Roberto Bruni, Ugo Montanari

Models of Computation

– Monograph –

March 6, 2016

Springer

Mathematical reasoning may be regarded rather schematically as the exercise of a combination of two facilities, which we may call intuition and ingenuity.

Alan Turing[1]

[1] The purpose of ordinal logics (from Systems of Logic Based on Ordinals), Proceedings of the London Mathematical Society, series 2, vol. 45, 1939.

Contents

Acronyms

$\sim$	operational equivalence in IMP (see Definition 3.3)
$\equiv_{den}$	denotational equivalence in HOFL (see Definition 10.4)
$\equiv_{op}$	operational equivalence in HOFL (see Definition 10.3)
$\simeq$	CCS strong bisimilarity (see Definition 11.5)
$\approx$	CCS weak bisimilarity (see Definition 11.16)
$\cong$	CCS weak observational congruence (see Section 11.7.2)
$\approx_d$	CCS dynamic bisimilarity (see Definition 11.17)
$\overset{\circ}{\sim}_E$	π-calculus early bisimilarity (see Definition 13.3)
$\overset{\circ}{\sim}_L$	π-calculus late bisimilarity (see Definition 13.4)
$\sim_E$	π-calculus strong early full bisimilarity (see Section 13.5.3)
$\sim_L$	π-calculus strong late full bisimilarity (see Section 13.5.3)
$\overset{\bullet}{\approx}_E$	π-calculus weak early bisimilarity (see Section 13.5.4)
$\overset{\bullet}{\approx}_L$	π-calculus weak late bisimilarity (see Section 13.5.4)
$\mathscr{A}$	interpretation function for the denotational semantics of IMP arithmetic expressions (see Section 6.2.1)
ack	Ackermann function (see Example 4.18)
Aexp	set of IMP arithmetic expressions (see Chapter 3)
$\mathscr{B}$	interpretation function for the denotational semantics of IMP boolean expressions (see Section 6.2.2)
Bexp	set of IMP boolean expressions (see Chapter 3)
$\mathbb{B}$	set of booleans
$\mathscr{C}$	interpretation function for the denotational semantics of IMP commands (see Section 6.2.3)
CCS	Calculus of Communicating Systems (see Chapter 11)
Com	set of IMP commands (see Chapter 3)
CPO	Complete Partial Order (see Definition 5.11)
$\text{CPO}_\perp$	Complete Partial Order with bottom (see Definition 5.12)
CSP	Communicating Sequential Processes (see Section 16.2)
CTL	Computation Tree Logic (see Section 12.1.2)
CTMC	Continuous Time Markov Chain (see Definition 14.15)

DTMC	Discrete Time Markov Chain (see Definition 14.14)
Env	set of HOFL environments (see Chapter 9)
fix	(least) fixpoint (see Definition 5.2.2)
FIX	(greatest) fixpoint
gcd	greatest common divisor
HML	Hennessy-Milner modal Logic (see Section 11.5)
HM-Logic	Hennessy-Milner modal Logic (see Section 11.5)
HOFL	A Higher-Order Functional Language (see Chapter 7)
IMP	A simple IMPerative language (see Chapter 3)
int	integer type in HOFL (see Definition 7.2)
Loc	set of locations (see Chapter 3)
LTL	Linear Temporal Logic (see Section 12.1.1)
LTS	Labelled Transition System (see Definition 11.2)
lub	least upper bound (see Definition 5.7)
$\mathbb{N}$	set of natural numbers
$\mathscr{P}$	set of closed CCS processes (see Definition 11.1)
PEPA	Performance Evaluation Process Algebra (see Chapter 16)
Pf	set of partial functions on natural numbers (see Example 5.10)
PI	set of partial injective functions on natural numbers (see Problem 5.11)
PO	Partial Order (see Definition 5.1)
PTS	Probabilistic Transition System (see Section 14.3.2)
$\mathbb{R}$	set of real numbers
$\mathscr{T}$	set of HOFL types (see Definition 7.2)
Tf	set of total functions from $\mathbb{N}$ to $\mathbb{N}_\perp$ (see Example 5.11)
Var	set of HOFL variables (see Chapter 7)
$\mathbb{Z}$	set of integers

Part II
IMP: a simple imperative language

This part focuses on models for sequential computations that are associated to IMP, a simple imperative language. The syntax and natural semantics of IMP are studied in Chapter 3, while its denotational semantics is presented in Chapter 6, where it is also reconciled with the operational semantics. Chapter 4 explains several induction principles exploited to prove properties of programs and semantics. Chapter 5 fixes the mathematical basis of denotational semantics. The concepts in Chapters 4 and 5 are extensively used in Chapter 6 and in the rest of the monograph.

Chapter 4
Induction and Recursion

To understand recursion, you must first understand recursion.
(traditional joke)

Abstract In this chapter we presents some induction techniques that will turn out useful for proving formal properties of the languages and models presented in the book. We start by introducing Noether principle of well-founded induction, from which we then derive induction principles over natural numbers, terms of a signature and derivations in a logical system. The chapter ends by presenting well-founded recursion.

4.1 Noether Principle of Well-founded Induction

In the literature several different kinds of induction are defined, but they all rely on the so-called *Noether principle of well-founded induction*. We start by defining this important principle and will then derive several induction methods.

4.1.1 Well-founded Relations

We recall some key mathematical notions and definitions.

Definition 4.1 (Binary relation). A *binary relation* (*relation* for short) $\prec$ over a set A is a subset of the cartesian product $A \times A$.

$$\prec \subseteq A \times A$$

For $(a,b) \in \prec$ we use the infix notation $a \prec b$ and also write equivalently $b \succ a$. Moreover, we write $a \not\prec b$ in place of $(a,b) \notin \prec$.

A relation $\prec \subseteq A \times A$ can be conveniently represented as an *oriented graph* whose nodes are the elements of A and whose arcs $n \to m$ represent the pairs $(n,m) \in \prec$ in the relation.

$$\begin{tikzcd}
 & & c \arrow[r, bend left] \arrow[dd, bend left] & d \\
a \arrow[r, bend left] & b \arrow[ur, bend left] & & \\
 & & e \arrow[ul, bend left] \arrow[r, bend right] & f
\end{tikzcd}$$

Fig. 4.1: Graph of a relation

Example 4.1. The graph in Figure 4.1 represents the relation $\prec$ over the set $\{a,b,c,d,e,f\}$ with $a \prec b$, $b \prec c$, $c \prec d$, $c \prec e$, $e \prec f$, $e \prec b$.

Definition 4.2 (Infinite descending chain). Given a relation $\prec$ over the set A, an *infinite descending chain* is an infinite sequence $\{a_i\}_{i \in \mathbb{N}}$ of elements in A such that

$$\forall i \in \mathbb{N}.\ a_{i+1} \prec a_i$$

An infinite descending chain can be represented as a function a from $\mathbb{N}$ to A such that $a(i)$ decreases (according to $\prec$) as i grows:

$$a(0) \succ a(1) \succ a(2) \succ \cdots$$

Definition 4.3 (Well-founded relation). A relation is *well-founded* if it has no infinite descending chains.

Definition 4.4 (Transitive closure). Let $\prec$ be a relation over A. The *transitive closure* of $\prec$, written $\prec^+$, is defined by the following inference rules

$$\frac{a \prec b}{a \prec^+ b} \qquad \frac{a \prec^+ b \qquad b \prec^+ c}{a \prec^+ c}$$

By the first rule, $\prec$ is always included in $\prec^+$. It can be proved that $(\prec^+)^+$ always coincides with $\prec^+$.

Definition 4.5 (Transitive and reflexive closure). Let $\prec$ be a relation over A. The *transitive and reflexive closure* of $\prec$, written $\prec^*$, is defined by the following inference rules

$$\frac{}{a \prec^* a} \qquad \frac{a \prec^* b \qquad b \prec c}{a \prec^* c}$$

It can be proved that both $\prec$ and $\prec^+$ are included in $\prec^*$ and that $(\prec^*)^*$ always coincides with $\prec^*$.

Example 4.2. Consider the usual "less than" $<$ relation over integers. Since we have, e.g., the infinite descending chain

$$4 > 2 > 0 > -2 > -4 > \ldots$$

it is not well-founded. Note that its transitive closure $<^+$ is the same as $<$.

Example 4.3. Consider the usual “less than” $<$ relation over natural numbers. We cannot have an infinite descending chain $\{a_i\}_{i\in\mathbb{N}}$ because there are only a finite number of elements less than a_0. Hence the relation $<$ over $\mathbb{N}$ is well-founded. Note that its transitive closure $<^+$ is the same as $<$.

Example 4.4. Consider the usual “less than or equal to” $\leq$ relation over natural numbers. Since we have, e.g., the infinite descending chain

$$4 \geq 2 \geq 0 \geq 0 \geq 0 \geq \ldots$$

it is not well-founded. Note also, than any infinite descending chain must include only a finite number of elements, because there are only a finite number of elements less than or equal to a_0, and therefore there exists some $k \in \mathbb{N}$ such that $\forall i \geq k.\ a_i = a_k$. Note that $\leq$ is the reflexive and transitive closure of $<$, i.e., $<^* = \leq$.

