Formalized Languages and Theories: Adequacy, Consistency, Power, and Limitations.

Alberto Pettorossi, Department of Computer Science, University of Roma Tor Vergata,
I-00133 Roma (Italy). email: adp@iasi.cnr.it
URL: http://www.iasi.cnr.it/~adp

Abstract.
We illustrate the rôle of formalized languages for the realization of hardware and software systems, and we indicate that the correspondence between a piece of reality and its description in a formalized language is crucial for the realization of high performance systems. We describe a technique based on the notion of bisimulation which can be used for checking that correspondence and showing the adequacy of a theory with respect to a given piece of reality. Bisimulation allows for some aspects of the reality not to be represented in the description. We then consider the problem of providing sound definitions within formalized theories and we show that even non-well-founded definitions are acceptable when interpreted on suitable mathematical domains. We illustrate some known results on the limitations of formalized theories. In particular, we consider the problem of proving consistency of a given theory, that is, the absence of contradictory statements. We indicate that in a formalized theory with enough descriptive power, it is impossible to find a proof of its consistency by making use of the proof techniques available within that same theory.

1. Introduction

The use of formalized languages, like those defined in Logic and Mathematics, is crucial in the development of theories and artifacts in science and technology. Systems with high performance and sophisticated behaviour, such as computer systems and communication systems, can be constructed only as a result of a complex intellectual and engineering process which is based on formalized languages.

This process of constructing systems includes the following steps.

Step 1: The description of physical or intellectual objects, that is, the construction of models in a formalized language. This description should express the properties which may influence the behaviour and the performance of those objects.
Step 2: The validation of models via experiments and measures.
Step 3: The design of systems by making use of models.
Step 4: The construction of systems according to their design.
Step 5: The use of systems and their maintenance.

As a simple example of such a five step process, we may consider the construction of a clock. At Step 1 one studies the oscillations of a pendulum and describes them by using differential equations. At Step 2 the theory of oscillations and its differential model is validated via experimental results. At Step 3 one uses some facts of that theory, for instance, Galileo's law of isochronicity, to design a clock based on the oscillations of a pendulum. At Step 4 one constructs that clock, and finally, at Step 5 that clock is used and maintained for measuring time according to the facts holding in the

theory (for instance, the period of one oscillation corresponds the duration of one second).

One can give much more complex examples than this one concerning the construction of a clock, but we believe that in every case when a significantly high precision or sophisticated performance has to be obtained, one has to go through a similar five step process which inevitably requires in the first step the construction of models in formalized languages. Thus, before building hardware or software systems we have to build "models" of some "physical or intellectual objects" by using symbols taken from a formalized language and denoting suitable properties (or qualities) of the objects of interest. By taking advantage of these properties, we may be able to construct systems with the desired performance.

The models to be constructed at Step 1 are themselves intellectual objects in which the properties of interest, such as the period of oscillation and the length of a pendulum in the case of a clock, are related with each other via suitable mathematical equations or relations. In this case as in many other cases, the formalized language of mathematical equations provides a powerful tool for expressing the properties of the piece of reality which are relevant to the construction of the systems of interest.

Therefore, it is important on the one hand, to develop the language and the theories of Mathematics, and on the other hand, to develop the understanding of the reality via experiments, so to express its laws by using, for instance, equations and formal statements within suitable mathematical theories.

We do not want to illustrate here the techniques for a correct development of the mathematical language and a correct understanding of physical or intellectual realities. We will, instead, present some results obtained in Theoretical Computer Science, which may be of some use in the construction of models of the reality and in their validation.

In Section 2 we present the method of bisimulation which is a technique for understanding the relationship between the descriptions (or models) and the objects which they describe. The method of bisimulation may also be used for relating various descriptions of the same object. In Section 3 we present a contribution to the theory of descriptions by indicating how one can provide sound mathematical meaning to non-well-founded definitions and also to impredicative definitions. In Section 4 we present some achievements which show the power of formalized languages in the area of automatic theorem proving. In Section 5 we indicate some well known limitations of formalized languages based on unsolvability results and Gödel's Incompleteness Theorem.

2. The bisimulation technique for checking the adequacy of a description with respect to a piece of reality.

When describing a piece of reality we make use of formalized theories, that is, theories written in a formal language. As an example let us consider the situation depicted in Fig. 1 below. We take the piece of reality which is the elastic collision between two balls. We may describe that phenomenon by using equations as usually done in classical

Mechanics. Such description, also called *model*, requires an *abstraction* (denoted by the function α in that figure), so that some features of the balls are forgotten (for instance, their colour) and only their masses, dimensions, positions, and velocities are taken into account. These quantities are, indeed, the elements which enter into the description, called δ-before, and that description in classical Mechanics is sufficient to describe the effects of the elastic collision and, in particular, it is sufficient to compute the state of the system in any future instant, that is, the future positions and velocities of the two balls. The dynamics, that is, the evolution in time, of the physical reality is computed by using the dynamic laws of Mechanics so that the description, called
δ-after, can be derived from the description δ-before by solving equations.

