For Discrete Mathematics 2018-19: Well-founded Induction and Recursion

An addition to Part IA Comp. Sci. Lecture Notes

November 25, 2018

Chapter 1

Well-founded induction

This additional note introduces the powerful general proof principle of well-founded induction and its associated method of definition called well-founded recursion. They are based on the concept of a well-founded relation. Well-founded induction has many applications but is especially important for defining, and proving properties of, terminating programs.

1.1 Structural induction

We have seen mathematical induction and strong induction on the natural numbers $\mathbb{N}$. The principles are used to prove properties of natural numbers but they can also be turned to a means of defining functions on natural numbers, definition by mathematical induction, giving a form of recursion.

Before treating well-founded induction, let's examine another well-used proof principle, that of *structural induction*. We proceed from an example. The syntax of Boolean propositions is sometimes given by the following grammar:

$$\mathtt{A}, \mathtt{B}, \ldots ::= \mathtt{a}, \mathtt{b}, \mathtt{c}, \cdots \mid \mathtt{T} \mid \mathtt{F} \mid \mathtt{A} \wedge \mathtt{B} \mid \mathtt{A} \vee \mathtt{B} \mid \neg \mathtt{A}$$

By which we mean a proposition, which we will typically call $\mathtt{A}, \mathtt{B}, \cdots$, is either a propositional variable from among $\mathtt{a}, \mathtt{b}, \mathtt{c} \cdots \in \mathtt{Var}$, a set of propositional variables, the proposition true $\mathtt{T}$ or the proposition false $\mathtt{F}$, or built up using the logical operations of conjunction $\wedge$, disjunction $\vee$ or negation $\neg$. To avoid excessive brackets in writing Boolean propositions we adopt the usual convention that the operation $\neg$ binds more tightly than the two other operations $\wedge$ and $\vee$, so that $\neg\mathtt{A} \vee \mathtt{B}$ means $(\neg\mathtt{A}) \vee \mathtt{B}$. Boolean propositions are ubiquitous in science and everyday life. They are an unavoidable ingredient of almost all precise discourse, and of course of mathematics and computer science.

If we wish to establish that a property holds of all Boolean propositions we often carry out a *proof by structural induction*. We prove a property (the induction hypothesis, IH) holds of all propositions by showing

- IH holds of all atomic propositions (propositional variables, $\mathtt{T}$, $\mathtt{F}$), and

- that IH holds of compound propositions (for instance $\mathtt{A} \wedge \mathtt{B}$) follows from IH holding of immediate subexpressions (in this instance $\mathtt{A}$ and $\mathtt{B}$).

We can also give *definitions by structural induction*. For example, we can define the length of a Boolean proposition by structural induction as follows:

$$\begin{aligned}
&|\mathtt{a}| = 1, \quad |\mathtt{T}| = 1, \quad |\mathtt{F}| = 1, \\
&|\mathtt{A} \wedge \mathtt{B}| = |\mathtt{A}| + |\mathtt{B}| + 1, \\
&|\mathtt{A} \vee \mathtt{B}| = |\mathtt{A}| + |\mathtt{B}| + 1, \quad |\neg \mathtt{A}| = |\mathtt{A}| + 1 \ .
\end{aligned}$$

We define the length of atomic propositions and then the length of of compound propositions (for instance $\mathtt{A} \wedge \mathtt{B}$) follows from the length of immediate subexpressions (in this instance $\mathtt{A}$ and $\mathtt{B}$).

More interestingly perhaps, we can define a translation which eliminates disjunction from Boolean propositions by the following structural induction:

$$\begin{aligned}
&tr(\mathtt{a}) = \mathtt{a}, \quad tr(\mathtt{T}) = \mathtt{T}, \quad tr(\mathtt{F}) = \mathtt{F}, \\
&tr(\mathtt{A} \wedge \mathtt{B}) = tr(\mathtt{A}) \wedge tr(\mathtt{B}), \\
&tr(\mathtt{A} \vee \mathtt{B}) = \neg(\neg tr(\mathtt{A}) \wedge \neg tr(\mathtt{B})), \quad tr(\neg \mathtt{A}) = \neg tr(\mathtt{A}) \ .
\end{aligned}$$