Theorem 4.1. *Let $\prec$ be a relation over A. For any $x, y \in A$, $x \prec^+ y$ if and only if there exist a finite number of elements $z_0, z_1, \ldots, z_k \in A$ such that*

$$x = z_0 \prec z_1 \prec \cdots \prec z_k = y.$$

The proof of the above theorem is left as an exercise (see Problem 4.4).

With respect to the oriented graph associated with the relation $\prec$, we note that $a \prec^+ b$ means that there is a non-empty finite path from a to b, while $a \prec^* b$ means that there is a (possibly empty) finite path from a to b.

Theorem 4.2 (Well-foundedness of $\prec^+$). *A relation $\prec$ is well-founded if and only if its transitive closure $\prec^+$ is well-founded.*

Proof. One implication is trivial: if $\prec^+$ is well-founded then $\prec$ is obviously well-founded, because any descending chain for $\prec$ is also a descending chain for $\prec^+$ (and all such chains are finite by hypothesis).

For the other direction, let us assume $\prec^+$ is non well-founded and take any infinite descending chain

$$a_0 \succ^+ a_1 \succ^+ a_2 \cdots$$

But whenever $a_i \succ^+ a_{i+1}$ there must be a finite descending $\prec$-chain of elements between a_i and a_{i+1} and therefore we can build an infinite descending chain

$$a_0 \succ \cdots \succ a_1 \succ \cdots \succ a_2 \succ \cdots$$

leading to a contradiction. $\square$

Example 4.5. Consider the “immediate precedence” relation $\prec$ over natural numbers, such that $n \prec n+1$ for all $n \in \mathbb{N}$. Note that the transitive closure of $\prec$ is the usual “less than” $<$ relation over natural numbers, i.e., $\prec^+ = <$. By Theorem 4.2 and Example 4.3 the relation $\prec$ over $\mathbb{N}$ is well-founded.

Definition 4.6 (Acyclic relation). We say that $\prec$ has a cycle if $\exists a \in A.\, a \prec^+ a$. We say that $\prec$ is *acyclic* if it has no cycle (i.e., $\forall a \in A.\, a \not\prec^+ a$).

Theorem 4.3 (Well-founded relations are acyclic). *If the relation $\prec$ is well-founded, then it is acyclic.*

Proof. We need to prove that:

$$\neg \exists a \in A.\ a \prec^+ a$$

By contradiction, we assume there is $a \in A$ such that $a \prec^+ a$. This means that $\prec^+$ is not well-founded, because we have an infinite descending chain

$$a \succ^+ a \succ^+ a \succ^+ a \ldots$$

By Theorem 4.2, $\prec$ is not well-founded, leading to a contradiction, because $\prec$ is well-founded by hypothesis. □

For example, the relation in Figure 4.1 is not acyclic and thus it is not well-founded.

Theorem 4.4 (Well-founded relations over finite sets). *Let A be a finite set and let $\prec$ be acyclic, then $\prec$ is well-founded.*

Proof. Since A is finite, any descending chain strictly longer than $|A|$ must contain (at least) two occurrences of a same element (by the so-called "pigeon hole principle") that form a cycle, but this is not possible because $\prec$ is acyclic by hypothesis. □

Definition 4.7 (Minimal element). Let $\prec$ be a relation over the set A. Given a set $Q \subseteq A$, we say that $m \in Q$ is *minimal* if there is no element $x \in Q$ such that $x \prec m$, i.e., $\forall x \in Q.\ x \not\prec m$.

It follows that Q has no minimal element if $\forall m \in Q.\ \exists x \in Q.\ x \prec m$.

Lemma 4.1 (Well-founded relation). *Let $\prec$ be a relation over the set A. The relation $\prec$ is well-founded if and only if every nonempty subset $Q \subseteq A$ contains a minimal element m.*

Proof. Since any double implication $P \Leftrightarrow Q$ is equivalent to $\neg P \Leftrightarrow \neg Q$, the statement of this lemma can be rephrased by saying that: *the relation $\prec$ has an infinite descending chain if and only if there exists a nonempty subset $Q \subseteq A$ with no minimal element*.

We prove each implication (of the transformed statement) separately.

$\Leftarrow$) We assume that $\prec$ has an infinite descending chain $a_1 \succ a_2 \succ a_3 \succ \cdots$ and we let $Q = \{a_1, a_2, a_3, \ldots\}$ be the set of all the elements in the infinite descending chain. The set Q has no minimal element, because for any candidate $a_i \in Q$ we know there is one element $a_{i+1} \in Q$ with $a_i \succ a_{i+1}$.

$\Leftarrow$) Let Q be a nonempty subset of A with no minimal element. Since Q is nonempty, it must contain at least an element. We randomly pick an element $a_0 \in Q$. Since a_0 is not minimal there must exists an element $a_1 \in Q$ such that $a_0 \succ a_1$, and we can iterate the reasoning (i.e. a_1 is not minimal and there is $a_2 \in Q$ with $a_0 \succ a_1 \succ a_2$, etc.). So we can build an infinite descending chain. □

Example 4.6 (Natural numbers). Both $n \prec n+1$ (the immediate precedence relation) and $n < n+1+k$ (the usual "less than" relation), with $n,k \in \mathbb{N}$, are simple examples of well-founded relations. In fact, from every element $n \in \mathbb{N}$ we can start a descending chain of length at most n.

Definition 4.8 (Terms over one-sorted signatures). Let $\Sigma = \{\Sigma_n\}_{n\in\mathbb{N}}$ a one-sorted signature, i.e., a set of ranked operators f such that $f \in \Sigma_n$ if f takes n arguments. We define the set of Σ-terms as the set T_Σ that is defined inductively by the following inference rule:

$$\frac{t_i \in T_\Sigma \quad i = 1,\dots,n \quad f \in \Sigma_n}{f(t_1,\dots,t_n) \in T_\Sigma} \tag{4.1}$$

Definition 4.9 (Terms over many sorted signatures). Let

- S be a set of *sorts* (i.e. the set of the different data types we want to consider);
- $\Sigma = \{\Sigma_{s_1\dots s_n,s}\}_{s_1,\dots,s_n,s\in S}$ be a *signature* over S, i.e. a set of typed operators ($f \in \Sigma_{s_1\dots s_n,s}$ is an operator that takes n arguments, the ith argument being of type s_i, and gives a result of type s).

We define the set of Σ-*terms* as the set

$$T_\Sigma = \{T_{\Sigma,s}\}_{s\in S}$$

where, for $s \in S$, the set $T_{\Sigma,s}$ is the set of terms of sort s over the signature Σ, defined inductively by the following inference rule:

$$\frac{t_i \in T_{\Sigma,s_i} \quad i = 1,\dots,n \quad f \in \Sigma_{s_1\dots s_n,s}}{f(t_1,\dots,t_n) \in T_{\Sigma,s}}$$

(When S is a singleton, we are in the same situation as in Definition 4.8 and write just Σ_n instead of $\Sigma_{w,s}$ with $w = \underbrace{s \dots s}_{n}$.)

Since the operators of the signature are known, we can specialise the above rule 4.1 for each operator, i.e. we can consider the set of inference rules:

$$\left\{ \frac{t_i \in T_{\Sigma,s_i} \quad i = 1,\dots,n}{f(t_1,\dots,t_n) \in T_{\Sigma,s}} \right\}_{f\in\Sigma_{s_1\dots s_n,s}} \tag{4.2}$$

Note that, as special case of the above inference rule, for constants $a \in \Sigma_{\varepsilon,s}$ we have:

$$\frac{}{a \in T_{\Sigma,s}} \tag{4.3}$$

Example 4.7 (IMP Signature). In the case of IMP, we have $S = \{Aexp, Bexp, Com\}$ and then we have an operation for each production in the grammar.

For example, the sequential composition of commands ";" corresponds to the binary infix operator $(-;-) \in \Sigma_{ComCom,Com}$.

Similarly the equality expression is built using the operator $(- = -) \in \Sigma_{AexpAexp,Bexp}$.

By abusing the notation, we often write *Com* for $T_{\Sigma,Com}$ (respectively, *Aexp* for $T_{\Sigma,Aexp}$ and *Bexp* for $T_{\Sigma,Bexp}$).

Then, we have inference rules instances such as:

$$\frac{}{skip \in Com} \qquad \frac{skip \in Com \quad x := a \in Com}{skip\,;x := a \in Com}$$

The programs we consider are (well-formed) terms over a suitable signature Σ (possibly many-sorted). Therefore it is useful to define a well-founded containment relation between a term and its subterms. For example, we will exploit this relation when dealing with structural induction in Section 4.1.5.

Example 4.8 (Terms and subterms). For any n-ary function symbol $f \in \Sigma_n$ and terms $t_1,\ldots,t_n$, we let:

$$t_i \prec f(t_1,\ldots,t_n) \qquad i = 1,\ldots,n$$

The idea is that a term t_i precedes (according to $\prec$, i.e. it is less than) any term that contains it as a subterm (e.g. as an argument).

