CS/Math 240: Introduction to Discrete Mathematics 2/15/2011

Lecture 8 : Structural Induction

Instructor: Dieter van Melkebeek Scribe: Dalibor Zelený

DRAFT

Last week we discussed proofs by induction. We use induction to prove statements of the form $(\forall n)P(n)$. This is done in two steps. First, we prove the base case $P(0)$. Afterwards, as the inductive step, we show that the implication $P(n) \Rightarrow P(n+1)$ holds for all n. We also saw some modifications of this proof technique.

This week we will see some applications of proofs by induction. Today we look at structural induction and invariants. The study of invariants will lead to the topic of program correctness.

8.1 Inductive Definitions and Structural Induction

We can define some concepts more precisely using an *inductive definition*. This is useful for concepts whose instances are built from a few elementary building blocks. An inductive definition of a concept consists of two parts.

- A *foundation rule* which says what are the simplest instances of the concept being defined.
- A *constructor rule* which says how to combine simpler instances of the concept being defined into a more complex instance.

For example, we can use an inductive definition to define propositional formulas.

Definition 8.1 (Propositional formula)**.** *Every propositional variable is a propositional formula. If F_1 and F_2 are propositional formulas, then so are $\neg F_1$, $F_1 \wedge F_2$, $F_1 \vee F_2$, $F_1 \Rightarrow F_2$, and $F_1 \iff F_2$. Nothing else is a propositional formula.*

In Definition 8.1, the first sentence is the foundation rule and the second sentence is the constructor rule.

We can exploit the structure of an inductive definition such as Definition 8.1 using *structural induction*. In a proof by structural induction, we prove that some property holds for all instances by induction on the number of times we use the constructor rule. This works because every instance is is a result of some number of applications of the constructor rule.

Note that there is a direct correspondence between the two parts of an inductive definition and the two parts of an inductive proof. To prove some property holds for the elementary instances given in the foundation rule, we prove the base case for each elementary instance. (Note that there can be multiple base cases.) To prove the inductive step, we exploit the constructor rule.

Theorem 8.2 is an example of a statement we can prove using structural induction. The proof exploits the inductive definition of propositional formulas.

Theorem 8.2. *In every propositional formula, the number of variable occurrences is one more than the number of occurrences of binary operators.*

In order to use induction, we need a predicate P that depends on a natural number n, so that the statement of Theorem 8.2 has the form $(\forall n)P(n)$. We use $P(n)$: "In every propositional formula that can be built using n applications of the constructor rule, the number of variable occurrences is one more than the number of occurrences of binary operators." Now that's a rather long statement and we will probably use it a lot. In particular, we will talk about numbers of variables and binary operators, so let's develop some notation for those. Let vars(F) be the number of variable occurrences in a propositional formula F, and let binops(F) be the number of occurrences of binary operators in F. Then $P(n)$ becomes "In every propositional formula that can be built using n applications of the constructor rule, vars(F) = binops(F) + 1".

In practice, you will probably not develop all notation right away. Instead, you may find that your first draft of a proof is too verbose, and choose to develop notation in response.

Finally, we remark that since we can only build a propositional formula using our inductive definition, proving the statement $(\forall n)P(n)$ proves the theorem. We use strong induction in the proof that follows. We could also use regular induction to prove Theorem 8.2.

Proof of Theorem 8.2. We give a proof by structural induction.

We show $(\forall n)P(n)$ where $P(n)$ says that "for every propositional formula F built using n applications of the constructor rule, vars(F) = binops(F) + 1".

For the base case, we consider a formula built using the foundation rule. Such formula consists of a single variable and no binary operators. This proves the base case.

In the inductive step, we prove $(\forall n \geq 0) \bigwedge_{m=0}^{n} P(m) \Rightarrow P(n)$. Consider a formula F built using $n+1$ applications of the constructor rule. Then there are 5 cases depending on what operator was used to make F.