Exercise 1.1 Prove by structural induction on Boolean propositions that

$$|tr(\mathtt{A})| \leq 3|\mathtt{A}| - 1 \ ,$$

for all Boolean propositions $\mathtt{A}$. □

Later in the course, you'll meet the principle of *rule induction*, a name and a way of presenting inductive definitions to computer scientists I introduced in my first stint here back in the 1980's. It concerns sets which are built up by repeatedly applying rules and their associated induction principles. For the natural numbers $\mathbb{N}$ the rules would say: adjoin 0; then adjoin successors. For Boolean expressions the rules would be those expressed by their grammar: add atomic propositions; add the compound propositions you can build from them.

1.2 Well-founded relations

Mathematical and structural induction are special cases of a general and powerful proof principle called well-founded induction. In essence structural induction works because breaking down an expression into subexpressions cannot go on forever, eventually it must lead to atomic expressions which cannot be broken down any further. If a property fails to hold of any expression then it must fail on some minimal expression which when it is broken down yields subexpressions, all of which satisfy the property. This observation justifies the principle of structural induction: to show a property holds of all expressions it is sufficient to show that property holds of an arbitrary expression if it holds of all its subexpressions. Similarly with the natural numbers, if a property fails to hold

of all natural numbers then there has to be a smallest natural number at which it fails. The essential feature shared by both the subexpression relation and the predecessor relation on natural numbers is that they do not give rise to infinite descending chains. This is the feature required of a relation if it is to support well-founded induction.

Definition: A *well-founded relation* is a binary relation $\prec$ on a set A such that there are no infinite descending chains $\cdots \prec a_i \prec \cdots \prec a_1 \prec a_0$. When $a \prec b$ we say a is a *predecessor* of b.

Note a well-founded relation is necessarily *irreflexive i.e.*, for no a do we have $a \prec a$, as otherwise there would be the infinite descending chain $\cdots \prec a \prec \cdots \prec a \prec a$. We shall generally write $\preceq$ for the reflexive closure of the relation $\prec$, *i.e.*

$$a \preceq b \iff a = b \text{ or } a \prec b.$$

(A relation $\prec$ for which $\preceq$ is a total order is traditionally called a *well-order*.)

Sometimes one sees an alternative definition of well-founded relation, in terms of minimal elements:

Proposition 1.2 *Let $\prec$ be a binary relation on a set A. The relation $\prec$ is well-founded iff any nonempty subset Q of A has a minimal element, i.e. an element m such that*

$$m \in Q \ \& \ \forall b \prec m.\ b \notin Q.$$

Proof.
"if": Suppose every nonempty subset of A has a minimal element. If $\cdots \prec a_i \prec \cdots \prec a_1 \prec a_0$ were an infinite descending chain then the set $Q = \{a_i \mid i \in \mathbb{N}\}$ would be nonempty without a minimal element, a contradiction. Hence $\prec$ is well-founded.
"only if": To see this, suppose Q is a nonempty subset of A. Construct a chain of elements as follows. Take a_0 to be any element of Q. Inductively, assume a chain of elements $a_n \prec \cdots \prec a_0$ has been constructed inside Q. Either there is some $b \prec a_n$ such that $b \in Q$ or there is not. If not , then stop the construction. Otherwise take $a_{n+1} = b$. As $\prec$ is well-founded the chain $\cdots \prec a_i \prec \cdots \prec a_1 \prec a_0$ cannot be infinite. Hence it is finite, of the form $a_n \prec \cdots \prec a_0$ with $\forall b \prec a_n.\ b \notin Q$. Take the required minimal element m to be a_n. □

Exercise 1.3 Let $\prec$ be a well-founded relation on a set B. Prove

(i) its transitive closure $\prec^+$ is also well-founded,

(ii) its reflexive, transitive closure $\prec^*$ is a partial order.

Elements $a, c \in B$ are in the transitive closure, *i.e.* $a \prec^+ c$, iff

$$a = b_1 \prec \cdots \prec b_n = c\ ,$$

for some $b_1, \cdots, b_n \in B$. □

1.3 Well-founded induction

Well-founded relations support an important proof principle.