As a concrete example, let us consider the signature Σ with $\Sigma_0 = \{c\}$ and $\Sigma_2 = \{f\}$. Then, we have, e.g.:

$$c \prec f(c,c) \prec f(f(c,c),c) \prec f(f(f(c,c),c),f(c,c))$$

If we look at terms as trees (function symbols as nodes with one children for each argument and constants as leaves), then we can observe that whenever $s \prec t$ the depth of s is strictly less than the depth of t. Therefore any descending chain is finite (the length is at most the depth of the first term of the chain). Moreover, in the particular case above, c is the only constant and therefore the only minimal element.

Example 4.9 (Lexicographic order). A quite common (well-founded) relation is the so-called lexicographic order. The idea is to have elements that are strings over a given ordered alphabet and to compare them symbol-by-symbol, from the leftmost to the rightmost: as soon as we find a symbol in one string that precedes the symbol in the same position of the other string, then we assume that the former string precedes the latter (independently from the remaining symbols of the two strings).

As a concrete example, let us consider the set of all pairs $\langle n,m\rangle$ of natural numbers ordered by immediate precedence. The lexicographic order relation is defined as (see Figure 4.2):

- $\forall n,m,k.\ (\langle n,m\rangle \prec \langle n+1,k\rangle)$
- $\forall n,m.\ (\langle n,m\rangle \prec \langle n,m+1\rangle)$

$$\begin{tikzcd}
\vdots & \vdots & \vdots & \\
\langle 2,0\rangle \arrow[r] & \langle 2,1\rangle \arrow[r] & \langle 2,2\rangle & \cdots \\
\langle 1,0\rangle \arrow[r] \arrow[u] \arrow[ur] \arrow[urr] & \langle 1,1\rangle \arrow[r] \arrow[u] \arrow[ul] \arrow[ur] & \langle 1,2\rangle \arrow[u] \arrow[ul] \arrow[ull] & \cdots \\
\langle 0,0\rangle \arrow[r] \arrow[u] \arrow[ur] \arrow[urr] & \langle 0,1\rangle \arrow[r] \arrow[u] \arrow[ul] \arrow[ur] & \langle 0,2\rangle \arrow[u] \arrow[ul] \arrow[ull] & \cdots
\end{tikzcd}$$

Fig. 4.2: Graph of the lexicographic order relation over pairs of natural numbers.

By Theorem 4.2, the relation $\prec$ is well-founded if and only if its transitive closure is such. Note that the relation $\prec^+$ has no cycle and any descending chain is bound by the only minimal element $\langle 0,0\rangle$. For example, we have:

$$\langle 5,1\rangle \succ^+ \langle 4,25\rangle \succ^+ \langle 3,100\rangle \succ^+ \langle 3,14\rangle \succ^+ \langle 2,1\rangle \succ^+ \langle 1,1000\rangle \succ^+ \langle 0,0\rangle$$

It is worth to note that any element $\langle n,m\rangle$ with $n \geq 1$ is preceded by infinitely many elements (e.g., $\forall k.\ \langle 0,k\rangle \prec \langle 1,0\rangle$) and it can be the first element of infinitely many (finite) descending chains (of unbounded length).

Still, given any nonempty set $Q \subseteq \mathbb{N}\times\mathbb{N}$, it is easy to find a minimal element $m \in Q$, namely such that $\forall b \prec^+ m.\ b \notin Q$. In fact, we can just take $m = \langle m_1, m_2\rangle$, where m_1 is the minimum (w.r.t. the usual less-than relation over natural numbers) of the set $Q_1 = \{n_1 | \langle n_1,n_2\rangle \in Q\}$ and m_2 is the minimum of the set $Q_2 = \{n_2 | \langle m_1,n_2\rangle \in Q\}$. Note that Q_1 is nonempty because Q is such by hypothesis, and Q_2 is nonempty because $m_1 \in Q_1$ and therefore there must exists at least one pair $\langle m_1,n_2\rangle \in Q$ for some n_2. Thus

$$\langle m_1 = min\{n_1 | \langle n_1,n_2\rangle \in Q\}, min\{n_2 | \langle m_1,n_2\rangle \in Q\}\rangle$$

is a (the only) minimal element of Q. By Lemma 4.1 the relation $\prec^+$ is well-founded and so is $\prec$ (by Theorem 4.2).

4.1.2 Noether Induction

Theorem 4.5. *Let $\prec$ be a well-founded relation over the set A and let P be a unary predicate over A. Then:*

$$(\forall a \in A.\ (\forall b \prec a.\ P(b)) \Rightarrow P(a)) \quad \Leftrightarrow \quad \forall a \in A.\ P(a)$$

Proof. We prove the two implications separately:

$\Rightarrow$) We proceed by contradiction by assuming $\neg(\forall a \in A.P(a))$, i.e., that $\exists a \in A.\ \neg P(a)$. Let us consider the nonempty set $Q = \{\ a \in A \mid \neg P(a)\ \}$ of all those elements a in A for which $P(a)$ is false. Since $\prec$ is well-founded, we know by Lemma 4.1 that there is a minimal element $m \in Q$. Obviously $\neg P(m)$ (otherwise m cannot be in Q). Since m is minimal in Q, then $\forall b \prec m.b \notin Q$, i.e., $\forall b \prec m.P(b)$. But this leads to a contradiction, because by hypothesis we have $\forall a \in A.(\forall b \prec a.P(b)) \rightarrow P(a)$ and instead the predicate $(\forall b \prec m.P(b)) \rightarrow P(m)$ is false. Therefore Q must be empty and $\forall a \in A.P(a)$ must hold.

$\Leftarrow$) We observe that if $\forall a.P(a)$ then $(\forall b \prec a.\ P(b)) \rightarrow P(a)$ is true for any a because the premise $(\forall b \prec a.\ P(b))$ is not relevant (the conclusion of the implication is true). □

From the first implication, it follows the validity of the following induction principle.

Definition 4.10 (Noether induction). Let $\prec$ be a well-founded relation over the set A and let P be a unary predicate over A. Then the following inference rule is called *Noether induction*.

$$\frac{\forall a \in A.\ (\forall b \prec a.\ P(b)) \Rightarrow P(a)}{\forall a \in A.\ P(a)} \tag{4.4}$$

We call a *base case* any element $a \in A$ such that the set of its predecessors $\{\, b \in A \mid b \prec a \,\}$ is empty.

4.1.3 Weak Mathematical Induction

The principle of weak mathematical induction is a special case of Noether induction that is frequently used to prove formulas over the set on natural numbers: we take

$$A = \mathbb{N} \qquad n \prec m \Leftrightarrow m = n + 1$$

In this case:

- if we take $a = 0$ then $(\forall b \prec a.\ P(b)) \Rightarrow P(a)$ amounts to $P(0)$, because there is no $b \in \mathbb{N}$ such that $b \prec 0$;
- if we take $a = n + 1$ for some $n \in \mathbb{N}$, then $(\forall b \prec a.\ P(b)) \Rightarrow P(a)$ amounts to $P(n) \Rightarrow P(n+1)$.

In other words, to prove that $P(n)$ holds for any $n \in \mathbb{N}$ we can just prove that:

- $P(0)$ holds (base case), and
- that, given a generic $n \in \mathbb{N}$, $P(n+1)$ holds whenever $P(n)$ holds (inductive case).

Definition 4.11 (Weak mathematical induction).

$$\frac{P(0) \qquad \forall n \in \mathbb{N}.\ (P(n) \Rightarrow P(n+1))}{\forall n \in \mathbb{N}.\ P(n)} \tag{4.5}$$

The weak mathematical induction principle is helpful, because it allows us to exploit the hypothesis $P(n)$ when proving $P(n+1)$.

4.1.4 Strong Mathematical Induction

The principle of strong mathematical induction extends the weak one by strengthening the hypotheses under which $P(n+1)$ is proved to hold. We take:

$$A = \mathbb{N} \qquad n \prec m \Leftrightarrow \exists k \in \mathbb{N}.\ m = n+k+1$$

In this case:

- if we take $a = 0$ then $(\forall b \prec a.\ P(b)) \Rightarrow P(a)$ amounts to $P(0)$, as for the case of weak mathematical induction;
- if we take $a = n+1$ for some $n \in \mathbb{N}$, then $(\forall b \prec a.\ P(b)) \Rightarrow P(a)$ amounts to $(P(0) \wedge P(1) \wedge \cdots \wedge P(n)) \Rightarrow P(n+1)$, i.e., using a more concise notation to $(\forall i \leq n.P(i)) \Rightarrow P(n+1)$.

In other words, to prove that $P(n)$ holds for any $n \in \mathbb{N}$ we can just prove that:

- $P(0)$ holds, and
- that, given a generic $n \in \mathbb{N}$, $P(n+1)$ holds whenever $P(i)$ holds for all $i = 0,...,n$.