Case 1: F is of the form $\neg F_1$. Since F is made using $n+1$ applications of the constructor rule, F_1 can be made using n applications of the constructor rule, and the induction hypothesis implies that

$$\text{vars}(\text{F}_1) = \text{binops}(\text{F}_1) + 1. \tag{8.1}$$

Note that negating F_1 doesn't add any additional variables. Also, since negation is a unary operator, negating F_1 doesn't produce any additional binary operators. Hence, vars(F) = vars(F_1), binops(F) = binops(F_1), and substituting those two equalities into (8.1) yields vars(F) = binops(F)+ 1.

Case 2: F is of the form $F_1 \wedge F_2$. If $F_1 \wedge F_2$ is built using $n+1$ applications of the constructor rule, then F_1 and F_2 are built using at most n applications of the constructor rule each. By the induction hypothesis, we have

$$\text{vars}(\text{F}_1) = \text{binops}(\text{F}_1) + 1 \tag{8.2}$$

$$\text{vars}(\text{F}_2) = \text{binops}(\text{F}_2) + 1 \tag{8.3}$$

By construction of F, we also have the following two equalities.

$$\text{vars}(\text{F}) = \text{vars}(\text{F}_1) + \text{vars}(\text{F}_2) \tag{8.4}$$

$$\text{binops}(\text{F}) = \text{binops}(\text{F}_1) + \text{binops}(\text{F}_2) + 1 \tag{8.5}$$

where the additional 1 in (8.5) comes from the $\wedge$ operator used to combine F_1 and F_2 into F.

Substituting (8.2) and (8.3) into (8.4) yields

$$\text{vars}(\text{F}) = (\text{binops}(\text{F}_1) + 1 + \text{binops}(\text{F}_2)) + 1. \tag{8.6}$$

Note that the three terms in parentheses in (8.6) are equal to the right-hand side of (8.5), so we get vars(F) = binops(F) + 1, which is what we wanted to show.

The remaining three cases for the operators $\vee$, $\Rightarrow$ and $\iff$ have exactly the same proof as Case 2, so we omit them.

This completes the proof of the inductive step, and so the theorem is proved. □

Another example of an inductive definition is the Fibonacci sequence. One common example of the use of the Fibonacci sequence is in the following simple model of a rabbit colony. First month, we have one newborn male-female pair. Starting from the second month of their life, male-female pairs mate and have one male and one female offspring at the end of every month. We want to know the population of the rabbit colony as a function of the number of months.

Let F_i be the number of rabbit pairs in month i. Let's look at the first few months and find out how large the rabbit population is each month.

Month 1: $F_1 = 1$ because we have one new pair of rabbits.

Month 2: Our first pair starts mating, but no new rabbits are born yet, so $F_2 = 1$.

Month 3: At the end of the second month, our only pair of rabbits produces its first children, so $F_3 = F_2 + 1 = 2$.

Month 4: This new pair of rabbits isn't mating in the third month, but the first one is and produces another pair of offspring at the end of the third month. Thus, $F_4 = F_3 + 1 = 3$.

Month 5: Our original pair of rabbits and its first children were mating in the fourth month, so $F_5 = F_4 + 2 = 5$.

We observe the pattern of the first five months and notice that all pairs of rabbits that lived in month $n-1$ still live in month n. In addition, all pairs of rabbits that lived in month $n-2$ have offspring at the end of month $n-1$, and those offspring also contribute to the population in month n. Thus,

$$F_n = F_{n-1} + F_{n-2}, \qquad n \geq 3. \tag{8.7}$$

Equation (8.7) is an example of a *recurrence*. A recurrence defines one term of a sequence using previous terms. In the case of (8.7), we define the n-th term in terms of the $(n-1)$st term and the $(n-2)$nd term. This looks like the constructor rule in an inductive definition. Moreover, the first and second term are defined separately to be 1, which looks like the fondation rule in an inductive definition.

