CS 70 Fall 2003 — Discrete Mathematics for CS — Wagner

Lecture 3

This lecture covers further variants of induction, including strong induction and the closely related well-ordering axiom. We then apply these techniques to prove properties of simple recursive programs.

Strong induction

Axiom 3.1 (Strong Induction): For any property P,
if $P(0)$ and $\forall n \in \mathbf{N}\ (P(0) \wedge P(1) \wedge \ldots \wedge P(n) \implies P(n+1))$,
then $\forall n \in \mathbf{N}\ P(n)$.

This says that if all the following sentences are true:

$$\begin{array}{rcl} & & P(0) \\ P(0) & \implies & P(1) \\ P(0) \wedge P(1) & \implies & P(2) \\ P(0) \wedge P(1) \wedge P(2) & \implies & P(3) \\ P(0) \wedge P(1) \wedge P(2) \wedge P(3) & \implies & P(4) \end{array}$$

and so on, then $P(n)$ must be true for all n. Intuitively, this seems quite reasonable. If the truth of P all the way up to n always implies the truth of $P(n+1)$, then we immediately obtain the truth of P all the way up to $n+1$, which implies the truth of $P(n+2)$, and so on *ad infinitum*.

If we compare the Strong Induction axiom to the original Induction axiom from Lecture 2, we see that Strong Induction appears to make it *easier* to prove things. With simple induction, one must prove $P(n+1)$ given the inductive hypothesis $P(n)$; with strong induction one gets to assume the inductive hypothesis $P(0) \wedge P(1) \wedge \ldots \wedge P(n)$, which is much stronger.

Consider the following example, which is one half of the Fundamental Theorem of Arithmetic. (The other half says that the product is unique.)

Theorem 3.1: *Any natural number $n > 1$ can be written as a product of primes.*

To prove this, of course, we need to define prime numbers:

Definition 3.1 (Prime): A natural number $n > 1$ is prime iff it has exactly two factors (1 and n). 1 itself is not prime.

Let's see first what happens when we try a simple induction:

Proof: (Attempt 1) The proof is by induction over the natural numbers $n > 1$.

- Base case: prove $P(2)$.
  $P(2)$ is the proposition that 2 can be written as a product of primes. This is true, since 2 can be written as the product of one prime, itself. (Remember that 1 is not prime!)

- Inductive step: prove $P(n) \implies P(n+1)$ for all natural numbers $n > 1$.

1. The inductive hypothesis states that n can be written as a product of primes.
2. To prove: $n+1$ can be written as a product of primes.
3. We're stuck: given $P(n)$, we could easily establish $P(2n)$ or $P(7n)$, but $P(n+1)$ is unconnected to $P(n)$.

□

With a strong induction, we can make the connection between $P(n+1)$ and earlier facts in the sequence that *are* relevant. For example, if $n+1=72$, then $P(36)$ and $P(24)$ are useful facts.

Proof: The proof is by strong induction over the natural numbers $n > 1$.

- Base case: prove $P(2)$, as above.
- Inductive step: prove $P(2) \wedge \ldots \wedge P(n) \implies P(n+1)$ for all natural numbers $n > 1$.
  1. The inductive hypothesis states that, for all natural numbers m from 2 to n, m can be written as a product of primes.
  2. To prove: $n+1$ can be written as a product of primes.
  3. Proof by cases:
     - $n+1$ is prime: then $n+1$ can be written as the product of one prime, itself.
     - $n+1$ is not prime: then by the definition of prime numbers, there exist integers a, b such that $2 \leq a,b < n+1$ and $n+1=a \cdot b$. By the inductive hypothesis, both a and b can be written as a product of primes. Hence $n+1$ can be written as a product of primes.

□

Consider the following example, which is of immense interest to post offices and their customers:

Theorem 3.2: *Any integer amount of postage from* 8¢ *upwards can be composed from* 3¢ *and* 5¢ *stamps.*

With a strong induction, we can make the connection between $P(n+1)$ and earlier facts in the sequence. In particular, $P(n-2)$ is relevant because $n+1$ can be composed from the solution for $n-2$ plus one 3¢ stamp. So the inductive step works if $P(n-2)$ is known already. This will not be the case when $n+1$ is 9 or 10, so we will need to handle these separately.