Definition 4.12 (Strong mathematical induction).

$$\frac{P(0) \qquad \forall n \in \mathbb{N}.\ (\forall i \leq n.P(i)) \Rightarrow P(n+1)}{\forall n \in \mathbb{N}.\ P(n)} \tag{4.6}$$

The adjective "strong" comes from the fact that for proving $P(n+1)$ we can now exploit the *stronger* hypothesis $P(0) \wedge P(1) \wedge ... \wedge P(n)$ instead of just $P(n)$.

4.1.5 Structural Induction

The principle of structural induction is a special instance of Noether induction for proving properties over the set of terms generated by a given signature. Here, the order relation binds a term to its subterms.

Structural induction takes T_Σ as set of elements and subterm-term relation as well-founded relation:

$$A = T_\Sigma \qquad t_i < f(t_1,\ldots,t_n) \quad i = 1,\ldots,n$$

Definition 4.13 (Structural induction).

$$\frac{\forall t \in T_\Sigma.\ (\forall t' < t.\ P(t')) \Rightarrow P(t)}{\forall t \in T_\Sigma.\ P(t)} \tag{4.7}$$

By exploiting the definition of the well-founded subterm relation, we can expand the above principle as the rule

$$\frac{\forall f \in \Sigma_{s_1 \ldots s_n, s}.\ \forall t_1 \in T_{\Sigma, s_1} \ldots \forall t_n \in T_{\Sigma, s_n}.\ (P(t_1) \wedge \ldots \wedge P(t_n)) \Rightarrow P(f(t_1, \ldots, t_n))}{\forall t \in T_\Sigma.\ P(t)}$$

An easy link can be established w.r.t. mathematical induction by taking a unique sort, a constant 0 and a unary operation *succ* (i.e., $\Sigma = \Sigma_0 \cup \Sigma_1$ with $\Sigma_0 = \{0\}$ and $\Sigma_1 = \{succ\}$). Then, the structural induction rule would become:

$$\frac{P(0) \quad \forall t.\ (P(t) \Rightarrow P(succ(t)))}{\forall t.\ P(t)}$$

Example 4.10. Let us consider the grammar of IMP arithmetic expressions:

$$a \quad ::= \quad n \quad | \quad x \quad | \quad a_0 + a_1 \quad | \quad a_0 - a_1 \quad | \quad a_0 \times a_1$$

How do we exploit structural induction to prove that a property $P(\cdot)$ holds for all arithmetic expressions a? (Namely, we want to prove that $\forall a \in Aexp.\ P(a)$.) The structural induction rule is:

$$\frac{\begin{array}{c}\forall n.\ P(n) \qquad \forall x.\ P(x) \qquad \forall a_0, a_1.\ (P(a_0) \wedge P(a_1) \Rightarrow P(a_0 + a_1)) \\ \forall a_0, a_1.\ (P(a_0) \wedge P(a_1) \Rightarrow P(a_0 - a_1)) \qquad \forall a_0, a_1.\ (P(a_0) \wedge P(a_1) \Rightarrow P(a_0 \times a_1))\end{array}}{\forall a.\ P(a)}$$

Essentially, to prove that $\forall a \in Aexp.\ P(a)$, we just need to show that the property holds for any production, i.e., we need to prove that all of the following hold:

- $P(n)$ holds for any integer n;
- $P(x)$ holds for any identifier x;
- $P(a_0 + a_1)$ holds whenever both $P(a_0)$ and $P(a_1)$ hold;
- $P(a_0 - a_1)$ holds whenever both $P(a_0)$ and $P(a_1)$ hold;
- $P(a_0 \times a_1)$ holds whenever both $P(a_0)$ and $P(a_1)$ hold.

Example 4.11 (Termination of arithmetic expressions). Let us consider the case of arithmetic expressions seen above and prove that the evaluation of expressions always terminates (a property that is also called *normalisation*):[1]

$$\forall a \in Aexp, \forall \sigma \in \Sigma, \exists m \in \mathbb{N}.\ \langle a, \sigma \rangle \to m$$

[1] We recall that the (overloaded) symbol Σ stands here for the set of memories and not for a generic signature.

In this case we let

$$P(a) \stackrel{\text{def}}{=} \forall \sigma \in \Sigma, \exists m \in \mathbb{N}.\ \langle a, \sigma \rangle \to m.$$

We prove that $\forall a \in Aexp.\ P(a)$ by structural induction. This amounts to prove that

- $P(n) \stackrel{\text{def}}{=} \forall \sigma \in \Sigma, \exists m \in \mathbb{N}.\ \langle n, \sigma \rangle \to m$ holds for any integer n. Trivially, by applying rule (num) we take $m = n$ and we are done.
- $P(x) \stackrel{\text{def}}{=} \forall \sigma \in \Sigma, \exists m \in \mathbb{N}.\ \langle x, \sigma \rangle \to m$ holds for any location x. Trivially, by applying rule (ide) we take $m = \sigma(x)$ and we are done.
- $P(a_0) \wedge P(a_1) \Rightarrow P(a_0 + a_1)$ for any arithmetic expressions a_0 and a_1. We assume

  $$P(a_0) \stackrel{\text{def}}{=} \forall \sigma \in \Sigma, \exists m_0 \in \mathbb{N}.\ \langle a_0, \sigma \rangle \to m_0$$
  $$P(a_1) \stackrel{\text{def}}{=} \forall \sigma \in \Sigma, \exists m_1 \in \mathbb{N}.\ \langle a_1, \sigma \rangle \to m_1.$$

  We want to prove that

  $$P(a_0 + a_1) \stackrel{\text{def}}{=} \forall \sigma \in \Sigma, \exists m \in \mathbb{N}.\ \langle a_0 + a_1, \sigma \rangle \to m.$$

  Take a generic $\sigma \in \Sigma$. We want to find $m \in \mathbb{N}$ such that $\langle a_0 + a_1, \sigma \rangle \to m$. By applying rule (sum) we can take $m = m_0 + m_1$ if we prove that $\langle a_0, \sigma \rangle \to m_0$ and $\langle a_1, \sigma \rangle \to m_1$. But by inductive hypothesis we know that such m_0 and m_1 exist and we are done.
- $P(a_0) \wedge P(a_1) \Rightarrow P(a_0 - a_1)$ for any arithmetic expressions a_0 and a_1. The proof is analogous to the previous case and thus omitted.
- $P(a_0) \wedge P(a_1) \Rightarrow P(a_0 \times a_1)$ for any arithmetic expressions a_0 and a_1. The proof is analogous to the previous case and thus omitted.

Example 4.12 (Determinism of arithmetic expressions). Let us consider again the case of IMP arithmetic expressions and prove that their evaluation is deterministic:

$$\forall a \in Aexp, \forall \sigma \in \Sigma, \forall m, m' \in \mathbb{N}.\ (\langle a, \sigma \rangle \to m \wedge \langle a, \sigma \rangle \to m') \Rightarrow m = m'$$

In other words, we want to show that given any arithmetic expression a and any memory σ the evaluation of a in σ will always return exactly one value. We let

$$P(a) \stackrel{\text{def}}{=} \forall \sigma \in \Sigma, \forall m, m' \in \mathbb{N}.\ (\langle a, \sigma \rangle \to m \wedge \langle a, \sigma \rangle \to m') \Rightarrow m = m'$$

We proceed by structural induction.

$a = n$) We want to prove that

$$P(n) \stackrel{\text{def}}{=} \forall \sigma, m, m'.\ (\langle n, \sigma \rangle \to m \wedge \langle n, \sigma \rangle \to m') \Rightarrow m = m'$$

holds. Let us take generic σ, m, m'. We assume the premises $\langle n, \sigma \rangle \to m$ and $\langle n, \sigma \rangle \to m'$ and prove that $m = m'$. In fact, there is only one rule that can be used to evaluate an integer number, and it always returns the same value. Therefore $m = n = m'$.

$a = x$) We want to prove that

$$P(x) \overset{\text{def}}{=} \forall \sigma, m, m'.\ (\langle x, \sigma\rangle \to m \wedge \langle x, \sigma\rangle \to m') \Rightarrow m = m'$$

holds. We assume the premises $\langle x, \sigma\rangle \to m$ and $\langle x, \sigma\rangle \to m'$ and prove that $m = m'$. Again, there is only one rule that can be applied, whose outcome depends on σ. Since σ is the same in both cases, $m = \sigma(x) = m'$.