The principle of well-founded induction
Let $\prec$ be a well founded relation on a set A. To show $\forall a \in A.\ P(a)$ it suffices to prove that for all $a \in A$

$$[\forall b \prec a.\ P(b)] \Rightarrow P(a)\ .$$

The principle reduces showing that a property (the induction hypothesis) holds globally to showing that the property is preserved locally by the well founded relation.

We now prove the principle. The proof rests on the observation, Proposition 1.2, that any nonempty subset Q of a set A with a well-founded relation $\prec$ has a minimal element. To justify the principle, we assume $\forall a \in A.\ ([\forall b \prec a.\ P(b)] \Rightarrow P(a))$ and produce a contradiction by supposing $\neg P(a)$ for some $a \in A$. Then, as we have observed, there must be a minimal element m of the set $\{a \in A \mid \neg P(a)\}$. But then $\neg P(m)$ and yet $\forall b \prec m.\ P(b)$, which contradicts the assumption.

Example: If we take the relation $\prec$ to be the predecessor relation

$$n \prec m \text{ iff } m = n + 1$$

on the non-negative integers the principle of well-founded induction specialises to mathematical induction. □

Example: If we take $\prec$ to be the "strictly less than" relation $<$ on the non-negative integers, the principle specialises to strong induction: To show $P(n)$ for all nonnegative integers n, it suffices to show

$$(\forall m < n.\ P(m)) \Rightarrow P(n)$$

for all nonnegative integers n. □

Example: If we take $\prec$ to be the relation between syntactic expressions such that $a \prec b$ holds iff a is an immediate subexpression of b we obtain the principle of *structural induction* as a special case of well-founded induction. □

Proposition 1.2 provides an alternative to proofs by the principle of well-founded induction. Suppose A is a well-founded set. Instead of using well-founded induction to show every element of A satisfies a property, we can consider the subset of A for which the property fails, *i.e.* the subset Q of counterexamples. By Proposition 1.2, to show Q is $\emptyset$ it is sufficient to show that Q cannot have a minimal element. This is done by obtaining a contradiction from the assumption that there is a minimal element in Q. Whether to use this

approach or the principle of well-founded induction is largely a matter of taste, though sometimes, depending on the problem, one approach can be more direct than the other.

A special instance of Proposition 1.2 is well-known to be equivalent to mathematical induction. It is the principle that every nonempty subset of natural numbers has a least element.

Exercise 1.4 For a suitable well-founded relation on strings, use the "no counterexample" approach described above to show there is no string u which satisfies $au = ub$ for two distinct symbols a and b. □

Well-founded induction is the most important principle in proving the termination of programs. Uncertainties about termination arise because of loops or recursions in a program. If it can be shown that execution of a loop or recursion in a program decreases the value in a well-founded set then execution must eventually terminate.

1.4 Building well-founded relations

Applying the principle of well-founded induction often depends on a judicious choice of well-founded relation.

1.4.1 Fundamental well-founded relations

We have already made use of well-founded relations like that of proper subexpression on syntactic sets, or $<$ on natural numbers.

Here are some ways to construct further well-founded relations. Recall that we use $x \preceq y$ to mean ($x \prec y$ or $x = y$).

We use some basic set theory, in particular products of sets, covered later in the course.

1.4.2 Transitive closure

If $\prec$ is well-founded relation on A, then so is its transitive closure $\prec^+$. Clearly any infinite descending chain

$$\cdots \prec^+ a_n \prec^+ \cdots \prec^+ a_1 \prec^+ a_0$$

with respect to $\prec^+$ would induce an infinite descending chain with respect to $\prec$. (This was part of an earlier exercise!)

