Structural induction

From Wikipedia, the free encyclopedia

Structural induction is a proof method that is used in mathematical logic (e.g., in the proof of Łoś' theorem), computer science, graph theory, and some other mathematical fields. It is a generalization of mathematical induction over natural numbers, and can be further generalized to arbitrary Noetherian induction. **Structural recursion** is a recursion method bearing the same relationship to structural induction as ordinary recursion bears to ordinary mathematical induction.

Structural induction is used to prove that some proposition $P(x)$ holds for all x of some sort of recursively defined structure such as lists or trees. A well-founded partial order is defined on the structures ("sublist" for lists and "subtree" for trees). The structural induction proof is a proof that the proposition holds for all the minimal structures, and that if it holds for the immediate substructures of a certain structure S, then it must hold for S also. (Formally speaking, this then satisfies the premises of an axiom of well-founded induction, which asserts that these two conditions are sufficient for the proposition to hold for all x.)

A structurally recursive function uses the same idea to define a recursive function: "base cases" handle each minimal structure and a rule for recursion. Structural recursion is usually proved correct by structural induction; in particularly easy cases, the inductive step is often left out. The *length* and ++ functions in the example below are structurally recursive.

For example, if the structures are lists, one usually introduces the partial order '<' in which $L < M$ whenever list L is the tail of list M. Under this ordering, the empty list [] is the unique minimal element. A structural induction proof of some proposition $P(l)$ then consists of two parts: A proof that $P([])$ is true, and a proof that if $P(L)$ is true for some list L, and if L is the tail of list M, then $P(M)$ must also be true.

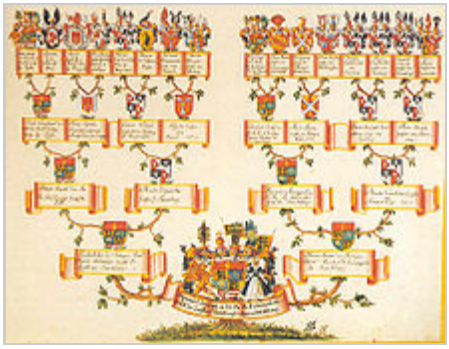
Eventually, there may exist more than one base case, and/or more than one inductive case, depending on how the function or structure was constructed. In those cases, a structural induction proof of some proposition $P(l)$ then consists of: **A)** a proof that $P(BC)$ is true for each base case BC, and **B)**: a proof that if $P(I)$ is true for some instance I, and M can be obtained from I by applying any one recursive rule once, then $P(M)$ must also be true.

Contents

Examples

An ancestor tree is a commonly known data structure, showing the parents, grandparents, etc. of a person as far as known (see picture for an example). It is recursively defined:

- in the simplest case, an ancestor tree shows just one person (if nothing is known about his/her parents);
- alternatively, an ancestor tree shows one person and, connected by branches, the two ancestor subtrees of his/her parents (using for brevity of proof the simplifying assumption that if one of them is known, both are).



Ancient ancestor tree, showing 31 persons in 5 generations.

As an example, the property "An ancestor tree extending over g generations shows at most 2^g-1 persons" can be proven by structural induction as follows:

- In the simplest case, the tree shows just one person and hence one generation; the property is true for such a tree, since $1 \le 2^1$-1.
- Alternatively, the tree shows one person and his/her parents' trees. Since each of the latter is a substructure of the whole tree, it can be assumed to satisfy the property to be proven (a.k.a. the *induction hypothesis*). That is, $p \le 2^g$-1 and $q \le 2^h$-1 can be assumed, where g and h denotes the number of generations the father's and the mother's subtree extends over, respectively, and p and q denote the numbers of persons they show.
  - In case $g \le h$, the whole tree extends over $1+h$ generations and shows $p+q+1$ persons, and $p+q+1 \le (2^g\text{-}1) + (2^h\text{-}1) + 1 \le 2^h + 2^h - 1 = 2^{1+h} - 1$, i.e. the whole tree satisfies the property.
  - In case $h \le g$, the whole tree extends over $1+g$ generations and shows

$p+q+1 \leq 2^{1+g}$ - 1 persons by similar reasoning, i.e. the whole tree satisfies the property in this case also.

Hence, by structural induction, each ancestor tree satisfies the property.

As another, more formal example, consider the following property of lists:

```
length (L ++ M) = length L + length M         [EQ]
```

Here ++ denotes the list concatenation operation, and L and M are lists.