$a = a_0 + a_1$) We assume the inductive hypotheses

$$P(a_0) \overset{\text{def}}{=} \forall \sigma, m_0, m_0'.\ (\langle a_0, \sigma\rangle \to m_0 \wedge \langle a_0, \sigma\rangle \to m_0') \Rightarrow m_0 = m_0'$$
$$P(a_1) \overset{\text{def}}{=} \forall \sigma, m_1, m_1'.\ (\langle a_1, \sigma\rangle \to m_1 \wedge \langle a_1, \sigma\rangle \to m_1') \Rightarrow m_1 = m_1'$$

and we want to prove that $P(a_0 + a_1)$, i.e., that:

$$\forall \sigma, m, m'.\ (\langle a_0 + a_1, \sigma\rangle \to m \wedge \langle a_0 + a_1, \sigma\rangle \to m') \Rightarrow m = m'$$

We assume the premises $\langle a_0 + a_1, \sigma\rangle \to m$ and $\langle a_0 + a_1, \sigma\rangle \to m'$ and prove that $m = m'$. By the first premise, it must be $m = m_0 + m_1$ for some m_0, m_1 such that $\langle a_0, \sigma\rangle \to m_0$ and $\langle a_1, \sigma\rangle \to m_1$, because there is only one rule applicable to $a_0 + a_1$; analogously, by the second premise, we must have $m' = m_0' + m_1'$ for some m_0', m_1' such that $\langle a_0, \sigma\rangle \to m_0'$ and $\langle a_1, \sigma\rangle \to m_1$. By inductive hypothesis $P(a_0)$ we know that $m_0 = m_0'$ and by $P(a_1)$ we have $m_1 = m_1'$. Thus, $m = m_0 + m_1 = m_0' + m_1' = m'$ and thus $P(a_0 + a_1)$ holds.

The remaining cases for $a = a_0 - a_1$ and $a = a_0 \times a_1$ follow exactly the same pattern as that of $a = a_0 + a_1$.

4.1.6 Induction on Derivations

We can define an induction principle over the set of derivations of a logical system. See Definitions 2.1 and 2.4 for the notion of inference rule and of derivation.

Definition 4.14 (Immediate sub-derivation). We say that d' is an *immediate sub-derivation* of d, or simply a sub derivation of d, written $d' \prec d$, if and only if d has the form $(\{d_1, \ldots, d_n\}\ /\ y)$ with $d_1 \Vdash_R x_1, \ldots, d_n \Vdash_R x_n$ and $(\{x_1, \ldots, x_n\}\ /\ y) \in R$ (i.e., $d \Vdash_R y$) and $d' = d_i$ for some $1 \leq i \leq n$.

Example 4.13 (Immediate sub-derivation). Let us consider the derivation

$$\dfrac{\dfrac{}{\langle 1,\sigma\rangle \to 1}\,\text{num} \qquad \dfrac{}{\langle 2,\sigma\rangle \to 2}\,\text{num}}{\langle 1+2,\sigma\rangle \to 1+2=3}\,\text{sum}$$

the two derivations

$$\dfrac{}{\langle 1,\sigma\rangle \to 1}\,\text{num} \qquad \dfrac{}{\langle 2,\sigma\rangle \to 2}\,\text{num}$$

are immediate sub-derivations of the derivation that exploits rule (sum).

We can derive the notion of proper sub-derivations out of immediate ones.

Definition 4.15 (Proper sub-derivation). We say that d' is a *proper sub-derivation* of d if and only if $d' \prec^+ d$.

Note that both $\prec$ and $\prec^+$ are well-founded, so they can be used in proofs by induction.

For example, the induction principle based on immediate sub-derivation can be phrased as follows.

Definition 4.16 (Induction on derivations). Let R be a set of inference rules and D the set of derivations defined on R, then:

$$\frac{\forall \{x_1,\dots,x_n\}/y \in R.\ (\forall d_i \Vdash_R x_i.\ P(d_1) \wedge \dots \wedge P(d_n)) \Rightarrow P(\{d_1,\dots,d_n\}/y)}{\forall d \in D.\ P(d)} \tag{4.8}$$

(Note that $d_1,\dots,d_n$ are derivation for $x_1,\dots,x_n$).

4.1.7 Rule Induction

The last kind of induction principle we shall consider applies to sets of elements that are defined by means of inference rules: we have a set of inference rules that establish which elements belong to the set (i.e. the theorems of the logical system) and we need to prove that the application of any such rule will not compromise the validity of a given predicate.

Remind tha a rule has the form $(\varnothing/y)$ if it is an axiom, or $(\{x_1,\dots,x_n\}/y)$ otherwise. Given a set R of such rules, the *set of theorems of* R is defined as

$$I_R = \{y \mid \Vdash_R y\}$$

The rule induction principle states that to show that the property P holds for all elements of I_R, we can prove that:

- $P(y)$ holds for any axiom $\varnothing/y \in R$;
- for any other rule $\{x_1,\dots,x_n\}/y \in R$ we have $(\forall 1 \le i \le n. x_i \in I_R \wedge P(x_i)) \Rightarrow P(y)$.

Definition 4.17 (Rule induction). Let R be a logical system. The principle of rule induction is:

$$\frac{\forall (X/y) \in R.\ (X \subseteq I_R\ \wedge\ \forall x \in X.\ P(x)) \Rightarrow P(y)}{\forall x \in I_R. P(x)} \tag{4.9}$$

Note that in many cases we will use the simpler but less powerful rule

$$\frac{\forall (X/y) \in R.\ (\forall x \in X \cdot P(x)) \Rightarrow P(y)}{\forall x \in I_R. P(x)} \tag{4.10}$$

In fact, if the latter applies, also the former does, since the implication in the premise must be proved in fewer cases: only for rules X/y such that all the formulas in X are theorems. However, usually it is difficult to take advantage of the restriction.

Example 4.14 (Determinism of IMP commands). We have seen in Example 4.12 that structural induction can be conveniently used to prove that the evaluation of arithmetic expressions is deterministic. Formally, we were proving the predicate $P(\cdot)$ over arithmetic expressions defined as

$$P(a) \overset{\text{def}}{=} \forall \sigma. \forall m, m'.\ \langle a, \sigma \rangle \to m \wedge \langle a, \sigma \rangle \to m' \Rightarrow m = m'$$

While the case of boolean expressions is completely analogous, for commands we cannot use the same proof strategy, because structural induction cannot deal with the rule (whtt). In this example, we show that rule induction provides a convenient strategy to solve the problem.

Let us consider the following predicate over theorems, i.e., statements of the form $\langle c, \sigma \rangle \to \sigma'$:

$$Q(\langle c, \sigma \rangle \to \sigma') \overset{\text{def}}{=} \forall \sigma_1 \in \Sigma.\ \langle c, \sigma \rangle \to \sigma_1 \Rightarrow \sigma' = \sigma_1$$

We proceed by rule induction:

rule skip): we want to show that

$$Q(\langle \mathbf{skip}, \sigma \rangle \to \sigma) \overset{\text{def}}{=} \forall \sigma_1.\ \langle \mathbf{skip}, \sigma \rangle \to \sigma_1 \Rightarrow \sigma_1 = \sigma$$

which is obvious because there is only one rule applicable to **skip**:

$$\langle \mathbf{skip}, \sigma \rangle \to \sigma_1 \quad \nwarrow_{\sigma_1 = \sigma} \quad \square$$

rule assign): assuming

$$\langle a, \sigma \rangle \to m$$

we want to show that

$$Q(\langle x := a, \sigma \rangle \to \sigma[^m/_x]) \overset{\text{def}}{=} \forall \sigma_1.\ \langle x := a, \sigma \rangle \to \sigma_1 \Rightarrow \sigma_1 = \sigma[^m/_x]$$

Let us take a generic memory σ_1 and assume the premise $\langle x := a, \sigma\rangle \to \sigma_1$ of the implication holds. We proceed goal oriented. We have:

$$\langle x := a, \sigma\rangle \to \sigma_1 \quad \nwarrow_{\sigma_1 = \sigma[^{m'}/_x]} \quad \langle a, \sigma\rangle \to m'$$

But we know that the evaluation of arithmetic expressions is deterministic and therefore $m' = m$ and $\sigma_1 = \sigma[^m/_x]$.

rule seq): assuming

$$Q(\langle c_0, \sigma\rangle \to \sigma'') \stackrel{\text{def}}{=} \forall \sigma_1''.\ \langle c_0, \sigma\rangle \to \sigma_1'' \Rightarrow \sigma'' = \sigma_1''$$
$$Q(\langle c_1, \sigma''\rangle \to \sigma') \stackrel{\text{def}}{=} \forall \sigma_1.\ \langle c_1, \sigma''\rangle \to \sigma_1 \Rightarrow \sigma' = \sigma_1$$

we want to show that

$$Q(\langle c_0; c_1, \sigma\rangle \to \sigma') \stackrel{\text{def}}{=} \forall \sigma_1.\ \langle c_0; c_1, \sigma\rangle \to \sigma_1 \Rightarrow \sigma_1 = \sigma'$$

We assume the premise $\langle c_0; c_1, \sigma\rangle \to \sigma_1$ and prove that $\sigma_1 = \sigma'$. We have:

$$\langle c_0; c_1, \sigma\rangle \to \sigma_1 \quad \nwarrow \quad \langle c_0, \sigma\rangle \to \sigma_1'', \quad \langle c_1, \sigma_1''\rangle \to \sigma_1$$

But now we can apply the first inductive hypotheses:

$$Q(\langle c_0, \sigma\rangle \to \sigma'') \stackrel{\text{def}}{=} \forall \sigma_1''.\ \langle c_0, \sigma\rangle \to \sigma_1'' \Rightarrow \sigma_1'' = \sigma''$$

to conclude that $\sigma_1'' = \sigma''$, which together with the second inductive hypothesis

$$Q(\langle c_1, \sigma''\rangle \to \sigma') \stackrel{\text{def}}{=} \forall \sigma_1.\ \langle c_1, \sigma''\rangle \to \sigma_1 \Rightarrow \sigma_1 = \sigma'$$

allow us to conclude that $\sigma_1 = \sigma'$.