1.4.3 Product

If $\prec_1$ is well-founded on A_1 and $\prec_2$ is well-founded on A_2 then taking

$$(a_1, a_2) \preceq (a_1', a_2') \Leftrightarrow_{def} a_1 \preceq_1 a_1' \text{ and } a_2 \preceq_2 a_2'$$

determines a relation $\prec = (\preceq \setminus \mathsf{id}_{A_1 \times A_2})$ in $A_1 \times A_2$ called the product relation:

Proposition 1.5 *The product relation of well-founded relations is well-founded.*

Proof. Suppose $\prec_1$ is well-founded on A_1 and $\prec_2$ is well-founded on A_2. Assume their product relation $\prec$ is not well-founded, *i.e.* that there is an infinite descending chain

$$\cdots \prec (x_n, y_n) \prec \cdots \prec (x_1, y_1) \prec (x_0, y_0) \ .$$

But then, from the definition of the product relation $\prec$, either

$$\cdots \prec_1 x_{n_k} \prec_1 \cdots \prec_1 x_{n_1} \prec_1 x_{n_0}$$

or

$$\cdots \prec_2 y_{n_k} \prec_2 \cdots \prec_2 y_{n_1} \prec_2 y_{n_0} \ ,$$

which contradicts the well-foundedness of $\prec_1$ and $\prec_2$. □

We'll see applications of the product of well-founded relations in the next section. However product relations are not as generally applicable as those produced by lexicographic products.

1.4.4 Lexicographic products

Let $\prec_1$ be well-founded on A_1 and $\prec_2$ be well-founded on A_2. Define their lexicographic product by

$$(a_1, a_2) \prec_{lex} (a_1', a_2') \text{ iff } a_1 \prec_1 a_1' \text{ or } (a_1 = a_1' \ \& \ a_2 \prec_2 a_2') \ .$$

Proposition 1.6 *The lexicographic product of well-founded relations is well-founded.*

Proof. Suppose $\prec_1$ is well-founded on A_1 and $\prec_2$ is well-founded on A_2. Assume their lexicographic product $\prec_{lex}$ is not well-founded, *i.e.* that there is an infinite descending chain

$$\cdots \prec (x_n, y_n) \prec \cdots \prec (x_1, y_1) \prec (x_0, y_0) \ .$$

From the definition of the lexicographic relation $\prec_{lex}$

$$\cdots \preceq_1 x_n \preceq_1 \cdots \preceq_1 x_1 \preceq_1 x_0 \ .$$

But $\prec_1$ is well-founded so from some stage on, say $m \geq n$, this chain is constant. But then from the definition of the lexicographic relation $\prec$,

$$\cdots \prec_2 y_{n+i} \prec_2 \cdots \prec_2 y_{n+1} \prec_2 y_n \ ,$$

which contradicts the well-foundedness of $\prec_2$. □

Exercise 1.7 Let $\prec$ be a well-founded relation on a set X such that $\preceq$ is a total order. Show it need not necessarily make the set

$$\{x \in X \mid x \prec y\}$$

finite for all $y \in X$.
[Recall a total order is a partial order $\leq$ such that $x \leq y$ or $y \leq x$ for all its elements x, y. Hint: Consider the lexicographic product of $<$ and $<$ on $\mathbb{N} \times \mathbb{N}$.]
□

1.4.5 Inverse image

Let f be a function from A to B and $\prec_B$ a well-founded relation on B. Then $\prec_A$ is well-founded on A where

$$a \prec_A a' \Leftrightarrow_{def} f(a) \prec_B f(a')$$

for $a, a' \in A$.

Exercise 1.8 Show the inverse image of a well-founded relation is a well-founded relation. □

1.5 Applications

1.5.1 Euclid's algorithm for gcd

We can use well-founded induction to show the correctness of Euclid's algorithm for calculating the greatest common divisor of a pair of positive natural numbers.[1] One way to formulate Euclid's algorithm is through a reduction relation $\longrightarrow_E$ between pairs of positive natural numbers defined as follows:

$$(m, n) \longrightarrow_E (m, n - m) \quad \text{if } m < n \ ,$$

$$(m, n) \longrightarrow_E (m - n, n) \quad \text{if } n < m \ .$$

So (m, n) reduces to $(m, n-m)$ if $m < n$ and to $(m-n, n)$ if $n < m$. Notice there is no reduction when $m = n$; in this case the reduction terminates. (To illustrate well-founded induction, I have chosen a simplified version of Euclid's algorithm in which the division algorithm of the notes, based on successive subtractions, is built into the algorithm for finding the gcd.)

