Poll: Strong vs. Structural Induction

What is the difference between strong induction and structural induction?

A) Strong induction is less complex, but also less powerful: some claims can be proven using structural induction that cannot be proven using strong induction.

B) Strong induction can be used to prove statements parameterized by an integer (e.g., $\forall n \in \mathbb{Z}^{\geq 0} P(n)$), whereas structural induction can be used to prove statements about non-integer things like data structures (lists, trees, etc.).

C) In the inductive step of the proof, strong induction requires making fewer assumptions (hence it is stronger).

D) None of the above

COSC 290 Discrete Structures

Lecture 15: Structural induction on trees

Prof. Michael Hay

Monday, Mar. 5, 2018

Colgate University

Plan for today

1. Examples of strong induction

2. Structural Induction

Examples of strong induction

Jacobsthal numbers & Tilings

Jacobsthal numbers are defined as follows:

- $J_0 := 0$
- $J_1 := 1$
- $J_n := J_{n-1} + 2J_{n-2}$ for $n \geq 2$

We have two separate claims about Jacobsthal numbers.

1. **Claim**: for any $n \geq 0$, given $n \times 2$ grid, the number of tilings using either 1×2 dominoes or 2×2 squares is J_{n+1}.
2. **Claim**: $J_n = \frac{2^n - (-1)^n}{3}$

Proof for first claim shown on board.

Proof of closed form solution for J_n

Claim: $J_n = \frac{2^n - (-1)^n}{3}$

Proof by strong induction Proof by induction on n.

- Base case ($n = 0$): $J_0 := 0$ and when $n = 0$, we have $\frac{2^n-(-1)^n}{3} = \frac{2^0-(-1)^0}{3} = \frac{1-1}{3} = 0$.
- Base case ($n = 1$): $J_1 := 1$ and when $n = 1$, we have $\frac{2^n-(-1)^n}{3} = \frac{2^1-(-1)^1}{3} = \frac{2-(-1)}{3} = 1$.
- Inductive case ($n \geq 2$): Assume for all $0 \leq m \leq n-1$, that $J_m = \frac{2^m-(-1)^m}{3}$. Want to show $J_n = \frac{2^n-(-1)^n}{3}$.
  Math on next slide...

Proof continued...

Inductive case ($n \geq 2$): Assume for all $0 \leq m \leq n-1$, that $J_m = \frac{2^m-(-1)^m}{3}$. Want to show $J_n = \frac{2^n-(-1)^n}{3}$.

$$\begin{aligned}
J_n &= J_{n-1} + 2J_{n-2} && \text{definition of } J_n \\
&= \frac{2^{(n-1)} - (-1)^{(n-1)}}{3} + 2\frac{2^{(n-2)} - (-1)^{(n-2)}}{3} && \text{inductive hypothesis} \\
&= \frac{1}{3}\left[2^{(n-1)} - (-1)^{(n-1)} + 2 \cdot 2^{(n-2)} - 2(-1)^{(n-2)}\right] && \text{rearranging terms} \\
&= \frac{1}{3}\left[2^n - (-1)^{(n-1)} - 2(-1)^{(n-2)}\right] && \text{alegbra on red parts} \\
&= \begin{cases} \frac{1}{3}[2^n - 1] & n \text{ is even} \\ \frac{1}{3}[2^n + 1] & n \text{ is odd} \end{cases} && \text{simplifying blue stuff} \\
&= \frac{1}{3}[2^n - (-1)^n] = \frac{2^n - (-1)^n}{3} && \square
\end{aligned}$$

Structural Induction

Recursively defined structures

A recursively defined structure is a structure defined in terms of one or more *base* cases and one or more *inductive* cases.

Applications in computer science

Many fundamental computer science structures are recursively defined structures:

- lists
- trees
- propositional logic
- circuits
- syntax of all programming languages

Having the ability to reason about such structures is important!

Real-world Example: Apache Spark RDDs

Many practical systems/applications are built using recursively defined structures.

Apache Spark Resilient Distributed Datasets (RDDs).