The description of the piece of reality we consider is given by a *static part*, that is, the one relative to the two balls, and a *dynamic part*, that is, the law of conservation of momentum which is the invariant preserved while reality evolves. These two parts together, the static and the dynamic part, contain enough information to describe the future behaviour of the system at least with respect to the mechanical properties, that is, positions and velocities.

A different example of abstraction is the one concerning the reality of a physician and his formalized description as a computer program, that is, "an expert system for medical diagnosis". The expert system should agree with the behaviour of the physician in the sense that it should be able to perform the same diagnosis performed by the physician when given the same input data.

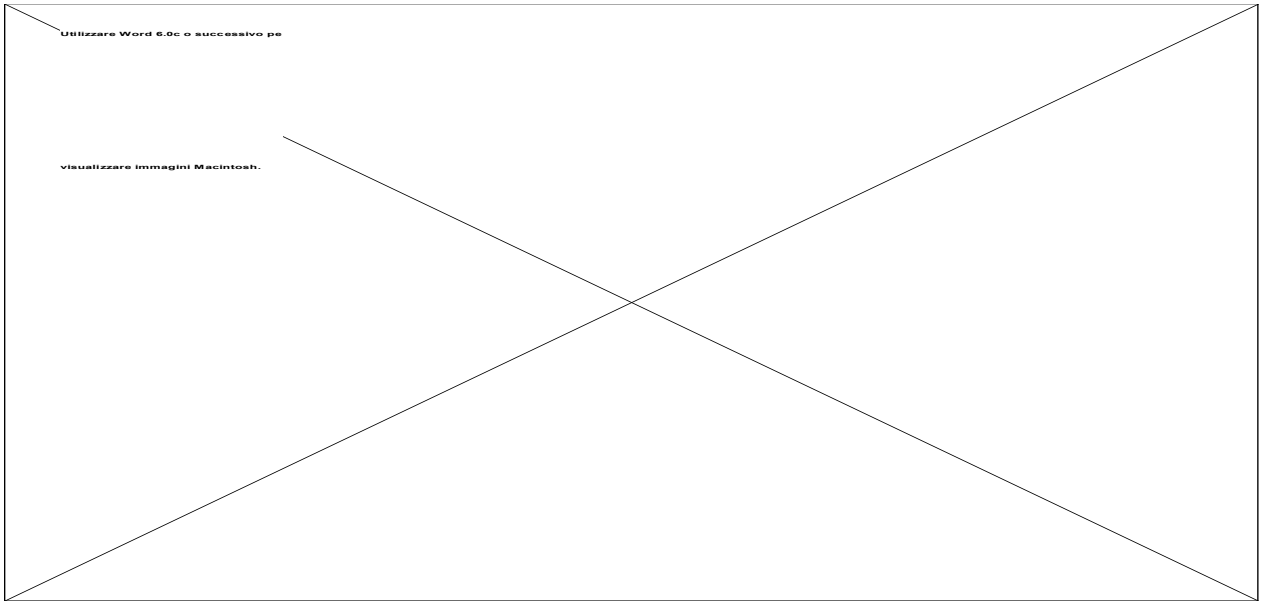

Figure 1. Bisimulation between a description with respect to a piece of reality.

The abstraction function α is made out of two components:

(i) a *theoretical component*, which is a set of sentences in a formalized language L, and defines the codomain of α, and
(ii) an *experimental component*, which describes a piece of reality by generating the corresponding sentence in L, and defines the mapping α as a set of <piece of reality, description> pairs.

The first component is the result of a theoretical, mental activity which abstracts away from the experimental facts and introduces a *working theory* to be used for the description of the reality. The second component is the result of an experimental activity which via suitable measurements, generates values and allows us to get the sentence of L which describes the piece of reality under observation.

These two components are not independent and we may change an old working theory and adopt a new theory, when this change is suggested by the existence of pieces of reality which cannot be described in an old working theory.

Together with the abstraction function α we may also consider a *concretization* function γ which goes in the opposite direction, and maps a sentence into a piece of reality. If the pair $<\alpha,\gamma>$ of functions determines a Galois insertion, then *abstract interpretations* may be used for analyzing the relationship between the pieces of reality and their descriptions [4]. We will not enter into this topic here.

We say that an abstraction α which generates a description of a piece of reality, is *adequate* if and only if there exists a relationship, called bisimulation, between that reality and its description in the sense which we now define.

Definition 1. We say the there exists a *bisimulation* between a reality and its description in a fomalized language L (and thus, abstraction α is adequate) if and only if

(i) whatever change occurs in the given reality from the state "reality-before" to any state "reality-after", then it is possible to derive from the description δ-before a suitable description δ-after, so that the observations on the state "reality-after" agree with the properties described by δ-after, and
(ii) whatever description δ-after can be computed from the description δ-before via the deductive rules of the formalized language used, then there exists a state "reality-after" in the evolution of the reality so that the observations on that state agree with the properties described by δ-after.
The deductive rules of the formalized language L are called the "Dynamics of the Theory".