In order to prove this, we need definitions for length and for the concatenation operation. Let (*h*:*t*) denote a list whose head (first element) is *h* and whose tail (list of remaining elements) is *t*, and let [] denote the empty list. The definitions for length and the concatenation operation are:

```
length []     = 0                 [LEN1]
length (h:t)  = 1 + length t      [LEN2]
```

```
[]    ++ list = list              [APP1]
(h:t) ++ list = h : (t ++ list)   [APP2]
```

Our proposition *P*(*l*) is that EQ is true for all lists *M* when *L* is *l*. We want to show that *P*(*l*) is true for all lists *l*. We will prove this by structural induction on lists.

First we will prove that *P([])* is true; that is, EQ is true for all lists *M* when *L* happens to be the empty list []. Consider EQ:

```
    length (L ++ M)  = length ([] ++ M)
                     = length M                    (by APP1)
                     = 0 + length M
                     = length [] + length M        (by LEN1)
                     = length L  + length M
```

So this part of the theorem is proved; EQ is true for all *M*, when *L* is [], because the left-hand side and the right-hand side are equal.

Next, consider any nonempty list *I*. Since *I* is nonempty, it has a head item, x, and a tail list, xs, so we can express it as (x:xs). The induction hypothesis is that EQ is true for all values of *M* when *L* is *xs*:

```
length (xs ++ M) = length xs + length M    (hypothesis)
```

We would like to show that if this is the case, then EQ is also true for all values of

M when $L = I$ = (x:xs). We proceed as before:

```
length L  + length M      = length (x:xs) + length M
                          = 1 + length xs + length M     (by LEN2)
                          = 1 + length (xs ++ M)         (by hypothesis)
                          = length (x: (xs ++ M))        (by LEN2)
                          = length ((x:xs) ++ M)         (by APP2)
                          = length (L ++ M)
```

Thus, from structural induction, we obtain that P(L) is true for all lists L.

Well-ordering

Just as standard mathematical induction is equivalent to the well-ordering principle, structural induction is also equivalent to a well-ordering principle. If the set of all structures of a certain kind admits a well-founded partial order, then every nonempty subset must have a minimal element. (This is the definition of "well-founded".) The significance of the lemma in this context is that it allows us to deduce that if there are any counterexamples to the theorem we want to prove, then there must be a minimal counterexample. If we can show the existence of the minimal counterexample implies an even smaller counterexample, we have a contradiction (since the minimal counterexample isn't minimal) and so the set of counterexamples must be empty.

As an example of this type of argument, consider the set of all binary trees. We will show that the number of leaves in a full binary tree is one more than the number of interior nodes. Suppose there is a counterexample; then there must exist one with the minimal possible number of interior nodes. This counterexample, C, has n interior nodes and l leaves, where $n+1 \neq l$. Moreover, C must be nontrivial, because the trivial tree has $n = 0$ and $l = 1$ and is therefore not a counterexample. C therefore has at least one leaf whose parent node is an interior node. Delete this leaf and its parent from the tree, promoting the leaf's sibling node to the position formerly occupied by its parent. This reduces both n and l by 1, so the new tree also has $n+1 \neq l$ and is therefore a smaller counterexample. But by hypothesis, C was already the smallest counterexample; therefore, the supposition that there were any counterexamples to begin with must have been false. The partial ordering implied by 'smaller' here is the one that says that $S < T$ whenever S has fewer nodes than T.

See also

- Coinduction
- Initial algebra

References

- Hopcroft, John E.; Rajeev Motwani, Jeffrey D. Ullman (2001). *Introduction to Automata Theory, Languages, and Computation* (2nd ed.). Reading Mass: Addison-Wesley. ISBN 0-201-44124-1.

Early publications about structural induction include:

- Burstall, R.M. (1969). "Proving Properties of Programs by Structural Induction". *The Computer Journal* **12** (1): 41–48. doi:10.1093/comjnl/12.1.41 (http://dx.doi.org/10.1093%2Fcomjnl%2F12.1.41).
- Aubin, Raymond (1976), *Mechanizing Structural Induction* (http://hdl.handle.net/1842/6649), EDI-INF-PHD, 76-002, University of Edinburgh
- Huet, G.; Hullot, J.M. (1980). "Proofs by Induction in Equational Theories with Constructors". *21st Ann. Symp. on Foundations of Computer Science*. IEEE. pp. 96–107.

Categories: Logic in computer science | Mathematical logic | Mathematical proofs | Wellfoundedness | Mathematical induction | Graph theory

---