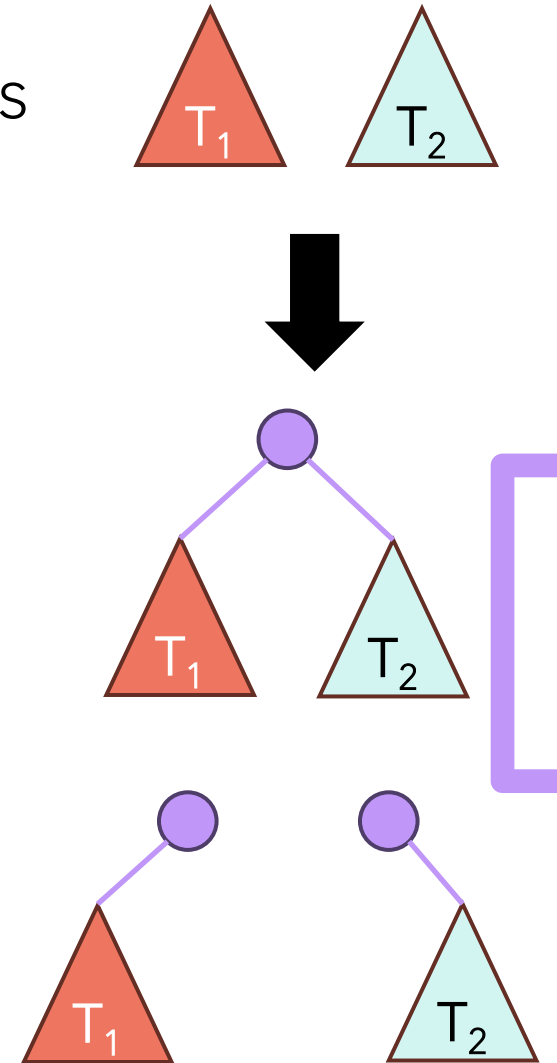
Rooted binary trees

- A rooted binary tree is a recursive structure, defined like so:
 - A single node is an RBT (and its own root).
 - If T_1 and T_2 are RBTs with $T_1 \cap T_2 = \emptyset$, then we can construct a new RBT by creating a new node (the "root") and doing one of three things:
 - Adding a link from the new node to the root of T_1 .
 - Adding a link from the new node to the root of T_2 .
 - Doing both.



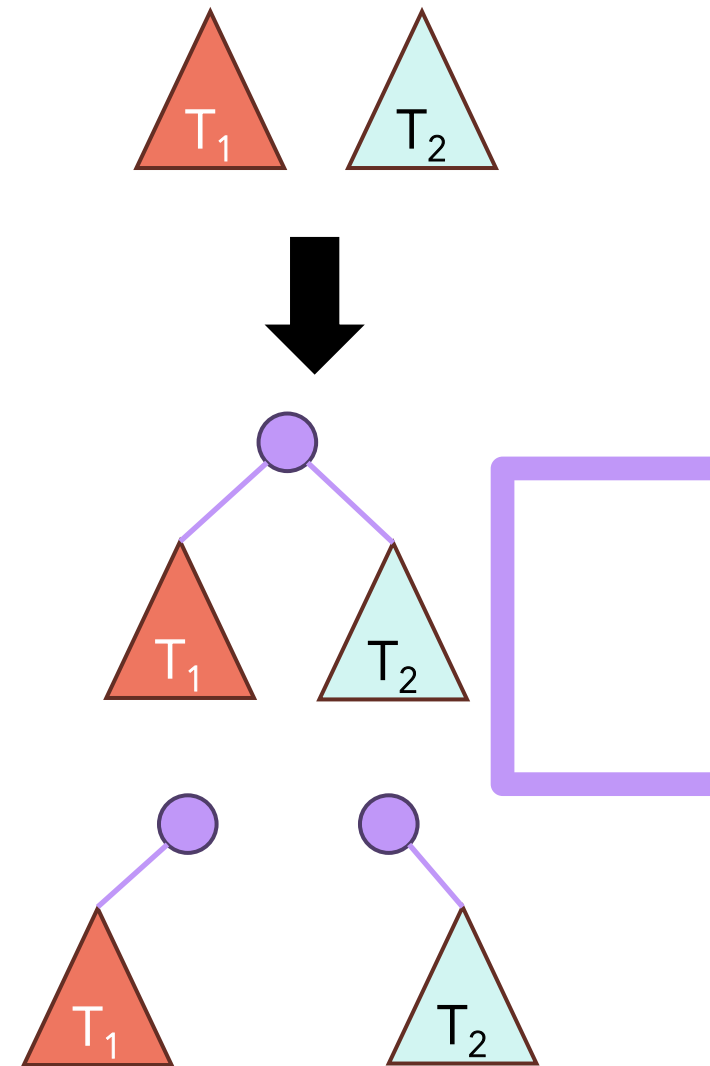
Putting the “structure” in structural induction

- Claim: Every rooted binary tree with n nodes has exactly $n - 1$ links.
- Prove the above claim by structural induction.



Putting the “structure” in structural induction

- Claim: Every rooted binary tree with n nodes has exactly $n - 1$ links.
- Base case: $n = 1$. The RBT has no links.
- Induction: Let T_1 have j nodes and $j - 1$ links, and T_2 have k nodes and $k - 1$ links.
 - First rule: The new tree has $j + 1$ nodes and j links.
 - Second rule: The new tree has $k + 1$ nodes and k links.
 - Third rule: The new tree has $j + k + 1$ nodes and $(j - 1) + (k - 1) + 2 = j + k$ links.



A final example

- Consider these rules for producing arithmetic expressions:
 - $1 \in A$
 - $x \in A \rightarrow (x + 1 + 1) \in A$
 - $x, y \in A \rightarrow (x \times y) \in A$
- What do the elements of A look like?
 - $(1 + 1 + 1); ((1 \times (1 + 1 + 1) + 1 + 1); ((1 \times 1) \times (1 \times 1)); \dots$
- What property do they all share?
- Prove it!

A final example

- $1 \in A$
- $x \in A \rightarrow (x + 1 + 1) \in A$
- $x, y \in A \rightarrow (x \times y) \in A$
- Claim: Each element of A comes out to an odd value.
- Base case: 1 is odd.
- Induction: Two rules:
 - If x is odd, then so is $x + 2$.
 - If x and y are both odd, then so is $x \times y$.

Summing up

- Use structural induction when:
 - You have a recursive defined set S .
 - You want to prove every element of S has some property P .
- To write a structural induction proof:
 - Prove that every base element of S has the property P .
 - Prove that every recursive rule for generating other elements of S maintains the property P .
- Do not use structural induction to prove that every object with property P is in S .

Questions?