We just saw that recurrences are inductive definitions in disguise, so it makes sense to try proving properties of sequences defined using recurrences by structural induction. To prove the base case, we prove the property for (possibly multiple) terms at the beginning of the sequence. In the proof of the inductive step, we assume that the property holds for the first n terms and use the recurrence to show that the property holds for the $(n+1)$st term too.

Here is an example of a property of the Fibonacci sequence we can prove using structural induction. We leave the proof to the reader.

Exercise 8.1: Prove that

$$F_n = \frac{\varrho^n + (1-\varrho)^n}{\sqrt{5}}, \quad \text{where} \quad \varrho = \frac{1+\sqrt{5}}{2}. \tag{8.8}$$

The number ϱ from (8.8) is known as the *golden ratio*, and appears often in nature.

Observe that when n is large, the term $(1-\varrho)^n$ in (8.8) becomes negligible. To see that, we approximate $\varrho \approx 1.62$ and $1 - \varrho \approx -0.62$ (The symbol $\approx$ means "approximately equal to".), and

observe that raising ϱ to power n produces a giant number for a large n, whereas raising $1-\varrho$ to power n produces a number that is close to zero for a large n. Thus, for large n, we can omit $(1-\varrho)^n$ from (8.8) and still get a very good estimate on the size of the rabbit population in month n. We approximate $F_n \approx \varrho^n$, which tells us that the population grows roughly by a factor of the golden ratio every month.

8.2 Invariants

We look at invariants when we analyze systems that evolve in discrete time steps. An *invariant* is a property of the system that holds at every time step. Usually, the state of the system in the next time step $t+1$ only depends on the state in the current time step t. This suggests we can use induction on the time step t to prove that a property is an invariant. In particular, we can show that

1. The system has the property in the first time step $t = 0$.
2. If the system has the property at time t, it has it at time $t+1$ as well.

The predicate $P(t)$ for the inductive proof would say "The system has the property at time t."

8.2.1 Finding Invariants

It turns out that finding invariants is not an easy problem in general. Let's start with an example where finding an invariant is not that hard.

Suppose we have a robot which walks on a 2-dimensional grid. The rows and columns of the grid are labeled by integers. Our robot starts at position $(0, 0)$, and can only move diagonally, one square at a time. We show all possible moves the robot can make from square (x, y) in Figure 8.1.

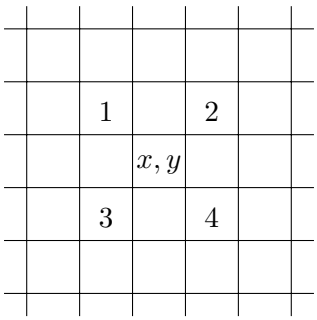


Figure 8.1: The robot can move from square (x, y) to the squares labeled by numbers 1 through 4.

Can the robot get to position $(8, 9)$? Intuitively, the answer is no. If it were able to get to $(8, 9)$, it could also get to $(7, 8)$. From there, it could get to $(6, 7)$, then to $(5, 6)$, and eventually it would end up at $(0, 1)$. But that seems impossible because it can't go to $(0, 1)$ in one step from $(0, 0)$.

The intuitive reasoning from the previous paragraph is a start, but by no means a proof that the robot can't get to $(8, 9)$. We argue that the answer is "no" rigorously by defining an invariant and proving it by induction. Our "system" is just our robot, and the "state" of the system is the robot's position. We represent the robot's position at time t with a pair of integers, (x_t, y_t).

After trying a few moves and writing down the positions after those moves, we observe that whenever the robot is at position (x, y), either both x and y are odd, or they are both even. In other words, $x+y$ is always even. Then we write our invariant as $P(t)$: "$x_t + y_t$ is even", and prove the following theorem.

Theorem 8.3. *At every time step t, $x_t + y_t$ is even.*

Proof. We prove by induction on the time step t that the invariant $P(t)$: "$x_t + y_t$ is even" holds at all time steps $t \geq 0$.

