# Cartesian versus Newtonian Paradigms for Recursive Program Synthesis

Marta Franova

# Cartesian versus Newtonian Paradigms for Recursive Program Synthesis

Marta Franova
LRI, UMR8623 du CNRS & INRIA Saclay
Orsay, France
mf@lri.fr

***Abstract* — In this paper, we bring a new solution to two unusual questions in Computer Science relative to recursive Program Synthesis (PS). To clarify our ideas we introduce the concepts of Newtonian and Cartesian paradigms to scientific creativity when related to PS. The main contribution of the paper is a thorough discussion on the difference between disruptive Cartesian creation and classical Newtonian construction of a theorem prover devoted to PS. We illustrate these ideas by an analysis of Peano's axioms defining the set of non negative integers, from the point of view of creativity and we explain why Newtonian systemic creativity is not suited for conceiving this simple recursive system. This analysis is then applied to a more complex case of the general framework for our own 'Constructive Matching Methodology' (*CMM*) as a Cartesian paradigm to the creation of an autonomous theorem prover for PS. This methodology illustrates that Cartesian Intuitionism can be viewed as a 'generator of new ideas'.**

***Keywords - evolving systems; Cartesian Intuitionism; Newtonian construction; Cartesian creation;* CMM.**

## I. INTRODUCTION

Autonomous Program Synthesis is a desirable goal even though, in case of synthesis of recursive programs, it is recognized as a theoretically inaccessible one. After thirty years of experiments and deep systemic and epistemological studies to build solid justifications for new pragmatic foundations, we were able, in [1] and [2], to launch a clearly defined new approach. This paper goes deeper into the fundamentals of our approach. These fundamentals are useful for all who are concerned by systemic scientific creativity in their work.

There are two main ways to tackle with recursive Program Synthesis, namely inductive and deductive. Automatic construction of programs speeds up the conception process and, in the case of deductive way, it guarantees the correctness of synthesized programs. Therefore, in this paper we are interested in the deductive approach to Program Synthesis (PS) introduced by Manna and Waldinger in the eighties [57] and followed by many authors, for instance [10], [64], [32], [11], [25], [59], [61] [18], [30], [55]. This problem is however undecidable as a consequence of Gödel's Theorems [51]. In this paper, we shall present an attempt to, as much as possible, approximate the automation of the deductive approach to PS by introducing the conceptual switch of 'Cartesian Intuitionism', described in the book [41] in an informal way and presented shortly in [2] and [1]. This paradigm is, from an epistemological point of view, an interesting and even necessary complement to the more formal Newtonian paradigms. From a practical point of view, by introducing concepts that are disruptive in Newtonian paradigm, Cartesian Intuitionism improves the rigor of communication and increases the creative potential of researchers in various domains not only in those related to PS.

Since dealing with existentially quantified variables in inductive proof is recognized by scientific community as a difficult problem (see [13], [17]), it is still too soon to compare the application of Cartesian and the Newtonian paradigms in PS on performance basis. However, our presentation in this paper will show how a somewhat disruptive but pragmatically and epistemologically justified conceptual switch (or 'epistemological rupture', as Gaston Bachelard says in [4]), may change the perspective of the focus in conceiving a PS system and thus enlarge and improve not only a frame of thought of the creators of a PS system but also of a user of a theorem prover in the process of recovery from a failure.

The paper is structured as follows.

In Section II, we recall the formulation of the deductive paradigm to PS and we present two basic problems and two unusual questions related to PS. We present a new and disruptive way of perceiving the limitations determined by Gödel [51]. This disruptive way is justified in the epistemological (rather than mathematical) Cartesian Intuitionism we present in this paper. In Section III, we present the main features of Newtonian and Cartesian paradigms to scientific creativity related to PS. In Section IV, we use the example of Peano's axioms in order to underline the deep gap between Cartesian creating a set of axioms, and Newtonian making use of a given set of axioms. This detailed example enables us to precise what is the difference between Newtonian synergetic *construction* and Cartesian symbiotic *creation* of a system. In Section V, we recall the basic notions of Cartesian Intuitionism illustrated in Sections III and IV. We shall devote Section VI to the description of our Constructive Matching Methodology (*CMM*) in the light of Cartesian Intuitionism. In particular, we describe a technique called *CM*-formula construction that is a strategic basis not only in conceiving inductive proofs typical for the deductive paradigm but also in conceiving our whole PS system. In Section VII, we present the main drawbacks and the main advantages of our approach in comparison with Newtonian approaches.