Proof: The proof is by strong induction over the natural numbers $n \geq 8$.

- Base case: prove $P(8)$.
  $P(8)$ is the proposition that 8¢ of postage can be composed from 3¢ and 5¢ stamps. This is true, requiring 1 of each.
- Inductive step: prove $P(8) \wedge \ldots \wedge P(n) \implies P(n+1)$ for all natural numbers $n \geq 8$.
  1. The inductive hypothesis states that, for all natural numbers m from 8 to n, m¢ of postage can be composed from 3¢ and 5¢ stamps.
  2. To prove: $(n+1)$¢ of postage can be composed from 3¢ and 5¢ stamps.
  3. The cases where $n+1$ is 9 or 10 must be proved separately. 9¢ can be composed from three 3¢ stamps. 10¢ can be composed from two 5¢ stamps.
  4. For all natural numbers $n+1 > 10$, the inductive hypothesis entails the proposition $P(n-2)$. If $(n-2)$¢ can be composed from 3¢ and 5¢ stamps, then $(n+1)$¢ can be composed from 3¢ and 5¢ stamps simply by adding one more 3¢ stamp.

$\square$

Notice that, as with the tiling problem, the inductive proof leads directly to a simple recursive algorithm for selecting a combination of stamps.

Notice also that a strong induction proof may require several "special case" proofs to establish a solid foundation for the sequence of inductive steps. It is easy to overlook one or more of these.

Simple induction and strong induction

We have seen that strong induction makes certain proofs easy even when simple induction appears to fail. A natural question to ask is whether the strong induction axiom is in fact *logically stronger* than the simple induction axiom; if so, then the theorems that can be proved using strong induction are a strict superset of the theorems that can be proved using simple induction.

Let's investigate this question. First, does the strong induction entail the simple induction axiom? Intuitively, this seems to be true. Let's put the two axioms side by side and examine their structure (we'll take the restriction to the natural numbers as implicit here):

$$\begin{array}{rll} \text{Simple:} & P(0) \wedge [\forall n\, P(n) \implies P(n+1)] & \implies \forall n\, P(n) \\ \text{Strong:} & P(0) \wedge [\forall n\, P(0) \wedge \ldots \wedge P(n) \implies P(n+1)] & \implies \forall n\, P(n) \end{array}$$

We can reduce this to the following basic form (with the obvious definitions for propositions A, B, B', and C):

$$\begin{array}{rll} \text{Simple:} & A \wedge B & \implies C \\ \text{Strong:} & A \wedge B' & \implies C \end{array}$$

Now if $P(n) \implies P(n+1)$, then $P(0) \wedge \ldots \wedge P(n) \implies P(n+1)$. Hence, $B \implies B'$ (i.e., b is stronger than B'). Hence, if $A \wedge B'$ suffice to prove C, then surely the stronger fact $A \wedge B$ also suffices to prove C. (This is easily checked using truth tables.) Therefore, the strong induction axiom entails the simple induction axiom.

Second, does the simple induction entail the strong induction axiom? One might expect not, but in fact it does! We can see this by defining, for any property $P(n)$, the proposition

$$Q(n) \Leftrightarrow P(0) \wedge \ldots \wedge P(n)$$

That is, $Q(n)$ is the property "P holds from 0 to n." The idea is that simple induction using Q is in fact identical to strong induction using P. The simple induction axiom for Q is

$$Q(0) \wedge [\forall n\, Q(n) \implies Q(n+1)] \implies \forall n\, Q(n)$$

Expanding out the definition of Q, we obtain

$$P(0) \wedge [\forall n\, (P(0) \wedge \ldots \wedge P(n)) \implies (P(0) \wedge \ldots \wedge P(n) \wedge P(n+1))] \implies [\forall n\, (P(0) \wedge \ldots \wedge P(n))]$$

A few moments' thought [we recommend thinking this thought yourself] reveals that this proposition is logically equivalent to the proposition

$$P(0) \wedge [\forall n\, P(0) \wedge \ldots \wedge P(n) \implies P(n+1)] \implies \forall n\, P(n)$$

which is the strong induction axiom. Therefore we have shown (rather informally perhaps) the following:

Theorem 3.3: *The strong induction axiom and the simple induction axiom are logically equivalent.*

Why have two different forms of induction then? The point is that strong induction reminds its user of the opportunity to use $P(0) \wedge \ldots \wedge P(n)$ in the inductive step when $P(n)$ is defined the "natural" way from the statement of the theorem to be proved.