rule iftt): assuming

$$\langle b, \sigma\rangle \to \mathbf{true}$$
$$Q(\langle c_0, \sigma\rangle \to \sigma') \stackrel{\text{def}}{=} \forall \sigma_1.\ \langle c_0, \sigma\rangle \to \sigma_1 \Rightarrow \sigma_1 = \sigma'$$

we want to show that

$$Q(\langle \mathbf{if}\ b\ \mathbf{then}\ c_0\ \mathbf{else}\ c_1, \sigma\rangle \to \sigma') \stackrel{\text{def}}{=} \forall \sigma_1.\ \langle \mathbf{if}\ b\ \mathbf{then}\ c_0\ \mathbf{else}\ c_1, \sigma\rangle \to \sigma_1 \Rightarrow \sigma_1 = \sigma'$$

Since $\langle b, \sigma\rangle \to \mathbf{true}$ and the evaluation of boolean expressions is deterministic, we have:

$$\langle \mathbf{if}\ b\ \mathbf{then}\ c_0\ \mathbf{else}\ c_1, \sigma\rangle \to \sigma_1 \quad \nwarrow \quad \langle c_0, \sigma\rangle \to \sigma_1$$

But then, exploiting the inductive hypothesis

$$Q(\langle c_0,\sigma\rangle \to \sigma') \stackrel{\text{def}}{=} \forall \sigma_1.\ \langle c_0,\sigma\rangle \to \sigma_1 \Rightarrow \sigma_1 = \sigma'$$

we can conclude that $\sigma_1 = \sigma'$.

rule ifff): omitted (it is analogous to the previous case).

rule whff): assuming

$$\langle b,\sigma\rangle \to \textbf{false}$$

we want to show that

$$Q(\langle \textbf{while}\ b\ \textbf{do}\ c,\sigma\rangle \to \sigma) \stackrel{\text{def}}{=} \forall \sigma_1.\ \langle \textbf{while}\ b\ \textbf{do}\ c,\sigma\rangle \to \sigma_1 \Rightarrow \sigma_1 = \sigma$$

Since $\langle b,\sigma\rangle \to \textbf{false}$ and the evaluation of boolean expressions is deterministic, we have:

$$\langle \textbf{while}\ b\ \textbf{do}\ c,\sigma\rangle \to \sigma_1 \quad \nwarrow_{\sigma_1=\sigma} \quad \square$$

rule whtt): assuming

$$\langle b,\sigma\rangle \to \textbf{true}$$
$$Q(\langle c,\sigma\rangle \to \sigma'') \stackrel{\text{def}}{=} \forall \sigma_1''.\ \langle c,\sigma\rangle \to \sigma_1'' \Rightarrow \sigma_1'' = \sigma''$$
$$Q(\langle \textbf{while}\ b\ \textbf{do}\ c,\sigma''\rangle \to \sigma') \stackrel{\text{def}}{=} \forall \sigma_1.\ \langle \textbf{while}\ b\ \textbf{do}\ c,\sigma''\rangle \to \sigma_1 \Rightarrow \sigma_1 = \sigma'$$

we want to show that

$$Q(\langle \textbf{while}\ b\ \textbf{do}\ c,\sigma\rangle \to \sigma') \stackrel{\text{def}}{=} \forall \sigma_1.\ \langle \textbf{while}\ b\ \textbf{do}\ c,\sigma\rangle \to \sigma_1 \Rightarrow \sigma_1 = \sigma'$$

Since $\langle b,\sigma\rangle \to \textbf{true}$ and the evaluation of boolean expressions is deterministic, we have:

$$\langle \textbf{while}\ b\ \textbf{do}\ c,\sigma\rangle \to \sigma_1 \quad \nwarrow \quad \langle c,\sigma\rangle \to \sigma_1'',\ \langle \textbf{while}\ b\ \textbf{do}\ c,\sigma_1''\rangle \to \sigma_1$$

But now we can apply the first inductive hypotheses:

$$Q(\langle c,\sigma\rangle \to \sigma'') \stackrel{\text{def}}{=} \forall \sigma_1''.\ \langle c,\sigma\rangle \to \sigma_1'' \Rightarrow \sigma_1'' = \sigma''$$

to conclude that $\sigma_1'' = \sigma''$, which together with the second inductive hypothesis

$$Q(\langle \textbf{while}\ b\ \textbf{do}\ c,\sigma''\rangle \to \sigma') \stackrel{\text{def}}{=} \forall \sigma_1.\ \langle \textbf{while}\ b\ \textbf{do}\ c,\sigma''\rangle \to \sigma_1 \Rightarrow \sigma_1 = \sigma'$$

allow us to conclude that $\sigma_1 = \sigma'$.

4.2 Well-founded Recursion

We conclude this chapter by presenting the concept of well-founded recursion. A recursive definition of a function f is *well-founded* when the recursive calls to f take as arguments values that are smaller w.r.t. the ones taken by the defined function (according to a suitable well-founded relation). A special class of functions defined on natural numbers according to the principle of well-founded recursion is that of *primitive recursive functions*.

Definition 4.18 (Primitive recursive functions). The primitive recursive functions are those (n-ary) functions over natural numbers obtained according to (any finite application of) the following rules:

zero: The constant 0 is primitive recursive.

succ.: The successor function $s : \mathbb{N} \to \mathbb{N}$ with $s(n) = n+1$ is primitive recursive.

proj.: For any $i, k \in \mathbb{N}, 1 \leq i \leq n$, the projection functions $\pi_i^k : \mathbb{N}^k \to \mathbb{N}$ with

$$\pi_i^k(n_1, ..., n_k) = n_i$$

are primitive recursive.

comp.: Given a k-ary primitive recursive function $f : \mathbb{N}^k \to \mathbb{N}$, and k primitive recursive functions $g_1, ..., g_k : \mathbb{N}^m \to \mathbb{N}$ of arity m, the m-ary function h obtained by composing f with $g_1, ..., g_k$ as shown below is primitive recursive:

$$h(n_1, ..., n_m) \stackrel{\text{def}}{=} f(g_1(n_1, ..., n_m), ..., g_k(n_1, ..., n_m))$$

pr.rec.: Given a k-ary primitive recursive function $f : \mathbb{N}^k \to \mathbb{N}$ and a $(k+2)$-ary primitive recursive function $g : \mathbb{N}^{k+2} \to \mathbb{N}$, the $(k+1)$-ary function $h : \mathbb{N}^{k+1} \to \mathbb{N}$ defined as the primitive recursion of f and g below is primitive recursive:

$$\begin{aligned} h(0, n_1, ..., n_k) &= f(x_1, ..., x_k) \\ h(s(n), n_1, ..., n_k) &= g(n, h(n, n_1, \ldots, n_k), n_1, \ldots, n_k) \end{aligned}$$

Note that $\pi_1^1 : \mathbb{N} \to \mathbb{N}$ is the usual identity function. It can be proved that every primitive recursive function is total and computable.

Example 4.15. Addition can be recursively defined with the rules:

$$\begin{aligned} add(0, m) &\stackrel{\text{def}}{=} m \\ add(n+1, m) &\stackrel{\text{def}}{=} add(n, m) + 1. \end{aligned}$$

This does not fit immediately into the above scheme of primitive recursive functions, but we can rephrase the definition as:

$$\begin{aligned} add(0,n_1) &\stackrel{\text{def}}{=} \pi_1^1(n_1) \\ add(s(n),n_1) &\stackrel{\text{def}}{=} s(\pi_2^3(n, add(n,n_1), n_1)) \end{aligned}$$

In the primitive recursive style, *add* plays the role of h, the identity function π_1^1 plays the role of f and the composition of s with π_2^3 plays the role of g (so that it receives the unnecessary arguments n and n_1).

Let us make the well-founded recursion more precise.

Definition 4.19 (Set of predecessors). Given a well founded relation $\prec \subseteq A \times A$, the set of *predecessors* of a set $I \subseteq A$ is the set

$$\prec^{-1} I = \{ b \in A \mid \exists a \in I.\ b \prec a \}$$

We recall that for $I \subseteq A$ and $f : A \to B$, we denote by $f \restriction I$ the restriction of f to values in I, i.e., $f \restriction I : I \to B$ and $(f \restriction I)(a) = f(a)$ for any $a \in I$.

Theorem 4.6 (Well-founded recursion). *Let $\prec \subseteq A \times A$ be a well-founded relation over A. Let us consider a function F with $F(a,h) \in B$, where*

- $a \in A$
- $h :<^{-1} \{a\} \to B$ *(i.e., h is a function that has the set of predecessors of a as a domain and B as a codomain)*

Then, there exists one and only one function $f : A \to B$ which satisfies the equation

$$\forall a \in A.\ f(a) = F(a, f \restriction \prec^{-1} \{a\})$$

Proof. The proof is divided in two parts: 1) we first demonstrate that if such a function f exists, then it is unique; and 2) we prove its existence.