## II. PROGRAM SYNTHESIS

### A. Definition of the Deductive Approach to Program Synthesis

By Program Synthesis (PS) we call here the deductive approach to automatic construction of recursive programs introduced in [57]. This paradigm starts with a specification formula of the form

$$\forall x\, \exists z\, \{P(x) \Rightarrow R(x,z)\},$$

where x is a vector of input variables, z is a vector of output variables, P(x) is the input condition. R(x,z) is a quantifiers-free formula and expresses the input-output relation, i.e., what the synthesized program should do. For instance, let us suppose that 'member' is a predicate deciding whether a natural number is an element of a given list and 'ltl' is a predicate that decides whether a given natural number is less than or equal to all elements of a given list. Then,

$$\forall x \in \text{LIST}\ \exists z \in N\ \{x \neq \text{nil} \Rightarrow \text{member}(z,x)\ \&\ \text{ltl}(z,x)\},$$

is a specification formula for a minimum of a list of natural numbers.

A proof by recursion of a specification formula, when successful, provides a program for the Skolem function sf that represents this program, i.e., R(x,sf(x)) holds for all x such that P(x) is verified. In other words, PS transforms the problem of program construction into a particular theorem proving problem.

The role of the deductive approach is thus to build an inductive theorem prover specialized for specification formulas (ITPPS).

### B. Problems

There are two main problems with respect to the goal to build an inductive theorem prover specialized for specification formulas:

(1) treatment of strategic aspects of inductive theorem proving system specialized for specification formulae,
(2) treatment of strategic aspects of creativity related to the design of such theorem prover.

As to (1), there is the above mentioned limitation determined by Gödel [51]. Because of the practical importance of PS, to build an ITPPS, standard approaches to PS use this worst-case limitation as an argument for adapting already existing mechanisms that may too be undecidable such as general term rewriting systems (see [31]), rippling (see [15]) or SMT (see [24]).

To our best knowledge the problem (2) was not yet treated in Computer Science. We think that it is so simply because, as we have just mentioned, researchers prefer adapt already existing tools to PS instead of asking two questions:

a) Can the logical limits of Gödel's results be 'overcome' by a pragmatic reformulation of PS problem?
b) Can there be a custom-designed theorem prover for PS?

We have asked these questions in eighties and our work is directed by these questions since. This is why this paper is concerned mainly with (2), which puts then (1) in another perspective. In the following sub-section we present our argument in favour of positive answer for a) and then we shall proceed to an extensive answer for b).

### C. A disruptive idea to 'overcome' limitations of Gödel's results

The goal of this section is to present a new pragmatic interpretation of Gödel's results. It is in no way intended as challenging Gödel's results. In other words, Gödel results hold also in this new paradigm. However, they have a stimulation effect instead of paralysis one. Understanding this new pragmatic interpretation is necessary for understanding the remaining parts of this paper.

First, let us recall what are the limitations specified by Gödel's results [51].

The first limitation is the total incompleteness result concerning natural numbers N. This practically means that there is a true statement F such that both F and not(F) can neither be proved nor disproved in N. Moreover, if F is added to the axioms defining N then there can still be found a new formula that is undecidable in this new system. And this holds *ad infinitum*.

The second limitation is the affirmation that there is no finite decision procedure for proving or disproving all formulae. This practically means that there is no deductive algorithm that could decide in a finite time whether an arbitrary formula G is true or false.

Let us consider the first limitation. What does incompleteness means practically? We have a very simple illustration for this problem in fifth Euclid's postulate (postulate for parallels) for geometry. For a long time mathematicians could not decide whether this postulate really is necessary for defining the usual geometry, i.e., whether the first four Euclid's postulates form a complete axiomatic system. It is only in 19th century that Lobachevski and Bolyai showed that when only first four postulates are considered, one can add to them one of negations of the fifth postulate and obtain thus new geometries completely different from that specified by Euclid. Nevertheless, while the notion of the straight line exists in all geometries, it *looks* differently in each of them. Similarly, in all geometries there exists the notion of triangle. However, in non-Euclidian geometries the sum of its angles is greater or less than 180°. So these triangles *look* differently from the Euclidian's one. This means that one postulate (in the case of the geometry the fifths one) can completely modify the perception of an incomplete system. What is the link to natural numbers? The incompleteness of N means that presently even banks use for computations a system of calculus which, for the same problem, can have different values for different banks such as we have seen for sum of angles of a triangle in different geometries. Nevertheless, there is a 'faith' that such situation cannot happen. It is thus possible that we all believe in some kind of 'practical completeness' of our natural numbers. Using this incomplete system we all believe that the formulae independent of N are somewhat properties of N that we do not need, that they are more a 'toy' for mathematicians to keep them busy in employing the undecidability results. More seriously now, as we pointed out previously, we do not suggest that the problem of

undecidability does not exist. What we try only to point out that if possible change of N will occur, we shall (or we should) simply 'be ready to deal with the situation' as we are used to be with our changing times. What it means for PS?