The well-ordering principle

If one thinks about why induction works, one might ask the question "How could the induction axiom fail to be true?" To violate the induction axiom, we would need to satisfy its antecedent (so $P(0)$ is true and $P(n) \implies P(n+1)$ for all n) while violating its consequent (so $\exists n \neg P(n)$). Let us consider the first n for which $P(n)$ is false. By definition, we know that $P(n-1)$ is true; and by assumption we know that $P(n-1) \implies P(n)$; therefore we have a straightforward contradiction!

Have we *proved* the induction axiom? Actually, no; we have proved that the induction axiom follows from another axiom, which was used implicitly in defining "the first n for which $P(n)$ is false."

Axiom 3.2 (Well-Ordering): Every nonempty set of natural numbers has a smallest element.

Duh. Doesn't every nonempty set of orderable elements have a smallest element? No! Every *finite* set has a smallest element, but not every *infinite* set. For example, neither the integers nor even the positive rationals have a smallest element.

The well-ordering principle not only underlies the induction axioms, but also has direct uses in its own right. A particularly elegant example concerns the existence of cycles in tournaments.

Definition 3.2 (Round-Robin): A round-robin tournament is one in which each player p plays each other player q exactly once and either wins ($p \succ q$) or loses ($q \succ p$).

Definition 3.3 (Cycle): A cycle in a tournament is a set of players $\{p_1 \ldots p_k\}$ such that $p_1 \succ p_2 \succ \ldots \succ p_{k-1} \succ p_k \succ p_1$.

Theorem 3.4: *In every round-robin tournament, if there is a cycle, then there is a cycle of length 3.*

Proof: The proof is by contradiction.

1. Assume the theorem is false. Consider the set of cycle lengths of the tournament. By assumption, this must be nonempty.
2. By the well-ordering principle, it must have a smallest element k. By assumption, $k > 3$.
3. Let the first three elements in this cycle be p_1, p_2, p_3, and consider the result of the match between p_1 and p_3.
4. Case 1: $p_1 \succ p_3$. Then we have $p_1 \succ p_3 \succ \ldots \succ p_{k-1} \succ p_k \succ p_1$, i.e., a cycle of length $k-1$, contradicting our assumption that the smallest cycle has length $k > 3$.
5. Case 2: $p_3 \succ p_1$. Then we have $p_1 \succ p_2 \succ p_3 \succ p_1$, i.e., a cycle of length 3, contradicting our assumption that the smallest cycle has length $k > 3$.
6. By the definition of round-robins, either $p_1 \succ p_3$ or $p_3 \succ p_1$. Therefore, a contradiction exists.
7. Hence, it must be the case that any tournament with a cycle has a cycle of length 3.

□

This proof illustrates a common way to use well-ordering combined with proof by contradiction. The well-ordering principle allows one to focus on a concrete counterexample with the property that every smaller example satisfies some property. For certain proofs, this can be an easier thought process than induction.

Induction and recursion

There is an intimate connection between induction and recursion. Essentially every recursive function relies for its correctness on an inductive proof. Remember that a recursive function applies itself to a "smaller" argument. The inductive proof says that if the recursive function works on all smaller arguments it will work on the current argument.

We'll begin with that old favourite, the factorial function. Let's give a recursive definition for a function $f(n)$ and show it's identical to $n!$. For any $n \in \mathbf{N}$,

$$\begin{aligned} f(n) &= 1 \text{ if } n = 0 \\ f(n) &= n f(n-1) \text{ otherwise} \end{aligned}$$

Theorem 3.5: *For all natural numbers n,* $f(n) = n!$.