1. The unicity property follows if we can prove the predicate $\forall a.\ P(a)$, where

$$\begin{aligned} P(x) \stackrel{\text{def}}{=} &\ (\ \forall y \prec^* x.\ (\ f(y) = F(y, f \restriction \prec^{-1} \{y\}) \ \wedge\ g(y) = F(y, g \restriction \prec^{-1} \{y\})\ )\ ) \\ &\Rightarrow\ f(x) = g(x) \end{aligned}$$

   In fact, suppose there are two functions $f, g : A \to B$ such that:

$$\begin{aligned} \forall a \in A.\ f(a) &= F(a, f \restriction \prec^{-1} \{a\}) \\ \forall a \in A.\ g(a) &= F(a, g \restriction \prec^{-1} \{a\}) \end{aligned}$$

   Clearly, for any $a \in A$ the premise

$$(\ \forall y \prec^* a.\ (\ f(y) = F(y, f \restriction \prec^{-1} \{y\}) \ \wedge\ g(y) = F(y, g \restriction \prec^{-1} \{y\})\ )\ )$$

   is true and thus we can conclude $f(a) = g(a)$.
   The proof that $\forall a.\ P(a)$ goes by well-founded induction on $\prec$. For $a \in A$, we assume that $\forall b \prec a.\ P(b)$ and we want to prove $P(a)$. Suppose that

$$(\ \forall y \prec^* a.\ (\ f(y) = F(y, f \restriction \prec^{-1} \{y\}) \ \wedge\ g(y) = F(y, g \restriction \prec^{-1} \{y\})\ )\ )$$

We need to prove that $f(a) = g(a)$. For $b \prec a$ we must have $f(b) = g(b)$, because $P(b)$ holds by inductive hypothesis. Thus

$$f \restriction \prec^{-1} \{a\} = g \restriction \prec^{-1} \{a\}$$

and therefore

$$f(a) = F(y, f \restriction \prec^{-1} \{a\}) = F(y, g \restriction \prec^{-1} \{a\}) = g(a)$$

2. For the existence, we build a family of functions

$$\{f_a : \prec^{*-1} \{a\} \to B\}_{a \in A}$$

and then take $f \stackrel{\text{def}}{=} \bigcup_{a \in A} f_a$. The existence of the functions f_a is guaranteed by proving that the following property holds for all $x \in A$:

$$Q(x) \stackrel{\text{def}}{=} \exists f_x : \prec^{*-1} \{x\} \to B.\ \forall y \prec^* x.\ f_x(y) = F(y, f_x \restriction \prec^{-1} \{y\})$$

The proof goes by well-founded recursion. We assume $\forall b \prec a.\ Q(b)$ and prove that $Q(a)$ holds. Let $b \prec a$ and $f_b : \prec^{*-1} \{b\} \to B$ be the function such that

$$\forall y \prec^* b.\ f_b(y) = F(y, f_b \restriction \prec^{-1} \{y\}).$$

We build a relation h defined by

$$h \stackrel{\text{def}}{=} \bigcup_{b \prec a} f_b$$

Now for any $b \prec a$ there is at least one pair of the form $(b, c) \in h$, because $b \prec^* b$. By the uniqueness property proved before, we have that such a pair is unique. Hence h satisfies the function property. Finally, we let

$$f_a \stackrel{\text{def}}{=} h \cup \{(a, F(a, h)\}$$

to get a function $f_a : \prec^{*-1} \{a\} \to B$ such that

$$\forall y \prec^* a.\ f_a(y) = F(y, f_a \restriction \prec^{-1} \{y\})$$

proving that $Q(a)$ holds. □

Theorem 4.6 guarantees that, if we (recursively) define f over any $a \in A$ only in terms of the predecessors of a, then f is uniquely determined on all a. Notice that F has a *dependent* type, since the type of its second argument depends on the value of its first argument.

In the following chapters we will exploit fixpoint theory to define the semantics of recursively defined functions. Well-founded recursion gives a simpler method, which however works only in the well-founded case.

Example 4.16 (Product as primitive recursion). Let us consider the Peano formula that defines the product of natural numbers

$$\begin{aligned} p(0,y) &\stackrel{\text{def}}{=} 0 \\ p(x+1,y) &\stackrel{\text{def}}{=} y + p(x,y) \end{aligned}$$

Let us write the definition in a slightly different way

$$\begin{aligned} p_y(0) &\stackrel{\text{def}}{=} 0 \\ p_y(x+1) &\stackrel{\text{def}}{=} y + p_y(x) \end{aligned}$$

Let us recast the Peano formula seen above to the formal scheme of well-founded recursion.

$$\begin{aligned} p_y(0) &\stackrel{\text{def}}{=} F_y(0, p_y \restriction \varnothing) = 0 \\ p_y(x+1) &\stackrel{\text{def}}{=} F_y(x+1, p_y \restriction \prec^{-1} \{x+1\}) = y + p_y(x) \end{aligned}$$

Example 4.17 (Structural recursion). Let us consider the signature Σ for binary trees $A = T_\Sigma$, where $\Sigma_0 = \{0,1,...\}$ and $\Sigma_2 = \mathsf{cons}$ (where $\mathsf{cons}(x,y)$ is the constructor for building a tree out of its left and right subtree). Take the well-founded relation $x_i \prec cons(x_1,x_2)$, $i = 1,2$. Let $B = \mathbb{N}$.

We want to compute the sum of the elements in the leaves of a binary tree. In Lisp-like notation:

$$sum(x) \stackrel{\text{def}}{=} \textbf{if } atom(x) \textbf{ then } x \textbf{ else } sum(car(x)) + sum(cdr(x))$$

where $atom(x)$ returns true if x is a leaf; $car(x)$ denotes the left subtree of x; $cdr(x)$ the right subtree of x. The same function defined in the structural recursion style is

$$\begin{aligned} sum(n) &\stackrel{\text{def}}{=} n \\ sum(\mathsf{cons}(x,y)) &\stackrel{\text{def}}{=} sum(x) + sum(y) \end{aligned}$$

or, according to the well-founded recursive scheme,

$$\begin{aligned} F(n, sum \restriction \varnothing) &\stackrel{\text{def}}{=} n \\ F(\mathsf{cons}(x,y), sum \restriction \{x,y\}) &\stackrel{\text{def}}{=} sum(x) + sum(y) \end{aligned}$$

For example, for $q \stackrel{\text{def}}{=} \mathsf{cons}(3, \mathsf{cons}(\mathsf{cons}(2,3),4))$ we have:

$$\begin{aligned} sum(q) &= sum(3) + sum(\mathsf{cons}(\mathsf{cons}(2,3),4)) \\ &= 3 + (sum(\mathsf{cons}(2,3)) + sum(4)) \\ &= 3 + ((sum(2) + sum(3)) + 4) \\ &= 3 + (2+3) + 4) \\ &= 12 \end{aligned}$$

Example 4.18 (Ackermann function). The Ackermann function is one of the earliest examples of a computable, total recursive function that is not primitive recursive: it grows faster than any such function. The Ackermann function $ack(z,x,y) = ack_y(z,x)$ is defined by well-founded recursion (exploiting the lexicographic order over pair of natural numbers) by letting

$$\begin{cases} ack(\ 0\ ,\ 0\ ,\ y\ ) = y \\ ack(\ 0\ ,\ x+1\ ,\ y\ ) = ack(0,x,y)+1 \\ ack(\ 1\ ,\ 0\ ,\ y\ ) = 0 \\ ack(\ z+2\ ,\ 0\ ,\ y\ ) = 1 \\ ack(\ z+1\ ,\ x+1\ ,\ y\ ) = ack(z, ack(z+1,x,y),y) \end{cases}$$

We have

$$\begin{cases} ack(0,0,y) = y \\ ack(0,x+1,y) = ack(0,x,y)+1 \end{cases} \quad \Rightarrow \quad ack(0,x,y) = y + x$$

$$\begin{cases} ack(1,0,y) = 0 \\ ack(1,x+1,y) = ack(0,ack(1,x,y),y) = ack(1,x,y)+y \end{cases} \quad \Rightarrow \quad ack(1,x,y) = y \times x$$

Intuitively, $ack(1,x,y)$ applies addition of y for x times.

$$\begin{cases} ack(2,0,y) = 1 \\ ack(2,x+1,y) = ack(1,ack(2,x,y),y) = ack(2,x,y) \times y \end{cases} \quad \Rightarrow \quad ack(2,x,y) = y^x$$

In other words, $ack(2,x,y)$ applies multiplication for y for x times. Likewise, $ack(3,x,y)$ applies exponentiation to the yth power for x times, and so on.