An RDD is either a dataset (e.g., collection of files) or it is the result of a transformation of one or more RDDs. Transformations include operations such as map, filter, sample, intersection, union, etc.

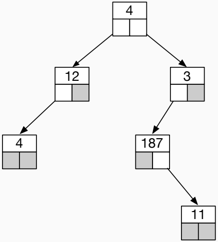
Example: Binary Tree

A binary tree is either:

a) (base case) an empty tree, denoted *null*
b) (inductive case) a root node x, a left subtree T_ℓ, and a right subtree T_r where x is an arbitrary value and T_ℓ and T_r are both *binary trees*.

Tree: nodes and edges

A tree with six nodes and five edges.



Property of trees

Claim: For any binary tree T, if T is non-empty, then $edges(T) = nodes(T) - 1$ where $edges(T)$ denotes the number of edges in T and $nodes(T)$ denotes the number of nodes.

Proof of claim

- **Claim**: Let T be a binary tree. If T is non-empty, then $edges(T) = nodes(T) - 1$.
- **Proof by structural induction**:
  - **Base cases:** T is empty, therefore...
  - **Inductive case:** T is non-empty, consisting of node x and left and right subtrees T_ℓ and T_r.
    Therefore...

Poll: proof for the base case?

- **Claim**: If T is non-empty, then $edges(T) = nodes(T) - 1$.
- **Proof by structural induction**:
  - **Base cases:** T is empty, therefore... what goes here?

A) $nodes(T) = 1$ and $edges(T) = 0$ because T is empty.

B) The claim does not apply because T is empty.

C) The claim does is false because T is empty.

D) The claim is true because T is empty.

E) None of the above.

Base case

- **Claim**: If T is non-empty, then $edges(T) = nodes(T) - 1$.
- **Proof by structural induction**:
  - **Base cases:** T is empty, therefore the statement is true (because the antecedent is False and $p \implies q$ is True whenever p is False).

Poll: Inductive case

Claim: If T is non-empty, then $edges(T) = nodes(T) - 1$.

Inductive case: T is non-empty, consisting of node x and left and right subtrees T_ℓ and T_r.

Inductive hypothesis: $edges(T_\ell) = nodes(T_\ell) - 1$ (same for T_r).

$$
\begin{aligned}
nodes(T) &= 1 + nodes(T_\ell) + nodes(T_r) && \text{(b. +1 for root)}\\
edges(T) &= (1 + edges(T_\ell)) + (1 + edges(T_r)) && \text{(c. +1 for each edge)}\\
&= (1 + (nodes(T_\ell) - 1)) + (1 + (nodes(T_r) - 1)) && \text{(d. inductive hypo.)}\\
&= nodes(T_\ell) + nodes(T_r)\\
&= nodes(T) - 1
\end{aligned}
$$

A) The proof is correct.
B) Step b
C) Step c
D) Step d
E) None of above / more than one

- The inductive hypothesis only asserts a property for non-empty trees!
- What if T_ℓ is empty? What if T_r is empty?

Proof of claim

Claim: If T is non-empty, then $edges(T) = nodes(T) - 1$.

Inductive case: T is non-empty, consisting of node x and left and right subtrees T_ℓ and T_r. Cases:

- $T_\ell =$ null and $T_r =$ is null: $nodes(T) = 1$ and $edges(T) = 0$.
- $T_\ell \neq$ null and $T_r =$ null:

$$nodes(T) = 1 + nodes(T_\ell)$$
$$edges(T) = 1 + edges(T_\ell) = 1 + (nodes(T_\ell) - 1) = nodes(T) - 1$$

- $T_\ell =$ null and $T_r \neq$ null: same ideas as previous.
- $T_\ell \neq$ null and $T_r \neq$ null:

$$
\begin{aligned}
nodes(T) &= 1 + nodes(T_\ell) + nodes(T_r)\\
edges(T) &= 2 + edges(T_\ell) + edges(T_r)\\
&= 2 + (nodes(T_\ell) - 1) + (nodes(T_r) - 1)\\
&= nodes(T_\ell) + nodes(T_r) = nodes(T) - 1
\end{aligned}
$$