It is easy to check that the following properties hold for the hcf of natural numbers:

Proposition 1.9

(a) $gcd(m, n) = gcd(m, n - m)$ *if* $m < n$,

(b) $gcd(m, n) = gcd(m - n, n)$ *if* $n < m$,

(c) $gcd(m, m) = m$.

Proof. The highest common factor of natural numbers m and n, $gcd(m, n)$, is characterised by:

(i) $gcd(m, n)$ divides m and n;

(ii) if k divides m and n, then k divides $gcd(m, n)$.

[1] Another name for greatest common divisor is *highest common factor* (hcf).

In all cases the proof proceeds by showing any divisor of the left is also a divisor of the right and *vice versa*; two natural numbers with the same divisors must be equal. As an example we show (a) $gcd(m,n) = gcd(m,n-m)$ assuming $m < n$. Suppose k divides the lhs $gcd(m,n)$. Then k certainly divides m and n by (i), and so divides m and $n-m$. Thus k divides the rhs $gcd(m,n-m)$ by (ii). Suppose now that k divides the rhs $gcd(m,n-m)$. Then k divides m and $n-m$ by (i). It follows that k divides m and n, and so the lhs $gcd(m,n)$ by (ii). □

Euclid's reduction terminates with the gcd of the positive natural numbers it starts with:

Theorem 1.10 *For all $m,n \in \mathbb{N}$ with $m,n > 0$,*

$$(m,n) \longrightarrow_E^* (gcd(m,n), gcd(m,n)) \ .$$

(The relation $\longrightarrow_E^*$ relates pairs which are equal or related by $\longrightarrow_E^+$, i.e. connected by a $\longrightarrow_E$ chain.)

Proof. Let $\prec$ between positive natural numbers be the well-founded relation constructed as the product of $<$ and $<$ on positive natural numbers. Take

$$P(m,n) \Leftrightarrow_{def} \ (m,n) \longrightarrow_E^* (gcd(m,n), gcd(m,n))$$

as the induction hypothesis. We prove $P(m,n)$ for all positive natural numbers m,n by well-founded induction.

Let (m,n) be a pair of positive natural numbers. Assume $P(m',n')$ for all $(m',n') \prec (m,n)$. Consider the cases:

Case $m < n$. In this case $(m,n) \longrightarrow_E (m,n-m)$ and because $P(m,n-m)$ by the induction hypothesis,

$$(m,n-m) \longrightarrow_E^* (gcd(m,n-m), gcd(m,n-m)) \ .$$

Hence

$$(m,n) \longrightarrow_E^* (gcd(m,n-m), gcd(m,n-m)) \ ,$$

by the properties of the reflexive transitive closure $\longrightarrow_E^*$. Also $gcd(m,n) = gcd(m,n-m)$. Thus $P(m,n)$ in this case.

Case $n < m$. This case is very similar.

Case $m = n$. In this case

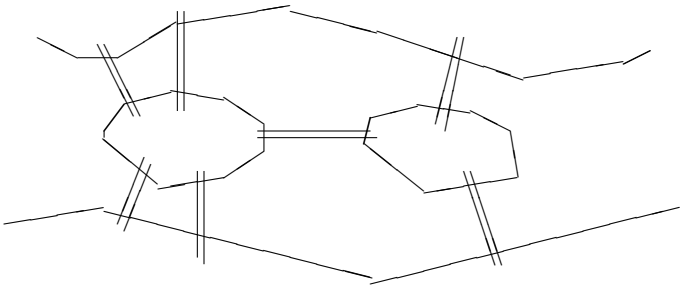
$$(m,n) \longrightarrow_E^* (m,n)$$

as $\longrightarrow_E^*$ is reflexive. Also $gcd(m,n) = m = n$. Thus $P(m,n)$ in this case.

