

CS 273: Intro to Theory of Computation, Fall 2007

Head-banging 3 (25-27 Sept)

More Structural Induction: Let $\Sigma = \{a, b, c\}$, and recursively define $L \subseteq \Sigma^*$ as the smallest set satisfying the following rules:

Rule 1: $ba \in L$

Rule 2: If $w \in L$, then $baw \in L$.

Rule 3: If $w \in L$, and $w = xaby$ for strings $x, y \in \Sigma^*$, then $xcby \in L$.

Rule 4: If $w \in L$, and $w = xcy$ for strings $x, y \in \Sigma^*$, then $xcby \in L$.

Note that x and y do not have to be in L in rules 3 and 4.

Prove that for every $w \in L$, the last character of w is an a .

Solution:

Base case: $ba \in L$, ends in an a .

Induction Hypothesis: Let $w \in L$ end in an a .

Then baw ends in an a . If $w = xaby$, then $y \neq \epsilon$, and so y ends in an a . Thus $xcby$ ends in an a as well. Similarly, if $w = xcy$, then $y \neq \epsilon$, and so y ends in an a . Thus $xcby$ ends in an a as well.

By induction, the result is proved. \square

All this structural induction business seems like subterfuge or trickery, doesn't it? Consider the following totally equivalent proof by induction on the number of applications of the rules:

Solution 2: Let $L_0 = \emptyset$, and let L_{n+1} be obtained from L_n by applying rules 1-4 to all elements of L_n , and taking the union with L_n . $L_1 = \{ba\}$, $L_2 = \{ba, baba\}$, $L_3 = \{ba, baba, bababa, bcba\}$, $L_4 = \{ba, baba, bababa, babababa, bcba, bcbaba, bacbca, bcbeba\}$, etc.

Base case: ($n = 0$) Since $L_0 = \emptyset$, vacuously, all strings in it end in an a .

Induction Hypothesis: Assume that all strings in L_n end in an a .

Let $w \in L_{n+1}$, then either $w \in L_n$, or w is obtained either from rule 1, or by applying one of rules 2, 3 or 4 to a string $w' \in L_n$. If $w \in L_n$, then w ends in an a by the induction hypothesis. If w was obtained from rule 1, then $w = ba$, which ends in an a . Now assume that w was obtained by applying one of rules 2-4 to $w' \in L_n$. Note that w' ends in an a . If w was obtained by applying rule 2, then $w = baw'$, and hence ends in an a , since w' does. If w was obtained by applying rule 3, then $w' = xaby$ for some strings x and y , and $w = xcby$. Since w' ends in an a , $y \neq \epsilon$, and y ends in an a . Thus w ends in an a because y does. If w was obtained by applying rule 4, then $w' = xcy$ for some strings x and y , and $w = xcby$. Since w' ends in an a , $y \neq \epsilon$, and y ends in an a . Thus w ends in an a because y does.

Thus each element of L_{n+1} ends in an a .

Since each element of L is in L_n for some n (this is essentially what "the smallest set satisfying the following rules" says), it ends in an a as well. \square

Hopefully, you can see how the two proofs are related. All we *actually* need to make the induction step go through is that all the inputs to the rules end in an a . The general form of recursion-structural induction problems that we've been doing goes something like this:

Theorem. Let S be the smallest set satisfying the following rules:

Rule 0: $a_0, a_1, a_2, \dots, a_n \in S$.

...

Rule n : If $x_0, \dots, x_k \in S$, and have property R_n , then $f_n(x_0, \dots, x_k) \in S$.

...

For all $x \in S$, x satisfies property P .

Proof: Base case: a_0, a_1, \dots, a_n satisfy property P because...

Induction step: (Repeat for each rule, though they can be combined if possible)

Induction Hypothesis: Let $x_0, \dots, x_k \in S$ have property R_n and property P .

Then $f_n(x_0, \dots, x_k) \in S$ by rule n , and $f_n(x_0, \dots, x_k)$ has property P because... \square

There is a theorem which states that you can translate any proof of this form into a proof using induction on the number of applications of the rules. Since this method of proof is so much simpler, it's useful to have around. Some important things to remember:

1. Always have a base case. At this point, it's a good idea to write "Base Case: ..."
2. Always have an induction hypothesis. Again, it's a good idea to write out explicitly what your induction hypothesis is. Writing the steps out makes it easier to assign partial credit and to understand what you're doing. Make absolutely sure that your induction hypothesis is not identical with the thing you're trying to prove. They usually look quite similar, but are logically quite different.
3. If you're trying to prove two things are identical, you need separate variable names for them. Also, there are usually two directions to identity proofs—e.g. you must show that the first set is contained in the second, then that the second is contained in the first.