Thus, the adequacy of the abstraction α, that is, the validation of the description of the reality, is checked by: (i) considering a set of possible experiments which either stimulate changes of the reality or observe its autonomous changes, and (ii) verifying via measurements that the results of the experiments are in accordance with the description δ-after.

The definition of bisimulation we have given above, derives directly from the one introduced in [8,10] for relating nondeterministic behaviours of machines and establishing equivalence among them. The use we make here of the notion of

bisimulation is informal, because only one of the two entities, that is, the description of the reality, is assumed to be expressed using sentences of a formalized language, while the other entity, that is, the reality itself, is not assumed to be a formal system. The reality can only be observed by means of experiments or observations and by such experiments it produces values in a formalized language. Those values allow us to check whether or not the description is adequate.

The use of bisimulation is also related to the idea of "testing equivalence" among systems by performing experiments, as described in [3]. We cannot enter into the details here.

In the above Definition 1 it may seem too restrictive to require that the reality should evolve according to the deductive rules of the formalized language L, and indeed, one may release that condition by allowing that some descriptions which can be deduced, may not correspond to states reached by the evolution of the reality.

Notice also that the notion of bisimulation may allow for some changes occurring in the reality not to be representable by suitable descriptions, because the observations may not be capable of determining the values of the quantities which are changed. In particular, the bisimulation allows for some "internal interactions" among parts of the reality not to be observed by experiments and represented by descriptions. Thus, we can incorporate within this framework "approximated descriptions" of the reality. This is an important feature because the formalized language we use cannot always be expressive enough to completely describe the systems we consider. Approximations are imposed on the descriptions of the physical world if we assume that Heisenberg's uncertainty principle holds.

It should be noticed that the reality we have considered in Fig.1, can itself be an intellectual entity. Thus, the bisimulation concept can also be used between different descriptions of the same piece of reality. Indeed, it may be necessary to construct different models of the same piece of reality because we have to abstract away from some details of that reality in different ways. We encounter this situation very often in various scientific fields, such as Mathematics, Physics, Logic, and Computer Science. For instance, in Logic we can view the propositional calculus as an abstraction of first order predicate calculus, and in Computer Science we can view the class of minimal finite-state automata as an abstraction of the class of finite-state automata.

When the reality is an intellectual entity, the experiments by which we derive values and check the adequacy of the models, have to be performed in a mental way by applying the rules, that is, the dynamics, of the corresponding formal theories.

We present now a simple example of bisimulation between intellectual entities. It is the bisimulation between ordinary arithmetic and arithmetic modulo 9. With reference to Fig. 1, let us take the "reality" to be the natural numbers with the addition operation and let us consider the abstraction performed in the "9 proof" technique, that is, the abstraction which abstracts a number into the sum modulo 9 of its figures.

For instance, let us assume that the reality-before is 33+47, the reality-after (that is, after performing the addition) is 80, δ-before is 6+2 (because 3+3=6, 4+7=11, and

1+1=2), and δ-after is 8 (because 8+0=8). The dynamics of the reality in this case is determined by the rules of addition, and the dynamics of the description is determined by the rules of addition modulo 9.

Let us now consider the case in which two distinct descriptions of the same physical or intellectual reality are taken into consideration (see Fig. 2). The two descriptions refer to two different abstractions, say $\alpha 1$ and $\alpha 2$. In this case if we require that a bisimulation exists between each description and the reality, then the two descriptions should be, in some sense, in accordance with each other.

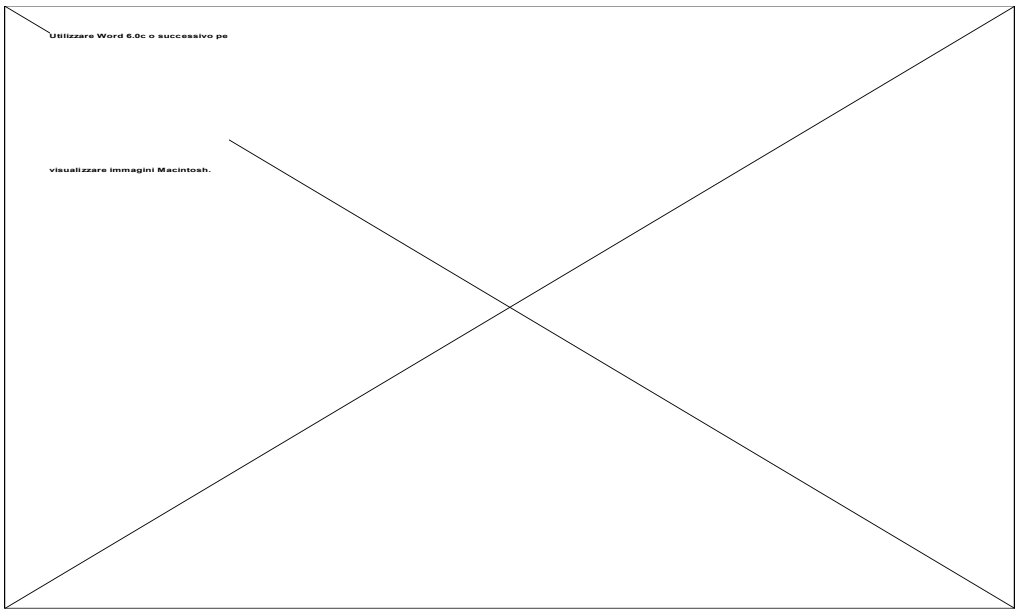

Figure 2. The case of two distinct descriptions of the same reality.