In all possible cases for (m,n) we can derive $P(m,n)$ from the assumption that $P(m',n')$ holds for all $\prec$-predecessors (m',n'). Hence by well-founded induction we have established $P(m,n)$ for all positive natural numbers m,n. □

1.5.2 Eulerian graphs

Well-founded induction is a proof principle of widespread applicability. Here's an example of its use in graph theory. A graph is a pair (V, E) consisting of a set of *vertices* V and a set of *edges* E—an edge between vertices v and v' is represented as an unordered pair $\{v, v'\}$. A graph is *connected* iff any two vertices v, v' are connected by a path of edges $\{v_0, v_1\}, \{v_1, v_2\}, \cdots, \{v_{n-1}, v_n\}$ where $v = v_0$ and $v_n = v'$. A circuit of a graph consists of a path $\{v_0, v_1\}, \{v_1, v_2\}, \cdots, \{v_{n-1}, v_n\}$ for which $v_0 = v_n$. A circuit is *Eulerian* iff it visits each edge exactly once. When does a finite connected graph have a Eulerian circuit? The answer to this question, the theorem below, is due to the great mathematician Leonhard Euler (1707-1783). Reputedly he was asked by the townspeople of Königsberg whether it was possible to go for a walk in the town so as to cross each of its numerous bridges exactly once (Was it? See the figure and Theorem 1.11 below).



Theorem 1.11 *A finite connected graph has an Eulerian circuit iff every vertex has even degree, i.e. has an even number of edges connected to it.*

Proof.
"only if": Consider a finite connected graph. Assume it has an Eulerian circuit. Because the graph is connected, each vertex must appear at least once in the circuit (why?). Each occurrence of a vertex in the Eulerian circuit is accompanied by a pair of distinct edges—one going into the vertex and one going out. All edges appear precisely once in the Eulerian circuit, so each vertex has even degree.
"if": For finite connected graphs $G_1 = (V_1, E_1)$ and $G_2 = (V_2, E_2)$, define

$$G_1 \preceq G_2 \iff V_1 \subseteq V_2 \;\&\; E_1 \subseteq E_2 \ .$$

The relation $\prec$ between finite connected graphs is an example of the product of two well-founded relations, so is itself well-founded. We shall use well-founded induction to establish the following property of all finite connected graphs:

> if each vertex has even degree, then the graph has an Eulerian circuit.

We take the above as our induction hypothesis.

Let G be a finite connected graph in which each vertex has even degree. Assume that for all graphs G' with $G' \prec G$ if each vertex of G' has even degree,

then G' has an Eulerian circuit. That is, we assume the induction hypothesis for all $G' \prec G$.

We first find a circuit C in the graph. Starting at some vertex (it doesn't matter which) form a maximal path along edges in which no edge appears more than once. Because the graph is finite such a path must contain a loop, the circuit C. Any occurrence of a vertex in C is accompanied by a pair of distinct edges—one ingoing and one outgoing. Remove all the edges of C from the graph G. This will result in one or more connected components G', where all vertices of G' have even degree and $G' \prec G$. Hence, each such connected component has an Eulerian circuit. Linking these into C we obtain an Eulerian circuit for G. □

1.6 Well-founded recursion

One sees definition by structural induction and definition by induction. Such definitions are a form of recursive definition: the result of a function on an argument is defined in terms of the results of the same function on strictly smaller arguments. For example, earlier in Section **??** we saw the definition by structural induction of the length of Boolean propositions. A well-known example from mathematics is that of the Fibonacci numbers $0, 1, 1, 2, 3, 5, 8, 13, \cdots$. They are given by a recurrence relation

$$\mathit{fib}(0) = 0, \quad \mathit{fib}(1) = 1, \quad \mathit{fib}(n) = \mathit{fib}(n-1) + \mathit{fib}(n-2) \text{ for } n > 1 \ ,$$

in which the nth Fibonacci number is defined in terms of the two preceding numbers.

Exercise 1.12 There are five equally-spaced stepping stones in a straight line across a river. The distance d from the banks to the nearest stone is the same as that between the stones. You can hop distance d or jump $2d$. So for example you could go from one river bank to the other in 6 hops. Alternatively you might first jump, then hop, then jump, then hop. How many distinct ways could you cross the river (you always hop or jump forwards)?

