Lecture 20: Other flavors of induction

The principle of strong induction

The method of induction we have seen until now is actually a particular flavor known as *ordinary* induction.

We now discuss a second flavor known as *strong induction*. The method is a slight rearrangement of ordinary induction, and the structure for the proof is exactly the same. The only difference is that instead of assuming that $P(k)$ is true and proving $P(k+1)$ is true, you are free to assume that *all of* $P(1), P(2), \ldots, P(k)$ is true, and use any combination of these k predicates to prove that $P(k+1)$ is true.

The goals of ordinary and strong inductions are exactly the same. However, strong induction is useful when the truth of $P(k+1)$ does not follow directly from the truth of $P(k)$, but requires that $P(l)$ is true for *all* integers $l \leq k$.

Here is a more formal description. Let $P(n)$ be a predicate defined on the nonnegative integers. If:

- (Base case) $P(0)$ is true, and
- (Inductive step) $P(0) \wedge P(2) \wedge \ldots \wedge P(k) \implies P(k+1) \quad \forall k \geq 0$

then

- $P(n)$ is true for all nonnegative integers.

In terms of predicate logic, we can write the induction principle as the following rule of inference:

$$\frac{\begin{array}{c} P(0) \\ \forall k \geq 0,\ P(1) \wedge \ldots \wedge P(k) \implies P(k+1) \end{array}}{\therefore\ \forall m \geq 0, P(n)}$$

Ordinary and strong induction are very similar, conceptually. The difference lies in the assumptions you make, and how you use these assumptions. In strong induction, the assumptions are stronger, but that sometimes makes proving things easier than they would if we were only using ordinary induction.

Another point to keep in mind: often, it will be necessary to prove *multiple base cases* when doing a proof in strong induction.

Examples

Let us do a few examples that use strong induction.

We first prove the *Binary representation theorem*, which is stated as follows:

> Every positive integer n can be written as $n = b_r 2^r + b_{r-1} 2^{r-1} + \ldots + b_1 2 + b_0$.

Simply put, this states that every integer n (bigger than 0) can be written out in binary form $(b_r b_{r-1} \ldots b_1 b_0)$; for example, 14 can be written as (1110) in binary.

We prove this theorem by strong induction. Let $P(n)$ be the assertion that n has a binary representation.

(base cases) The base cases $P(1)$ and $P(2)$ are immediate since 1 is 1 in binary, and 2 is 10 in binary.

(strong induction hypothesis) Suppose that *all* of $P(1), P(2), .., P(k-1), P(k)$ are true.

(induction step) We need to prove that $P(k+1)$ is true. We have exactly two cases:

- $k+1$ is an even number. By definition $k+1 = 2p$ for some integer $p \leq k$. By the strong induction hypothesis, $p = b_r 2^r + b_{r-1} 2^{r-1} + \ldots + b_1 2 + b_0$. Therefore $k+1 = 2p = b_r 2^{r+1} + b_{r-1} 2^r + \ldots + b_1 2^2 + b_0 2$. Thus, $k+1 = (b_r b_{r-1} \ldots b_0 0)$ in binary.
- $k+1$ is an odd number. By definition $k+1 = 2p+1$ for some integer $p < k$. By the strong induction hypothesis, $p = b_r 2^r + b_{r-1} 2^{r-1} + \ldots + b_1 2 + b_0$. Therefore $k+1 = 2p+1 = b_r 2^{r+1} + b_{r-1} 2^r + \ldots + b_1 2^2 + b_0 2 + 1$. Thus, $k+1 = (b_r b_{r-1} \ldots b_0 1)$ in binary.

In either case, $k+1$ has a binary representation, and $P(k+1)$ is true.

Here is a second example. Suppose we wish to prove the following:

> Suppose we have two types of postage stamps worth 10 cents and 15 cents respectively. Then, we can exactly make any amount ≥ 10 cents that is a multiple of 5 cents.

In other words, any denomination that is a multiple of 5 cents can be represented using some combination of 10-cent and 15-cent postage stamps. Let $P(n)$ be the assertion that $5n$ cents can be realized in this manner. We need to show that $P(n)$ is true for all $n \geq 2$.

(base cases) The base cases $n = 2$, $n = 3$ are immediate since we can represent $10c$ and $15c$ using 1 postage stamp each.

(strong induction hypothesis) Assume that $P(2), P(3), .., P(k-1), P(k)$ is true.

