Structural Induction

Davood Pour Yousefian Barfeh

Discrete Structures
Department of Computer Science

Structural Induction

- Structural induction is a technique that allows us to apply induction on recursive definitions even if there is no integer.
- Structural induction is also no more powerful than regular induction, but can make proofs much easier

Structural Induction Overview

- Suppose we have:
  1. recursively defined structure S
  2. property P we'd like to prove about S

- Structural induction works as follows:
  1. **Base case:** Prove P about base case in recursive definition
  2. **Inductive step:** Assuming P holds for sub-structures used in the recursive step of the definition, show that P holds for the recursively constructed structure.

Sample 1

- Consider the following recursively defined set S:

    1. a ∈ S
    2. If x ∈ S, then (x) ∈ S

- Prove by structural induction that every element in S contains an equal number of right and left parentheses.
    - **Base case:** a has 0 left and 0 right parentheses
    - **Inductive step:** By the inductive hypothesis, x has equal number, say n, of right and left parentheses.
- Thus, (x) has n + 1 left and n + 1 right parentheses.

Induction Principle for Natural Numbers

- The set of natural numbers is the set $N=\{0,1,2,3,\ldots\}$.
- In the past, opinions have differed as to whether the set of natural numbers should start with 0 or 1, but these days most mathematicians take them to start with 0.
- Logicians often call the function $s(n)=n+1$ the successor function, since it maps each natural number, n, to the one that follows it.
- What makes the natural numbers special is that they are generated by the number zero and the successor function, which is to say, the only way to construct a natural number is to start with 0 and apply the successor function finitely many times.

Principle of Induction:

Let P be any property of natural numbers. Suppose P holds of zero, and whenever P holds of a natural number n, then it holds of its successor, n+1. Then P holds of every natural number.

This reflects the image of the natural numbers as being generated by zero and the successor operation: by covering the zero and successor cases, we take care of all the natural numbers.

Theorem: For every natural number n,

$$1+2+\ldots+2^n=2^{n+1}-1.$$

Proof: We prove this by induction on n. In the base case, when $n=0$, we have $1=2^{0+1}-1$, as required.

- For the induction step, fix n, and assume the *induction hypothesis*

$$1+2+\ldots+2^n=2^{n+1}-1.$$

- We need to show that this same claim holds with **n** replaced by **n+1**. But this is just a calculation:

$$\begin{aligned}1+2+\ldots+2^{n+1}&=(1+2+\ldots+2^n)+2^{n+1}\\&=2^{n+1}-1+2^{n+1}\\&=2\cdot 2^{n+1}-1\\&=2^{n+2}-1.\end{aligned}$$

Induction Principle for Strings

- The elements of a string are **symbols** drawn from a set of symbols called an ALPHABET, which is usually denoted Σ.
- For example, if $\Sigma=\{a,b\}$, then strings can consist of sequences of *a*s and *b*s.
- Σ^* denotes the set of all possible strings on the alphabet Σ, and always includes the empty string, which is denoted λ.
- Every symbol of Σ is also a string of length 1. (Note: this property in particular distinguishes strings from lists; but in general reasoning about strings is quite similar to reasoning about lists.)

- The basic way to construct strings is by **concatenation**.
- If $s1$ and $s2$ are strings, then their concatenation is also a string and is written $s1s2$ or $s1 \cdot s2$ if punctuation is needed for clarity.

Concatenation is defined as follows:

Axiom 4.1 (Concatenation):

$$\forall s \in \Sigma^* \lambda \cdot s = s \cdot \lambda = s$$

$$\forall a \in \Sigma \ \forall s1, s2 \in \Sigma^* \ (a \cdot s1) \cdot s2 = a \cdot (s1 \cdot s2)$$

- **Axiom 4.2 (Strings)**:

  The empty string is a string: $\lambda \in \Sigma$

  Joining any symbol to a string gives a string: $\forall a \in \Sigma\ \forall s \in \Sigma^*\ a \cdot s \in \Sigma^*$

Because these axioms do not strictly *define* strings, we need an induction principle to construct proofs over all strings:

Axiom 4.3 (String Induction):

For any property P,
if $P(\lambda)$ and $\forall a \in \Sigma\ \forall s \in \Sigma^*\ (P(s) \Rightarrow P(a \cdot s))$,
then $\forall s \in \Sigma^*\ P(s)$.

Induction Principle for Binary Trees

- **Trees** are a fundamental data structure in computer science, underlying efficient implementations in many areas including databases, graphics, compilers, editors, optimization, game-playing, and so on.
- Trees are also used to represent expressions in formal languages.
- Here we study their most basic form: the **binary tree**.
- Binary trees include lists (as in Lisp and Scheme), which have `nil` as the rightmost leaf.

- In the theory of binary trees, we begin with **atoms**, which are trees with no branches.
- **A** is the set of atoms, which may or may not be finite.
- We construct trees (**T**) using the • (cons) operator.

- **Axiom 4.5 (Binary Trees)**:

  Every atom is a tree: $\forall a \in \mathbf{A}\ [a \in \mathbf{T}]$

  Consing any two trees gives a tree: $\forall t1, t2 \in \mathbf{T}\ [t1 \bullet t2 \in \mathbf{T}]$

- The induction principle for trees says that if P holds for all atoms, and if the truth of P for any two trees implies the truth of P for their composition, then P holds for all trees:

- **Axiom 4.6 (Binary Tree Induction)**:

  For any property P,

  if $\forall a \in \mathbf{A}\ P(a)$

  and $\forall\ t1,t2 \in \mathbf{T}\ [P(t1) \wedge P(t2) \Rightarrow P(t1 \bullet t2)]$

  then $\forall\ t \in \mathbf{T}\ P(t)$.

- Many useful predicates and functions can be defined on trees, including:

- *leaf*(a, t) is true if atom a is a leaf of tree t.
- $t1 \prec t2$ is true if tree $t1$ is a proper subtree of tree $t2$.
- *count*(t) denotes the number of leaves of the tree t.
- *depth*(t) denotes the **depth** of the tree, where any atom has depth 0.
- *balanced*(t) is true if t is a balanced binary tree.

Induction of Pair of Numbers

- Often we need to prove properties over the **Cartesian product** of some given sets.
- The Cartesian product of sets **A** and **B** is written **A** x **B**.
- It is the set of all **pairs** (a,b) where $a \in$ **A** and $b \in$ **B**.
- For example, the set **N** x **N** is the set of all pairs of natural numbers.
- Such sets arise when we prove properties of functions with two arguments, when we prove facts about all points on a grid, etc.

- **Axiom 4.10 (Strong Induction (Pairs)):**

  For any property P,

  if $\forall x, y \in \mathbf{N}$

  $[\forall\, x', y' \in \mathbf{N}\ (x'+y') < (x+y) \Rightarrow P(x', y')] \Rightarrow P(x, y)$

  then $\forall x, y \in \mathbf{N}\ P(x, y)$.

References

- [1] (2020). Retrieved 3 September 2020, from https://www.cs.utexas.edu/~isil/cs311h/lecture-induction3-6up.pdf
- [2] (2020). Retrieved 16 August 2020, from https://dinus.ac.id/repository/docs/ajar/Discrete_-_Rosen-Ed7.pdf
- [3] 17. The Natural Numbers and Induction — Logic and Proof 3.18.4 documentation. (2020). Retrieved 4 September 2020, from https://leanprover.github.io/logic_and_proof/the_natural_numbers_and_induction.html#:~:text=The%20principle%20of%20induction%20provides,called%20a%20proof%20by%20induction.