Describe how many distinct ways you could cross a river with n similarly spaced stepping stones. □

In the same manner as in the definitions of length *length* and *fib* above, we are entitled to define functions on an arbitrary well-founded set. Suppose B is a set with a well-founded relation $\prec$. Definition by well-founded induction, traditionally called *well-founded recursion*, allows the definition of a function f from B by specifying its value $f(b)$ at an arbitrary b in B in terms of $f(b')$ for $b' \prec b$. In more detail:

Definition by well-founded recursion
Suppose B is a set with a well-founded relation $\prec$. Suppose C is a set and $F(b, c_1, \cdots, c_k, \cdots)$ is an expression such that

$$\forall b \in B,\ c_1, \cdots, c_k, \cdots \in C. \quad F(b, c_1, \cdots, c_k, \cdots) \in C \ .$$

Then, a recursive definition of the form, for all $b \in B$,

$$f(b) = F(b, f(b_1), \cdots, f(b_k), \cdots) \ ,$$

where $b_1 \prec b, \cdots, b_k \prec b, \cdots$, determines a unique function f from B to C (*i.e.*, there is a unique function f from B to C which satisfies the recursive definition).

You can check that definitions by mathematical induction, structural induction, and, in particular of the Fibonacci numbers fit the general scheme of definition by well-founded recursion.

Well-founded recursion and induction constitute a general method often appropriate when functions are intended to be total. For example, it immediately follows from well-founded recursion that that there is a unique total function on the nonnegative integers such that

$$ack(m,n) = \begin{cases} n+1 & \text{if } \ m = 0 \ , \\ ack(m-1,1) & \text{if } \ m \neq 0, n = 0 \ , \\ ack(m-1, ack(m, n-1)) & \text{otherwise} \ , \end{cases}$$

for all $m, n \in \mathbb{N}$; observe that the value of ack at the pair (m, n) is defined in terms of its values at the lexicographically smaller pairs $(m-1, 1)$ and $(m, n-1)$.

This is Ackermann's function. Ackermann's function provided a counterexample to the conjecture that all computable functions are primitive recursive—it grows way too fast.[2] As a recursive program Ackermann's function looks like:

$$\begin{aligned} A(x,y) = \ & \textbf{if } x = 0 \textbf{ then } y+1 \textbf{ else} \\ & \textbf{if } y = 0 \textbf{ then } A(x-1,1) \textbf{ else} \\ & \qquad A(x-1, A(x, y-1)) \end{aligned}$$

In practice a program to calculate Ackermann's function won't terminate in a reasonable time on any machine for all but the smallest values.

A great many recursive programs are written so that some measure within a well-founded set decreases as they are evaluated. Not all recursive definitions are well-founded; it's a fact of life that programs may fail to terminate, and so in general determine *partial* functions from input to output. The techniques of semantics, domain theory (where least fixed points play a central role) or operational semantics (based on inductive definitions) apply in this broader situation.[3]

Exercise 1.13 (McCarthy's 91 function) Show the relation $\prec$, where

$$n \prec m \Leftrightarrow m < n \leq 101,$$

for $n, m \in \mathbb{N}$, is well-founded.

[2] Computable and primitive recursive functions are central topics in the second-year CS course on "Computability."

[3] *Cf.* the Part IB course 'Semantics' and the Part II course 'Denotational Semantics.'

Deduce by well-founded recursion that there is a function f from $\mathbb{N}$ to $\mathbb{N}$ satisfying

$$f(x) = \begin{cases} x - 10 & \text{if } x > 100 \text{ ,} \\ f(f(x+11)) & \text{otherwise ,} \end{cases}$$

for all $x \in \mathbb{N}$.

Show by well-founded induction with respect to $\prec$ that

$$f(x) = \begin{cases} x - 10 & \text{if } x > 100 \text{ ,} \\ 91 & \text{otherwise ,} \end{cases}$$

for all $x \in \mathbb{N}$. □

For more on well-founded induction and recursion see the notes "Set Theory for Computer Science" on my homepage. They contain a proof to justify definition by well-founded recursion and explain how inductive definitions by rules lead to a technique of well-founded induction via the notion of derivations.