In the base case, $t = 0$, the robot is at position $(0, 0)$. Since $0 + 0 = 0$ is even, the base case holds.

For the inductive step, we need to argue $(\forall t \geq 0)P(t) \Rightarrow P(t + 1)$. If $P(t)$ holds, then at time t, $x_t + y_t$ is even. We argue by cases. There is one case for each possible move the robot can make from (x_t, y_t). The possible destinations are $(x_t - 1, y_t + 1)$, $(x_t + 1, y_t + 1)$, $(x_t - 1, y_t - 1)$ and $(x_t + 1, y_t - 1)$. (The possible destinations correspond to the four squares labeled by numbers 1 through 4 in Figure 8.1.)

Case 1: The robot moves to $(x_{t+1}, y_{t+1}) = (x_t - 1, y_t + 1)$. Then $x_{t+1} + y_{t+1} = x_t - 1 + y_t + 1 = x_t + y_t$, which is even by the induction hypothesis.

Case 2: The robot moves to $(x_{t+1}, y_{t+1}) = (x_t + 1, y_t + 1)$. Then $x_{t+1} + y_{t+1} = x_t + 1 + y_t + 1 = x_t + y_t + 2$. Now $x_t + y_t$ is even by the induction hypothesis, and 2 is also even, so $x_{t+1} + y_{t+1}$ is even.

Case 3: The robot moves to $(x_{t+1}, y_{t+1}) = (x_t - 1, y_t - 1)$. Then $x_{t+1} + y_{t+1} = x_t - 1 + y_t - 1 = x_t + y_t - 2$. Now $x_t + y_t$ is even by the induction hypothesis, and -2 is also even, so $x_{t+1} + y_{t+1}$ is even.

Case 4: The robot moves to $(x_{t+1}, y_{t+1}) = (x_t + 1, y_t - 1)$. Then $x_{t+1} + y_{t+1} = x_t + 1 + y_t - 1 = x_t + y_t$, which is even by the induction hypothesis.

Since the four cases above are the only possibilities, we have proved the inductive step, and thus also the theorem. □

Note that $8 + 9 = 17$ is odd, so Theorem 8.3 implies that the robot cannot reach position $(8, 9)$.

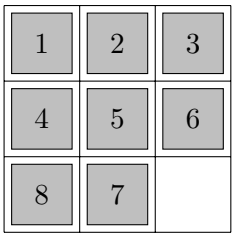
In fact, we can show that a position (x, y) can be reached if and only if the sum of the coordinates, $x + y$, is even. At Theorem 8.3, we argued the implication "if (x, y) can be reached then $x + y$ is even." To argue the other implication, that is, to argue that if $x + y$ is even, then the robot can reach (x, y), we would demonstrate a sequence of moves that takes the robot from $(0, 0)$ to (x, y). We can do this by induction, and leave the proof as an exercise to the reader.

We conclude this section with a remark. In the last example, our invariant completely describes the set of squares our robot can reach. As systems get more complicated, the invariants we come up with become complicated as well, and often fail to describe the system's behavior completely. But we may be lucky enough and come up with invariants that are sufficient for our purposes, that is, although they don't fully characterize the system, they describe the characteristics of the system we actually care about.

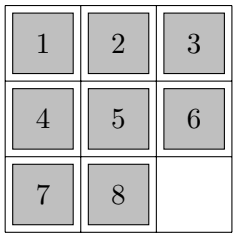
8.2.2 A More Complicated Invariant

Not all invariants are as simple. In the next example, it takes more work to find an invariant.

Consider the following puzzle, which we'll call the 8-puzzle. This is a simplification of a puzzle you probably saw when you were little. You have a 3×3 grid. Eight of the squares are filled with tiles numbered 1 through 8 as shown in Figure 8.2a, and the lower right square is blank. We can move a tile to the blank square from a neighboring square, which swaps the tile we move with the blank square. In Figure 8.2a, the only valid moves are swapping the tile labeled 6 with the blank square and swapping the tile labeled 7 with the blank square. Our goal is to rearrange the tiles so that the numbers are "in order" and the bottom right corner is blank. This is shown in Figure 8.2b. Of course, we are only allowed to make valid moves.