(induction step). To show that $P(k+1)$ is true, we observe that $5(k+1) = 5(k-1) + 10$. But we know $P(k-1)$ is true due to the strong induction hypothesis, i.e., we can represent $5(k-1)$ using some particular combination of stamps. Therefore, we can represent $5(k+1)$ by adding one $10c$ stamp to that combination.

Therefore by strong induction, $P(n)$ is true for all $n \geq 2$.

The principle of structural induction

By now, it should be clear that induction and recursion go hand in hand. Induction is basically a proof technique in which proofs of a given statement are recursively built up from proofs of other statements.

Until now, we have assumed that the statement that we want to prove is a predicate of the form $P(n)$ for integers n. But actually, induction is not limited to statements involving integers, and can be used in far more general problems – in fact, it can be used for any setting where the predicate is defined

over any other data structure which can be defined recursively. This flavor of induction is called *structural induction*.

Here is an example to illustrate what we mean. We defined trees and binary trees before, but in terms of graph theory concepts –

- trees are connected graphs with no cycles;
- binary trees are those where nodes have at most two children.

However, one can also define trees *recursively* in a somewhat abstract manner. Here is the definition. A (non-empty) binary tree (say T) is either:

- (base case) a single node, r, called the "root" with zero pointers; or
- (recursion) a root node with pointer(s) (edge(s)) to either one or two *other* binary trees, L and R.

Note this is more of a "pure" CS/CPRE definition of trees! No graph theoretical concepts such as cycles, connectedness, or children are necessary.

Using this type of recursive definition, we can prove things about trees. For example, we have proved before (using graph theory) that

> Every binary tree T with $|T|$ nodes has $|T| - 1$ edges.

Here is the proof. We will use structural induction.

(base case) T is a single node r ($n = 1$) and it has zero edges ($n - 1 = 0$).

(induction hypothesis) Let us presume the binary tree T has pointers to two other binary trees L and R with l and r nodes each. (The case where it points to only one other tree is similarly handled). We make the hypothesis that L and R satisfy the given assertion, i.e., L has $|L| - 1$ edges and R has $|R| - 1$ edges.

(induction step) We now prove that T has $|T| - 1$ edges. By the recursive definition of binary trees, the number of nodes in T is given by the number of nodes in L + the number of nodes in R + 1 (the root). In symbols,

$$|T| = |L| + |R| + 1.$$

Rerranging, we get:

$$|T| - 1 = |L| - 1 + |R| - 1 + 2.$$

This may look a bit funny to write down, but the right hand side has a specific meaning. The edges in T comprise the edges in L plus the edges in R, and 2 new points. By induction hypothesis, the number of edges in L is exactly $|L| - 1$ and the number of edges in R is $|R| - 1$. Therefore the right hand side is precisely the total number of *edges* in T, which according to the above equation, is $|T| - 1$. Done!

Spot the bug

Last point about induction. Can you spot the conceptual bug in the following (bogus) proof?

Recall the definition of Fibonacci numbers: $F_0 = 0, F_1 = 1$, and for $n \geq 2$, we have

$$F_n = F_{n-1} + F_{n-2}.$$

Claim: All Fibonacci numbers are even.

(Bogus) Proof:

Let $P(n)$ be the assertion that the n^{th} Fibonacci number is even.

(Base case) For $n = 0$, $F_n = 0$, which means that the assertion $P(0)$ is true.

(Strong induction hypothesis) Let the assertion be true for all $n = 0$ up to $n = k$, i.e., all Fibonacci numbers up to F_k are even.

(Induction step) We prove the statement for $n = k + 1$. Consider F_{k+1}. By definition $F_{k+1} = F_k + F_{k-1}$. But by the strong induction hypothesis, both F_k and F_{k-1} are even. Therefore, the assertion is true for $n = k + 1$. Therefore, by strong induction, the statement is true for all n. Done!

Obviously the is incorrect (since) we have both even and odd Fibonacci numbers.

The base case (as stated) is true.

The error lies in the mismatch between the problem statement and the strong induction hypothesis. While stating the hypothesis, we have assumed that all of $P(0), P(1), P(2), \ldots$ is true. However, this is incorrect! $P(1)$ cannot be true, since F_1 is equal to 1, which is odd, not even. So make sure the hypothesis that you make in an induction proof actually matches what is given in the problem.