The relation τ we have used in Fig. 2, is the mathematical way of expressing that accordance. Thus, in particular, if the description $\delta 2$ is more detailed than the description $\delta 1$ then it should be the case that both $\delta 2$-before implies $\delta 1$-before and $\delta 2$-after implies $\delta 1$-after.

The study of the abstract, algebraic properties of the bisimulation relation leads us to the discovery of the rules for a correct use of the descriptions. For instance, it is easy to see that some diagrams should commute. In particular, if we go via τ from $\delta 2$-before to $\delta 1$-before and then we apply the dynamics of Theory 1 we should obtain the same result we can get if we apply the dynamics of Theory 2 to $\delta 2$-before and then we go via τ from $\delta 2$-after to $\delta 1$-after.

This commutativity property should be preserved when different descriptions refer to the same reality. This commutativity also forces suitable correspondences between different theories, such as Physics and Chemistry, when describing the physical world, and it can also be viewed as a way of controlling different levels of formal reasoning and deductions across different sciences.

3. The power of formalization to give meaning to definitions.

In this section we illustrate an important issue related to the use of formalized languages when describing pieces of reality. In particular we present a few approaches concerning various techniques for giving meanings to definitions. Only definitions which convey a unique meaning, ensure a firm basis for developing theories which are free of contradictions, and only such theories may be used for describing pieces of reality.

We will give a few examples of non-trivial definitions in the formalized language of Mathematics and we will provide their formal meaning even if at a first sight, it may seem difficult to do so. We will show that in these cases, closed mathematical structures, that is, domains which are closed with respect to suitable operations, are sufficiently powerful tools for providing the required meaning.

Example 1. We start from a familiar example we all encountered at school. In this example the definition of entities, also called *unknowns*, are given in terms of each other by means of a *system of equations*. Take, for instance, the system of the following two equations:

$$x = -y+2, \qquad y = x+1 \tag{1}$$

which indeed define the two rational values: $x = 1/2$ and $y = 3/2$.
In this case the unique meanings of the unknowns x and y is provided by rational numbers because the domain of the rational numbers is closed with respect to the operations of addition, subtraction, multiplication, and division.

Example 2. This example refers to the definition of mathematical functions via *recursive equations*. Take, for instance, the following recursive equation which defines the factorial function:

$$\text{factorial}(x) = \textbf{if } \text{iszero}(x) \textbf{ then } 1 \textbf{ else } x*\text{factorial}(x-1) \tag{2}$$

We have that factorial(0) = 1 and factorial(3) = 3∗(2∗(1∗(factorial(0)))) = 3∗(2∗(1∗1)) = 6.

A computation process based on matchings and replacements, provides a unique meaning to Definition (2) for any non-negative integer value x, because there exists a well-founded ordering on the set of natural numbers, that is, in the set of natural numbers it does not exist an infinite descending sequence of the form: ... < x−2 < x−1 < x, for any natural number x.

Indeed, for any given value x > 0, the computation of factorial(x) requires the computation of factorial(x−1) which in turn may require the computation of factorial(x−2), and so on. This process is bound to terminate because the set of natural numbers is well-founded and eventually we are require to compute the value of factorial(0) which is 1.

Example 3. This is more complex example and it is related to the so called *domain equations* [11]. Let us consider, for instance, the following domain equation:

$$X = A + B \times X + X \to X \qquad (3)$$

which is supposed to define a domain X of objects, given the two domains A and B.

In Equation (3) the symbols plus (+), times (×), and arrow (→) denote the union, the cartesian product, and continuous function space operators, respectively. The unique meaning of the domain X is obtained in this case by considering the theory of the so called Scott's domains [11]. This theory allows us to solve isomorphisms of the form: $X = X \to X$, and also the isomorphisms such as Equation (3). The difficulty in this example derives from the fact that in ordinary Set Theory there is no isomorphism between a set X and the set of all functions from X to X, when the cardinality of X is larger than 1.