(a) Starting arrangement.



(b) Desired arrangement.

Figure 8.2: Two arrangements in the 8-puzzle.

It turns out that no sequence of valid moves can transform the arrangement from Figure 8.2a to the arrangement in Figure 8.2b. In the following paragraphs, we illustrate the thought process that leads us to this conclusion.

Consider the sequence formed by listing the labels on the tiles row by row, and listing tiles in a row from left to right. We also have a term, $\square$, for the blank square. For example, the sequences that correspond to the arrangements in Figure 8.2a and Figure 8.2b are $1, 2, 3, 4, 5, 6, 8, 7, \square$ and $1, 2, 3, 4, 5, 6, 7, 8, \square$, respectively.

We say that a pair of positions (i, j) in our sequence is *in error* if $i < j$, there are integers at both positions, but the integer at position i is greater than the integer at position j. For example, the sequence corresponding to the arrangement in Figure 8.2a has exactly one pair of possitions in error, namely $(7, 8)$, and the sequence corresponding to the arrangement in Figure 8.2b has no pair of positions in error.

Let's see what happens to the sequence corresponding to an arrangement as we perform certain moves. In particular, we investigate how the number of pairs in error changes after a move. There are two kinds of moves we consider:

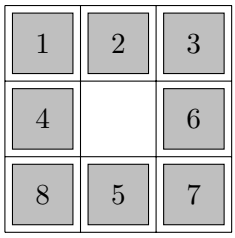
1. Row moves where the tile we move and the blank square are in the same row.

2. Column moves where the tile we move and the blank square are in the same column.

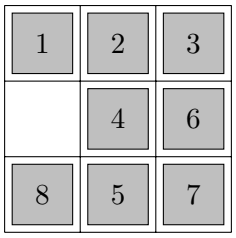
Consider how a move changes the sequence. Any move swaps the position of one tile with the blank square. Suppose the blank square is swapped with a tile labeled x. Then the positions of x and $\square$ switch after the move, and terms in all other positions in the sequence remain the same. Hence, a move of tile x can change the relative order of the integer x with other integer terms of our sequence, but can't change the relative order of two integer terms that correspond to tiles which stay put.

First consider a row move of tile x. By the way we constructed our sequence, it follows that x and $\square$ are adjacent terms in the sequence. If y is before x and $\square$ in the sequence, it will remain before x and $\square$ after the move because x and $\square$ just switch their positions in the sequence. Similarly, if y is after both x and $\square$ in the sequence, it will remain after x and $\square$ after the move. Since x and $\square$ are adjacent in the sequence, all other terms are either before them or after them, so we have shown that the relative order of no two integers changes after a row move. Hence, a row move of tile x causes no change in the number of pairs that are in error.

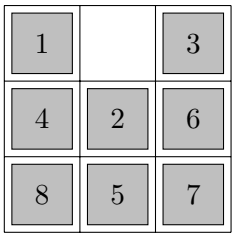
For example, the sequence representing the arrangement in Figure 8.3a is $1, 2, 3, 4, \square, 6, 8, 5, 7$, and the sequence for the arrangement in Figure 8.3b is $1, 2, 3, \square, 4, 6, 8, 5, 7$. As we can see, the only two terms in the sequence that changed their relative order correspond to the blank square and the tile that was moved. The relative order of no two integer terms changed.



(a) An arrangement obtained from the on in Figure 8.2a by moving tile 7 and then tile 5



(b) An arrangement obtained from the on in Figure 8.3a by moving tile 4. The move is a row move.



(c) An arrangement obtained from the on in Figure 8.3a by moving tile 2. The move is a column move.