Proof: The proof is by induction over the natural numbers. Let $P(n)$ be the proposition that $f(n) = n!$.

- Base case: prove $P(0)$.
  $P(0)$ is the proposition that $f(0) = 0!$. By the definition above, $f(0) = 1 = 0!$, hence $P(0)$ is true.
- Inductive step: prove $P(n) \implies P(n+1)$ for all $n \in \mathbf{N}$.
  1. The inductive hypothesis is $f(n) = n!$.
  2. To prove: $f(n+1) = (n+1)!$.
  3. By the definition above,

$$\begin{aligned} f(n+1) &= (n+1)\cdot f(n) \text{ because } n \in \mathbf{N} \text{ so } (n+1) \neq 0 \\ &= (n+1)\cdot n! \text{ by the inductive hypothesis} \\ &= (n+1)\cdot n \cdot (n-1) \cdots 1 = (n+1)! \end{aligned}$$

Hence, by the induction principle, $\forall n \in \mathbf{N}\ f(n) = n!$. $\Box$

Mathematical functions and real programs

The above discussion applies to a purely mathematical definition of $f(n)$. If we wanted to reason about a real program, first we have to write it in a real language, such as Scheme:

```
(define (factorial n)
  (if (= n 0)
      1
    (* n (factorial (- n 1)))))
```

The statement of correctness for a program is not quite so straightforward as for a mathematically defined function:

Theorem 3.6: *For all (computer representations of) natural numbers n, the result of evaluating the expression* `(factorial` n`)` *is the computer representation of* $n!$.

The proof that the program is correct is very similar to the proof that the recursive function has the right values. Notice the notation: actual syntactic elements of the programming language are in typewriter font, while variables that range over them are in italics.

Proof: The proof is by induction over the natural numbers. Let $P(n)$ be the proposition that `(factorial` n`)` $= n!$.

- Base case: prove $P(0)$.
  $P(0)$ is the proposition `(factorial 0)` $= 0!$. By the definition above,

  `(factorial 0)`
  $=$ `(if (= 0 0) 1 (* 0 (factorial (- 0 1))))`
  $=$ 1 by evaluation of `if`

- Inductive step: prove $P(n) \implies P(n+1)$ for all $n \in \mathbf{N}$.
  1. The inductive hypothesis is `(factorial` n`)` $= n!$.
  2. To prove: `(factorial` $(n+1)$`)` $= (n+1)!$.
  3. By the definition above,

     `(factorial` $(n+1)$`)`
     $=$ `(if (=` $(n+1)$ `0) 1 (*` $(n+1)$ `(factorial (-` $(n+1)$ `1))))`
     $=$ `(*` $(n+1)$ `(factorial (-` $(n+1)$ `1)))` because $n \in \mathbf{N}$ so $(n+1) \neq 0$
     $=$ `(*` $(n+1)$ `(factorial` n`))`
     $=$ `(*` $(n+1)$ $n!$`)` by the inductive hypothesis
     $=$ $(n+1)!$

Hence, by the induction principle, $\forall n \in \mathbf{N}$ `(factorial` n`)` $= n!$ □

Notes on this proof:

- We appeal implicitly to several aspects of the evaluation of programs such as the binding of parameters, the definition of `if`-expressions, and the correspondence between the mathematical function "$-$" and the built-in function "`-`". These lemmata are an essential part of the definition of the programming language and can be stated and proved once and for all.

- The theorem as stated is almost certainly false! A *real* proof of correctness, for a suitably reduced theorem, must handle the important differences between mathematical entities and the corresponding entities in the computer. For example, $n!$ is well-defined for any natural number, but `(factorial n)` fails if `n` is large enough to cause an overflow in integer multiplication. Another way to say this is that `*` is not the same as the mathematical multiplication function.

- As defined, `(factorial n)` causes an error for nonnumeric, noninteger, or negative inputs. (What error arises from negative inputs?) A full specification for a really robust system should lay out the correct responses to all possible inputs.

For the most part, we will use mathematical rather than Scheme definitions because it makes the proofs typographically cleaner and the theorems true.