Having Scott's domains at our disposal, it is also possible to provide meaning to recursive definitions with *self-application*. Let us consider, for instance, the following equation:

$$\text{factorial}(x) = Y\ (\lambda g.\ \lambda x.\ \textbf{if}\ \text{iszero}(x)\ \textbf{then}\ 1\ \textbf{else}\ x * g(x-1))\ x \qquad (4)$$

$$\text{where}\quad Y = \lambda f.\ (\ (\lambda x.f(xx))\ (\lambda x.f(xx))\ )$$

For the non-expert reader we recall that in Equation (4) we have used the lambda notation to denote mathematical functions. In this lambda notation the function which given an input value x produces the expression e as output, is written as λx.e. For instance, the addition function will be written as λx.λy.x+y.

In Equation (4) there is a form of self-application: x has x itself as argument. The solution of the domain equation $X = X \to X$ using Scott's domains allows us to overcome also this self-application problem because we can view an element of X as a function from X to X by applying the suitable functions which define the isomorphism $X = X \to X$.

The self-application mechanism is not so strange as it may seem, and expressions such as Y above, are indeed used in the implementation of modern computer languages.

Example 4. In this example we consider other forms of definitions, the so called *impredicative definitions*, and we need to use suitable categorical structures to provide meaning to them.

A definition is said to be impredicative if it refers to the collection of objects which is defining. For instance, a set S defined as follows:

$$S = \{\ x \mid \forall y \in A.\ P(x,y)\}$$

is said to be defined in an impredicate way if S is an element of the set A.
This is an impredicative definition of the Peano natural numbers:

$$N = \{\ n \mid \forall X.\ ((0 \in X \text{ and } \forall x.\ (x \in X \to x+1 \in X)) \to n \in X)\}.$$

Another example of an impredicative definition can be given in the type system F of J.-Y. Girard. In the system F there is the collection of all types, called Types, and one allows for the following type:

$\forall X \in \text{Types}.\ A$

for any given type A. Thus, we get that: $(\forall X \in \text{Types}.\ A) \in \text{Types}$. Despite this circularity, one may show that the system F is coherent, that is, no contradiction can be derived in it. Categorical semantics for type systems of this form have been provided in [7].

Before closing this section we would like to mention other kinds of definitions which are given as formulas of the first order predicate calculus. We can provide a definite meaning to them by using the so called "three-valued interpretations" [1].

Example 5. To fix our ideas let us consider the following example (which, for simplicity, refers to the propositional calculus, rather than the first order predicate calculus):

$b \leftarrow \text{true}$
(5)
$p \leftarrow \neg\, q, b$
(6)

In these formulas, called *clauses*, the symbol $\leftarrow$ stands for "reverse implication", and $\neg$ and comma stand for the logical connective "not" and "and", respectively. We can view formulas (5) and (6) as the definition of the truth value of the propositions b and p, respectively. Formula (5) says that the truth value of b is "true" unconditionally, while formula (6) says that the truth value of p is "true" if the truth value of q is "false" and the truth value of b is "true". Here we face the problem of deciding whether or not the truth value of q is "false" because we do not have any formula from which we can derive it.

The approach based on three-valued interpretations solves this problem as follows. For simplicity we will describe this approach in the case of the propositional calculus and assuming that: (i) precisely one propositional letter occurs to the left of "$\leftarrow$", and (ii) negated propositions occur only to the right of "$\leftarrow$".

Let P denote the conjunction of some given clauses. Let P^- denote the same conjunction where the negated propositions are deleted, that is, the truth value of negated propositions is assumed to be "true". Let P^+ denote the same conjunction where the clauses with negated propositions are deleted, that is, the truth value of negated propositions is assumed to be "false".

Thus, in our case

P^- is: $b \leftarrow \text{true}$
$p \leftarrow b$ (P^- is derived from P by assuming that $\neg$ q has truth value "true") and

P^+ is: $b \leftarrow \text{true}$ (P^+ is derived from P by assuming that $\neg$ q has truth value "false").

We then consider the least Herbrand models of P^- and P^+, denoted by $M(P^-)$ and $M(P^+)$, respectively. We recall that the least Herbrand model of a conjunction of clauses without negated propositions, is the minimal set of propositions whose truth value can be shown to be "true" by using modus ponens (i.e., by using the rule by which for any proposition a and b, from b and a $\leftarrow$ b we can deduce a).
In our case we have that: $M(P^-) = \{b,p\}$ and $M(P^+) = \{b\}$. We then consider the complement of $M(P^-)$, denoted by $M^-(P^-)$, with respect to the set of all propositions occurring in P. In our case we have that:

$M^-(P^-) = \{b,p,q\} - \{b,p\} = \{q\}$.

Now we compute new versions of the conjunctions P^- and P^+, starting from the initial conjunction P. These new versions are computed by suitable deletions of propositions or clauses by assuming that:
(i) the truth value of a negated proposition, say $\neg$ q, is "true" if q occurs in $M^-(P^-)$ and it is "false" if q occurs in $M(P^+)$, and
(ii) the truth value of the remaining negated propositions it assumed to be "true" for the computation of P^- and it assumed to be "false" for the computation of P^+.