Figure 8.3: Row moves and column moves in an 8-puzzle.

Now consider a column move of tile x. By the way we constructed our sequence, there are exactly two terms, say a and b, between x and $\square$, and the sequence before the move looks like one of the rows in Figure 8.4 below. A column move takes us from one row to the other one (whether from the top one to the bottom one or the other way around depends on the relative position of tile x and the blank square before the move). Thus, a column move only changes the relative order of x and a and the relative order of x and b.

$$\begin{array}{cccccc} \cdots & x & a & b & \square & \cdots \\ \cdots & \square & a & b & x & \cdots \end{array}$$

Figure 8.4: Two possibilities for the sequence before a column move of tile x. The dots indicate there are other terms before and after x, a, b and $\square$ in the sequence, and note that the relative order of x and those terms cannot change.

Suppose the sequence before the column move has the form of row 1 of Figure 8.4. Then if $x < a$, the pair (old position of x, position of a) was not in error, and also is not in error after the move. On the other hand, the pair (position of a, new position of x) was not in error before the move, but is in error after the move. So the move has increased the number of pairs involving a's position that are in error by one.

On the other hand, if $x > a$, the pair (old position of x, posotion of a) is in error, and the error goes away after the move. The pair (position of a, new position of x) was not in error before the move, and also isn't in error after the move. Thus, in this case, the move decreases the number of pairs involving a's position that are in error by one.

In either case, the number of pairs involving a's position that are in error changes by one. We can make the same argument about the number of pairs involving b's position. As we argued earlier, only pairs involving the positions of a, b and x can change their error states, so the net change in the number of pairs that change their error state is either -2, 0, or 2.

We can argue in a similar fashion if the sequence before the column move has the form of row 2 of Figure 8.4, which means that any column move changes the number of pairs in error by -2, 0, or 2.

For example, the sequence for the arrangement in Figure 8.3c is $1, \square, 3, 4, 2, 6, 8, 5, 7$, which is different from the sequence $1, 2, 3, 4, \square, 6, 8, 5, 7$ for the arrangement in Figure 8.3a. In terms of Figure 8.4, we have $x = 2$, $a = 3$, $b = 4$. We have changed the relative order of 2 with 3, and the relative order of 2 with 4. The move increased the number of pairs that were in error by 2 because pairs $(3, 5)$ and $(4, 5)$ are now in error.

We've seen that no matter what move we make, the change in the number of pairs in error is either -2, 0, or 2. All of these numbers are even. Hence, the invariant we deduce from the discussion above is that the *parity* of the number of pairs in error does not change. (Parity is either odd or even.) And now we can prove the following theorem about our invariant.

Theorem 8.4. *After n moves, the parity of the number of pairs in error is the same as initially.*

Proof Sketch. We use induction to prove the statement $(\forall n)P(n)$ where $P(n)$ is "after n moves, the parity of the number of pairs in error is the same as initially."

For the base case, notice that after zero moves we are still at the initial configuration, so the parity hasn't had a chance to change. Thus, the base case holds.

We would prove the inductive step by cases, copying the argument from our earlier discussion almost verbatim into this proof (which we don't do here to save some trees). Thus, the inductive step is proved, and so is the theorem. □

Since the parity of the number of pairs in error is odd in the arrangement in Figure 8.2a, and even in the arrangement in Figure 8.2b, Theorem 8.4 implies that no sequence of moves can transform the configuration from Figure 8.2a to the configuration in figure 8.2b.

It took us some time before we formulated our invariant. Indeed, it takes some ingenuity to recognize a common pattern such as the one described in Theorem 8.4. The kind of creative thinking that led us to the formulation of Theorem 8.4 is very common. In any area, and for any problem we face, we may not see a solution to the problem right away, so we play with the problem, fail a few times, try again using a modification of an earlier strategy or using an entirely new strategy, and then finally recognize a pattern that helps us solve the problem.