For example, we have:

$$\begin{aligned} ack(0,0,0) &= 0+0 = 0 \\ ack(1,1,1) &= 1 \times 1 = 1 \\ ack(2,2,2) &= 2^2 = 4 \\ ack(3,3,3) &= 3^{3^3} = 3^{27} \simeq 7.6 \times 10^{12} \end{aligned}$$

Problems

4.1. Consider the logical system R corresponding to the rules of the grammar:

$$S ::= aB \mid bA \qquad A ::= a \mid aS \mid bAA \qquad B ::= b \mid bS \mid aBB$$

where the well formed formulas are of the form $x \in L_X$, where X is either S or A or B and where x is a string on the alphabet $\{a,b\}$.

1. Write down explicitly the rules in R.
2. Prove by rule induction—in one direction—and by mathematical induction on the length of the strings—in the other direction—that the strings generated by S are all the nonempty strings with the same number of a's and b's (i.e., prove the formal predicate $P(x \in L_S) \stackrel{\text{def}}{=} x|_a = x|_b \neq 0$, where $x|_s$ denotes the number of occurrences of the symbol s in the string x), while A generates all the strings with an additional a (formally $P(x \in L_A) \stackrel{\text{def}}{=} x|_a = 1 + x|_b$) and B with an additional b.
3. Finally prove by induction on derivations that

$$P(d/(x \in L_X)) \stackrel{\text{def}}{=} |d| \leq |x|$$

i.e., the depth of any derivation d is smaller or equal that the length of the string x generated by it.

4.2. Define by well-founded recursion the function

$$locs : Com \longrightarrow \wp(\mathbf{Loc})$$

that, given a command, returns the set of locations that appear on the left-hand side of some assignment.

Then, prove that $\forall c \in Com, \forall \sigma, \sigma' \in \Sigma$

$$\langle c, \sigma \rangle \to \sigma' \qquad \text{implies} \qquad \forall x \notin locs(c).\, \sigma(x) = \sigma'(x).$$

4.3. Let w denote the IMP command

$$w \stackrel{\text{def}}{=} \textbf{while } x \neq 0 \textbf{ do } (x := x - 1 \,;\, y := y + 1).$$

Prove by rule induction that $\forall \sigma, \sigma' \in \Sigma$

$$\langle w, \sigma \rangle \to \sigma' \qquad \text{implies} \qquad \sigma(x) \geq 0 \wedge \sigma' = \sigma\left[{}^{\sigma(x)+\sigma(y)}/_{y},\ {}^{0}/_{x}\right].$$

4.4. Let R be a binary relation over the set A, i.e., $R \subseteq A \times A$. Let R^+, called the *transitive closure* of R, be the relation defined by the following two rules:

$$\frac{x R y}{x R^+ y} \qquad \frac{x R^+ y \quad y R^+ z}{x R^+ z}$$

1. Prove that for any x and y

$$xR^+y \quad \Leftrightarrow \quad \exists k > 0.\ \exists z_0,\dots,z_k.\ x = z_0 \ \wedge\ z_0 R z_1 \ \wedge\ \dots \wedge\ z_{k-1} R z_k \ \wedge\ z_k = y$$

   (Hint: Prove the implication $\Rightarrow$ by rule induction and the implication $\Leftarrow$ by induction on the length k of the R-chain).
2. Give the rules that define instead a relation R' such that

$$xR'y \quad \Leftrightarrow \quad \exists k \geq 0.\ \exists z_0,\dots,z_k.\ x = z_0 \ \wedge\ z_0 R z_1 \ \wedge\ \dots \wedge\ z_{k-1} R z_k \ \wedge\ z_k = y.$$

4.5. Let IMP$^-$ the language obtained from IMP by removing the **while** construct. Exploit the operational semantics to prove that in IMP$^-$, for every command c it holds the termination property

$$\forall \sigma \in \Sigma.\ \exists \sigma' \in \Sigma.\ \langle c, \sigma \rangle \to \sigma'$$

4.6. Let us consider the following rules, where m, n and k are natural numbers.

$$\frac{}{(m,m) \to m} \qquad \frac{(n,m) \to k}{(m,n) \to k}\ m < n \qquad \frac{(m-n,n) \to k}{(m,n) \to k}\ m > n$$

Prove by rule induction that, for any $n, m, k \in \mathbb{N}$,

$$(m,n) \to k \qquad \text{implies} \qquad k = \gcd(m,n).$$

Note: $\gcd(m,n)$ denotes the greatest common divisor of m and n, i.e., if we write $d|j$ to mean that d divides j (in other words, that there exists h such that $j = d \times h$), then $\gcd(n,m)$ is the natural number d such that $d|m \wedge d|n$ and for any d' such that $d'|m \wedge d'|n$ we have $d' \leq d$.

4.7. Prove that, according to the operational semantics of IMP, for any boolean expression b, command c and stores σ, σ'

$$\langle \textbf{while}\ b\ \textbf{do}\ c, \sigma \rangle \to \sigma' \qquad \text{implies} \qquad \langle b, \sigma' \rangle \to \mathit{false}$$

Explain which induction principle you exploit in the proof.

4.8. Exploit the property from Problem 4.7 to prove that for any $b \in \mathit{Bexp}$ and $c \in \mathit{Com}$ we have $c_1 \sim c_2 \sim c_3$ where:

$$\begin{aligned} c_1 &\stackrel{\text{def}}{=} \textbf{while}\ b\ \textbf{do}\ c \\ c_2 &\stackrel{\text{def}}{=} \textbf{while}\ b\ \textbf{do}\ (c\ ; \textbf{while}\ b\ \textbf{do}\ c) \\ c_3 &\stackrel{\text{def}}{=} (\textbf{while}\ b\ \textbf{do}\ c)\ ;\ \textbf{while}\ b\ \textbf{do}\ c \end{aligned}$$

4.9. Define by well-founded recursion the function

$$locs : Aexp \longrightarrow \wp(\mathbf{Loc})$$

that, given an arithmetic expression a, returns the set of locations that occur in a. Use structural induction to show that $\forall a \in Aexp, \forall \sigma, \sigma' \in \Sigma, \forall n, m \in \mathbb{Z}$

$$\langle a, \sigma \rangle \to n \ \wedge \ \langle a, \sigma' \rangle \to m \ \wedge \ \forall x \in locs(a).\ \sigma(x) = \sigma'(x) \qquad \text{implies} \qquad n = m$$

4.10. Consider the IMP program

$$w \stackrel{\text{def}}{=} \textbf{while}\ \neg(x = y)\ \textbf{do}\ (x := x + 1\ ;\ y := y - 1)$$

Define the set of stores $T = \{\sigma \mid ...\}$ for which the program w terminates and:

1. Prove formally that for any store $\sigma \in T$ there exists σ' such that $\langle w, \sigma \rangle \to \sigma'$. (Hint: use well-founded induction on T)
2. Prove (by using the rule for divergence) that $\forall \sigma \notin T.\ \langle w, \sigma \rangle \not\to$.

4.11. Let us consider the IMP command

$$w \stackrel{\text{def}}{=} \textbf{while}\ x \neq 0\ \textbf{do}\ x := x - y.$$

1. Prove that, for any σ, σ', if $\langle w, \sigma \rangle \to \sigma'$ then there exists an integer k such that $\sigma(x) = k \times \sigma(y)$.
2. Give a store σ such that $\sigma(x) = k \times \sigma(y)$ for some integer k but such that $\langle w, \sigma \rangle \not\to$. Explain why the program diverges for the given σ.
3. Define a command c such that, for any σ, σ', $\langle c, \sigma \rangle \to \sigma'$ iff $\sigma(x) = k \times \sigma(y)$ for some integer k. Sketch the proof of the double implication.

4.12. Recall that the *height* or *depth* of a derivation d is recursively defined as follows:

$$depth(\varnothing / y) \stackrel{\text{def}}{=} 1$$
$$depth(\{d_1, ..., d_n\} / y) \stackrel{\text{def}}{=} 1 + \max\{depth(d_1), ..., depth(d_n)\}$$

Given the IMP command

$$w \stackrel{\text{def}}{=} \textbf{while}\ x > 0\ \textbf{do}\ x := x - 1$$

prove by induction on n that for any $\sigma \in \Sigma$ with $\sigma(x) = n \geq 0$ the derivation of

$$\langle w, \sigma \rangle \to \sigma'$$

has depth $n + 3$.

4.13. The *binomial coefficients* $\binom{n}{k}$, with $n, k \in \mathbb{N}$ and $0 \leq k \leq n$, are defined by:

$$\binom{n}{0} \stackrel{\text{def}}{=} 1 \qquad \binom{n}{n} \stackrel{\text{def}}{=} 1 \qquad \binom{n+1}{k+1} \stackrel{\text{def}}{=} \binom{n}{k} + \binom{n}{k+1}.$$

Prove that the above definition is given by well-founded recursion, specifying the well-founded relation and the corresponding function $F(b,h)$.

4.14. Consider the well-known sequence of Fibonacci numbers, defined as:

$$F(0) \stackrel{\text{def}}{=} 1 \qquad F(1) \stackrel{\text{def}}{=} 1 \qquad F(n+2) \stackrel{\text{def}}{=} F(n+1) + F(n).$$

where $n \in \mathbb{N}$. Explain why the above definition is given by well-founded recursion and make explicit the well-founded relation to be used.