We then continue by considering the new values of $M^-(P^-)$ and $M(P^+)$ until the values of $M^-(P^-)$ and $M(P^+)$ do not change by considering new versions of the conjunctions P^- and P^+, that is, until a fixpoint is reached. The value of $M(P^+)$ at the fixpoint is assumed to be the meaning of the definition P.

In our case the meaning of the definition P is that both b and p have truth values "true".

4. The power of formalization to achieve high quality reasoning.
In this section we present two important properties of formalized languages. They illustrate the power of formalization when denoting computations, and they will also show what automatic tools and computer programs can do for improving the quality of human reasoning.

(i) The first property is that formalization improves precision and speed in mechanical computations.
This property is essential in real time systems where the results of some computations have to be computed within given time bounds: a result which arrives too late may no longer be useful or may cause disasters. This may be the case when we need to solve the differential equation which describes the motion of the Moon and we need to compute for how long an engine should work for a correct landing of a spacecraft on the Moon. If the solution arrives too late, it may be useless or, indeed, a disaster may happen. In this case, having the formalized language of Mathematics, we may apply the techniques which are available within Mathematics to compute the solutions of differential equations. We may also estimate how long it takes to compute the solution and thus, we may also evaluate whether or not we can meet some given safety requirements.

(ii) The second property is that formalization allows machines to enhance the deductive and inductive capabilities of human beings.

Indeed, through suitable formal theories, one can build expert systems which can improve the intellectual performance of people. It is a reality of the present technology that one can construct computer programs which can safely take the place of aircraft pilots, perform medical diagnoses, suggest chains of chemical reactions to synthesize molecules with given properties, assist in geological prospectings, play chess games with very high performance, etc.

Here, in particular, we would like also to recall that computer programs based on formalized theories such as modal logics and temporal logics, can verify other software products, safety critical systems, and communication protocols by applying theorem proving techniques. Experience shows that software products already in use, have actually been proved erroneous by using computer programs which were able to discover errors beyond the ability of trained persons in the field.

Recently, software systems which help humans in deriving programs from given initial specifications have been developed [12]. These systems are very powerful and often new and very efficient programs, much more efficient than those known before, have been derived.

Recent work also shows that inductive capabilities of humans can be enhanced by using computer programs. Among many other results, we would like to mention the one reported in [14] where it is shown that via Inductive Logic Programming a new indicator of molecular mutagenicity, that is, the capability of molecules of causing mutations in the DNA sequence (and often producing cancer), was found within a subset of previously published data.

Together with these practical achievements, we would like to mention also some theoretical achievements which show how powerful formalized theories and mechanical theorem provers based on those theories can be. Important and complex theorems of Mathematics and Mathematical Logic such as Cantor's Theorem (stating that a set has smaller cardinality than its powerset), the Church-Rosser property of λ-calculus, and Gödel's Incompleteness Theorem have recently been proved using mechanical theorem provers [13].

Let us make two final remarks before closing this section.
The first remark is that all the achievements we have mentioned above in the area of software production, do not provide a satisfactory answer to the question about the meaning of "intelligence' and in what sense intelligence is beyond "high quality computing" or "high quality deductive or inductive skill".

The second remark is about the ability of constructing theories, called *meta-theories*, describing other theories, called *object-theories*, and the ability of formalizing in suitable meta-theories the deductive or inductive apparatus of given object-theories.

These meta-theories allow one to produce very powerful and sophisticated tools for enhancing people's intellectual abilities. However, the "self-consciousness" experience of people who while thinking, are able to think about their own thoughts, appears to be much beyond the ability of using meta-theories to reason about the deduction or induction process which occurs within object-theories. We will not discuss these issues here.

5. The limits of formalization regardless the growth of technology.

In the previous section we have illustrated the power of formalization and we have indicated through some examples how formalization can be used for enhancing human intellectual capabilities. There are, however, some inherent limitations in the use of formalization. Indeed, when we use a formalized language for expressing concepts and stating problems, then it may be the case that those problems are *unsolvable*, that is, no computing device exists by which we can provide a solution to those problems.

Here is a simple example of an unsolvable problem. It is the problem of the equivalence of context-free languages. It can be stated as follows.

Let us consider a set L of upper-case and lower-case letters and let us consider a set R of rewriting rules of the form:

$A \rightarrow B\ C$ (an upper-case letter produces the sequence of two upper-case letters) or

$A \rightarrow a$ (an upper-case letter produces a lower-case letter)

where in each rewriting rule the upper-case letters and the lower-case letters are taken (possibly with repetition) from the set L.

Using a set R of rewriting rules and starting from an upper case letter, say A, we get a set S(R) of strings of lower-case letters. For instance, given the set of rules $R = \{B \rightarrow A\ B, A \rightarrow a, B \rightarrow b\}$, starting from B we derive all strings which begin by zero or more a's and terminate by b.

It can be shown that no computing device M exists such that for any given sets R1 and R2 of rules, M terminates after a finite number of computation steps and M returns "yes" if and only if S(R1) = S(R2), that is, M can tell us whether or not S(R1) = S(R2).

