CS 2742 (Logic in Computer Science) – Fall 2008
Lecture 27

Antonina Kolokolova

November 28, 2008

7.1 Equivalences of well-ordering, induction and complete induction.

Theorem 1. *Well-ordering principle, (weak) induction and strong induction are all equivalent to each other.*

Here is a very brief (and technical) outline of the main structure of the proof of the equivalences. The structure of the proof is circular: first we show that well-ordering implies induction, then that induction implies strong induction, and finally strong induction implies well-ordering, completing the cycle of implications.

Proof. 1) Well-ordering implies induction.
Assume well-ordering holds. Let $0 \in A$ and let $\forall i \in \mathbb{N}$, if $i \in A$ then $i+1 \in A$. Need to show $\mathbb{N} \subset A$. The rest is by contradiction. Look at $\bar{A} = \mathbb{N} - A$. If $\mathbb{N} \not\subseteq A$, then $\bar{A}$ is nonempty. By well-ordering, $\bar{A}$ has a minimal element j. That element is > 0, since $0 \in A$. Then $j-1$ is a natural number. But then $(j-1)+1$ must be in A. Contradiction.

2) Induction implies complete induction.
Prove by induction the following property: $P'(n) = \forall i < n P(i)$.

3) Complete induction implies well-ordering.
Let A be a subset of $\mathbb{N}$ with no minimal element. Show that A is empty. For any $i \in \mathbb{N}$, any number less than i is not in A. Then $i \notin A$ either (it would be the minimal element of A then). Look at the complement of A, $\bar{A}$. By complete induction, if any natural number less than i is in $\bar{A}$, then i is also in $\bar{A}$. But then every natural number is in $\bar{A}$. So A is empty.

□

7.2 Recursive definitions

Definition 1. *A recursive definition consists of:*

1) ***Base of recursion**: a statement that certain objects belong to a set.*
2) ***Recursion:** a collections of rules indicating how to form new set objects from those already known to be in the set.*
3) ***Restriction**: A statement that no objects belong to the set other than those coming from the base and the recursion rules.*

Example 1. Fibonacci: $F_0 = F_1 = 1$, and $F_k = F_{k-1} + F_{k-2}$.

Example 2. (Propositional formulas.)
Here we will give a formal definition of formulas of propositional logic.
Base of the recursion: propositional variables $p, q, r, \ldots$ and constants F, T are propositional formulas.
Recursion: If ϕ and ψ are propositional formulas, so are $\neg\phi$, $\phi \vee \psi$, $\phi \wedge \psi$, $\phi \to \psi$ and $\phi \leftrightarrow \psi$, as $(\neg\phi)$, $(\phi \vee \psi)$ and so on.
Restriction: ... and nothing else is a propositional formula.
For example, $(p \vee \neg q) \wedge T$ is a propositional formula, because it is made out of a $\wedge$ of $(p \vee \neg q)$ and T, and both of them are propositional formulas: T because it satisfies the base of induction, and $(p \vee \neg q)$ because it is a $\vee$ of two formulas p and $\neg q$, the first of which again satisfies the base case, and the second is a $\neg$ of a formula which is a base case.

Example 3. (Arithmetic expressions)
Base of the recursion: rational numbers and variables $x, y, z, \ldots$ are arithmetic expressions.
Recursion: For any two arithmetic expressions A and B, $A+B$, $A-B$, $A*B$, A/B $(A+B)$, $(A-B)$, $(A*B)$, (A/B) are arithmetic expressions.
Restriction:... and nothing else.
For example, $3 + 5 * x$ is an arithmetic expression.

The version of induction usually used to prove properties of objects in a recursively-defined set.

Definition 2. *(Structural induction)*

1) ***Base case:** prove that "simplest" elements (basis of recursion) satisfy the property.*
2) ***Induction step:** prove that each of the rules used to construct a more complex element preserves the property.*

Example 4. Here we will show that in any arithmetic expression (defined as above) the number of elements (such as variables or numbers) differs from the number of connectives (such as $+, -, *$ and $/$) by 1. Here, if a number or a variable occurs twice (as in $3 - x/3$) we count it twice. More precisely, given an expression A, if it has $el(A)$ elements and $con(A)$ connectives, then $el(A) = con(A) + 1$. For example, in the expression $3 + 5 * x$ there are two connectives $+$ and $*$, and three elements $3, 5$ and x.

The proof proceeds by structural induction. For the base case, in expressions consisting of just a number or just a variable there is one element (that number or a variable) and no connectives. So in the base case $el(A) = 1$, $con(A) = 0$ and thus $el(A) = con(A) + 1$ holds.

Now we will show that as long as an expression is constructed according to the rules, and out of valid expressions which satisfy (here is our induction hypothesis) the property we are trying to prove, the new constructed expression will also satisfy this property. That is, if C is constructed out of A and B according to the rules, and $el(A) = con(A) + 1$ and $el(B) = con(B)+$, then $el(C) = con(C) + 1$. But notice that all the rules here look like "expression followed by a connective followed by another expression". So $el(C) = el(A)+el(B)$ (because the new expression will have all the numbers and variables of the old ones, and nothing else – counting each occurrence of a number or variable as a separate element here). Similarly, $con(C) = con(A)+1+con(B)$, where that 1 comes from the connective that connects A and B. Thus, $con(C) = (el(A) - 1) + 1 + (el(B) - 1) = (el(A) + el(B) - 1 - 1 + 1 = el(C) - 1$, which is exactly what we were trying to show. We could also go through a separate proof for each of $+, -.*, /$ and the cases with parentheses (which we do not count here at all), but all these cases would be the same. So, for example, combining the expression 3 with the expression $5 * x$ using the connective $+$ gives us the expression $3+5*x$ with 1+2=3 elements and 0+1+1=2 connectives.