In more formal terms, no total recursive function M exists such that for any given sets R1 and R2 of rules M returns "yes" if and only if S(R1) = S(R2).

One more unsolvable problem is the so called Halting problem. The unsolvability of the Halting problem tells us that no computing device M exists such that for any given computing device N and input value i, the device M tells us after a finite number of computation steps whether or not N terminates on input i.

One should also notice that the limitations we have indicated through the existence of unsolvable problems are intrinsic to the formalization and they do not depend on the achievements of the technology, that is, on how fast or how powerful future computing devices can be.

We now illustrate the limitations of formalized languages through a negative result which holds within Peano Arithmetic, that is, the theory of natural numbers with the operations of addition and multiplication and the induction rule which says that for any formula P(x) we can infer that $\forall x.P(x)$ holds if P(0) holds and ($\forall x.$ P(x) implies P(x+1)) holds.

This negative result is the Gödel's Incompleteness Theorem (1931), which says that there are true sentences in Peano Arithmetic which are not provable in Peano Arithmetic itself. In particular, the consistency of Peano Arithmetic (i.e., the fact that in Peano Arithmetic there is no sentence F so that F and ¬ F can be proved) cannot be proved within Peano Arithmetic.

More recently Friedman (1981) has found a statement called the Friedman's Finite Form (FFF) Theorem, which is true in Peano Arithmetic and it cannot be proved within Peano Arithmetic. The FFF Theorem is a statement much simpler than the one which expresses the consistency of Peano Arithmetic. Due to its simplicity, it gives us a deeper understanding of the limitations of the formalized language of Peano Arithmetic and the limitations due to Gödel's Incompleteness Theorem. In order to express the FFF Theorem we need the following preliminary definitions.

A partial order over a set D, which we denote by $(D, \leq)$, is a reflexive, symmetric, and transitive binary relation over the set A. We denote by glb a binary operation, called greatest lower bound, over $D \times D$ such that: (i) $glb(x,y) \leq x$, (ii) $glb(x,y) \leq y$, and (iii) if $z \leq x$ and $z \leq y$ then $z \leq glb(x,y)$.

A total order over a set D, which we also denote by $(D, \leq)$, is a partial order over D such that for any x and y in D either $x \leq y$ or $y \leq x$ holds. In this case we say that D is a totally ordered set.

A sequence on a set S is a function from the set of natural numbers to S.

A finite tree T is a partial order $(T, \leq)$ such that T has a smallest element with respect to $\leq$ (often this smallest element is called root) and if $a \in T$ then $\{x \mid x \leq a\}$ is a totally ordered set.

An embedding between two partial orders $(T1, \leq_1)$ and $(T2, \leq_2)$ is a function f from T1 to T2 which preserves the greatest lower bounds, i.e., $f(glb_1(x,y)) = glb_2(f(x),f(y))$, where for i = 1, 2, by glb_i we denote the greatest lower bound in Ti. (Thus, an embedding is a monotonic function.)

We write $T1 \leq_T T2$ to mean that there is an embedding between the trees T1 and T2.

This is the statement of the FFF Theorem:

> For any n there exists m such that for any finite sequence $\langle T1, T2, ..., Tm \rangle$ of finite trees such that Ti has at most $n \times (i+1)$ elements, there exist j and k such that $j<k\leq m$ and $Tj \leq_T Tk$.

For the expert we recall that the FFF Theorem is a finite form of Kruskal's Theorem and it is a
Π^0_2 statement.

6. Concluding remarks.

We have seen in what sense a formalized language is strong and weak at the same time. It is strong because: (1) it gives meaning to definitions which are recursive or non-well-founded or impredicative, and (2) it provides tools for the mechanical derivations of

theorems, and thus, it enhances the ability of the human mind when performing deductions or inductions in formalized theories.

A formalized language is also weak because if it has high expressivity, then it impossible to formally show consistency of some theories one may construct using that language, and thus, one runs the risk of constructing in that language trivial theories where all sentences are true.

How can we overcome this problem? How can we look for truth in a highly expressive language and at the same time, keeping our reasoning within a formalized framework?

I propose two paths to follow. The first path is in the "intellectual optimism", by which I mean that our reason abilities, if well trained and used with discipline, lead us towards correct statements. Moreover, even if we cannot show that a given theory is consistent, nevertheless, we can use for our purposes a portion of that theory which is free from contradictions.

This intellectual optimism is acceptable in a very practical sense: firstly, because in the history of science many theories have been used and were useful without having the proof of their consistency, and secondly, because some achievements of the present technology such as computers, telecommunication systems, aeroplanes, and medical instruments, "go beyond the limitations" of showing the consistency of the theory which supports their design. This is also true on the negative side, in the sense that the destructive power of most weapons does not really depend on the proof of the consistency of the scientific theories on which their design is based.

The second path I would like to suggest for a sound search for truth, is "moderate formalization", which can also be considered as a sort of intellectual optimism. By moderate formalization I mean that the use of formalization should be advised "up to a certain degree", leaving some basic notions or notations as intuitively clear or unambiguous, and ultimately relying on the consistency of some mathematical theories, such as Set Theory or Peano Arithmetics. Those theories may be assumed as a sufficiently firm ground for the development of formalized reasoning in Mathematics and in other formal sciences.

In communicating to other people our ideas about some "pieces of reality" using formalized theories, we should not concentrate too much on the formalism we use, and on the contrary, we should somewhat rely on the ability of our listener to understand our ideas beyond the limitations of the language we use for expressing the "piece of truth" our statements convey. The two-way dialogue with our listener will increase the amount of "truth" we will be able to share with him as time progresses, by decreasing the inevitable ambiguity of the statements we use and eliminating imprecision and useless redundancy.

The objective of "moderate formalization" is that our listener may receive and understand our "piece of truth" with sufficient precision, so that he can use it correctly, according to the goals he sets for his own actions. Here is where orthopraxis, that is, correct actions, and orthodoxy, that is, correct ideas, have to interact in the unity of the human activity and human communication.

Finally, I would like to remark that some of the ideas for the correct use and development of formalized languages in science are strongly related to some of the criteria which Cardinal J. H. Newman proposed for a correct development of dogma in: "An Essay on the Development of Christian Doctrine" (1845,1878) [9], such as: (i) the logical coherence, (ii) the power of assimilation, (iii) the conservation of the past, and (iv) the anticipation of the future. I will not discuss this relationship here. The reader may find an illustration of those criteria in the contribution of Prof. Giuseppe Tanzella-Nitti in these Proceedings.

The relationship between the bisimulation approach we have presented here and the theories of explanation and confirmation in science [5] will be studied in the future.

7. Acknowledgements

Section 3 is based on the paper [6] by Prof. Giuseppe Longo from whom I have learnt that formalized languages may be very powerful in providing meaning to definitions, even when intuition does not help. I also would like to thank my colleague Dr. Maurizio Proietti at IASI-CNR, Roma (Italy), and the people who attended the Seminar "Interpretations of Reality: a Dialogue among Theology, Philosophy, and Sciences" at the Pontifical Lateran University, Roma (Italy) on April 16-17, 1999. I greatly profited from their conversations and comments.

8. References.

[1] K. R. Apt, R. N. Bol: "Logic Programming and Negation: A Survey", Journal of Logic Programming, 19, 20, 1994, pp. 9-71.

[2] J. Barwise, L. Moss: "Vicious Circles: on the mathematics of non-well-founded phenomena", CSLI Lecture Notes 060, Stanford University, USA (1996).

[3] R. De Nicola, M. C. Hennessy: "Testing Equivalence for Processes", Theoretical Computer Science 34, pp. 83-133.

[4] N. D. Jones, F. Nielson: "Abstract Interpretation: a Semantics-Based Tool for Program Analysis", Handbook of Logic in Computer Science. Vol. 4: Semantic Modelling. (S. Abramsky, D. M. Gabbay, and T. S. E. Maibaum, eds.), Oxford Science Publications, Clarendon Press, Oxford, 1995, 527-636.

[5] K. Lambert, G. G. Brittan, Jr. "An Introduction to the Philosophy of Science", Fourth Edition, Ridgeview Publishing Company, Atascadero, California, 1992.

[6] G. Longo: "Cercles Vicieux, Mathématiques et Formalisations Logiques" Invited talk, Coll. Logique et Sciences Humaines (also in: http://www.dmi.ens.fr/users/longo/) (1999).

[7] G. Longo, E. Moggi: "Constructive Natural Deduction and its Omega-Set Interpretation", Mathematical Structures in Computer Science, Vol. 1, n.2, 1991.

[8] R. Milner: "Communication and Concurrency", Prentice Hall, (1989).

[9] J. H. Newman: "The Development of Christian Doctrine", Longmans, London, 1914.

[10] D. M. R. Park: "Concurrency and Automata on Infinite Sequences" Lecture Notes in Computer Science n. 104, Springer Verlag, 1980.

[11] D. Scott: “Continuous Lattices”, Toposes, Algebraic Geometry and Logic, (Lavwere, ed.) Lecture Notes in Mathematics 274, Springer Verlag, 1972, pp. 97-136.

[12] D. R. Smith: “Mechanizing the Development of Software” In: “Calculational System Design”, Proceedings of the International Summer School Marktoberdorf, (M. Broy, ed.), NATO ASI Series, IOS Press, Amsterdam, 1999. Also: Kestrel Institute Technical Report KES.U.99.1, March 1999.

[13] Natarajan Shankar. ftp://ftp.cs.utexas.edu/pub/boyer/nqthm/nqthm-1992/examples/shankar.

[14] A. Srinivasan, S. H. Muggleton, R. D. King, M. J. E. Sternberg: “Theories for mutagenicity: A study of first order and feature based induction”, Artificial Intelligence, 85: 277-199, 1996.