There are two cases to be considered but the solution is pragmatically similar in both cases:

- an incomplete axiomatic system with respect to which a specification formula is given
- an incomplete ITPPS system that provides proofs for specification formulae is built

Let us consider the first case.

We enlarge our view here by focusing not only to the consideration of incomplete system N, but to any incomplete theory T.

Classical way to the PS problem is to develop decision procedures for specification formulae. Decision procedures are interested only in providing one of the two possible answers (TRUE or FALSE). Such procedures are thus unsuitable to deal with the failure cases due to the incompleteness of T. Cartesian way is to build a 'construction' procedure which, in case of failure due to the incompleteness of T provides a suggestion for missing axioms. These axioms have then to be approved by the user who knows (or should know) by which model he wants to complete T and thus these missing axioms or new ones proposed by the user are added to T. In [48] and [40], we have presented a successful solving of a simple example in robotics that suggests two missing and immediately useful axioms for the given incomplete description of the problem. This is why this constructive Cartesian paradigm seems promising.

The classical way (building decision procedures) can thus be formalized in the following way:

$$\exists \text{ Theory } \forall \text{ Specification Formula}$$
$$\text{Has\_a\_solution\_in(Theory, Specification Formula).}$$

This means that the classical decision procedures are restricted to considering only one theory and this is another reason why they are not well suited to handle failures when the given theory is incomplete.

The Cartesian way (building a construction procedure instead of a decision procedure) can be formalized in the following way:

$$\forall \text{ Specification Formula } \exists \text{ Theory}$$
$$\text{Has\_a\_solution\_in(Theory, Specification Formula)}$$

This formalization says that the construction theorem proving procedure builds up the theory at the same time as it constructs the proof for the specification formula.

It means that instead of fixing our focus on building one closed system and arguing that such a system cannot exist, what is a mathematical truth, we change our focus to building 'evolving' systems that are changed when a necessity brings a formula by which N (or a given theory T) has to be completed. Formally, this can be expressed as a change from the classical formulation of PS problem:

$$\exists \text{ PS-System } \forall \text{ Specification Formula}$$
$$\text{Solves(PS-System, Specification Formula)}$$

to 'Cartesian' formulation that oscillates without much difficulties between this classical formulation and the following disruptive one:

$$\forall \text{ Specification Formula } \exists \text{ PS-System}$$
$$\text{Solves(PS-System, Specification Formula)}$$

We say that such an oscillation will not be a reason for unbearable difficulty since difficulties are here good for learning and discovering new paradigms and sustaining opportunities.

Once such an opening of our perspective is accepted, we can open our perspective even more as we shall show later in this paper.

As far as the second limitation is concerned (namely that there is no algorithm for a decision procedure handling PS), we first need to describe this limitation in a more pragmatic way. Gödel's results concern dealing with the formal theories in which such a decision procedure should be expressed (and, in fact, it cannot be). Without neglecting the necessary rigor in formulating an 'algorithm' for proving the specification formulae in the complete theories, we suggest that some creative features of human's mathematical brain are exploited when custom-designing an ITPPS procedure. We suggest here developing custom-specified machine learning (computational creativity) techniques. This means that we shall no more be allowed to employ the word 'algorithm' for this procedure, however, we can speak about an artificially intelligent procedure or technology for PS. In short, we shall speak of a technology and not of an algorithm. This means that we shall no more try to find an 'approximation' of a decision procedure, but we shall use our brain to invent a custom designed evolving technology.

We have thus introduced two features by which the Cartesian paradigm differs from the classical Newtonian one.

- First, as a response to the incompleteness results, to consider evolving systems instead of closed ones.
- Second, as a response to the restriction of purely formal framework, to consider a custom-designed artificially intelligent technology instead of formal decision procedures.

At a first glance our suggestions may seem too disruptive. This is why, in the next sections, we are going to give an epistemological justification provided by Cartesian Intuitionism rediscovered by our study of Descartes' work [41]. In contrast to a logical justification that provides a logical proof for a considered hypothesis, an epistemological justification consists in giving arguments confirming a reasonable character of the hypothesis and, if possible, in giving references to recognized predecessors. In our case, the predecessors are Francis Bacon by his idea of recursive long-term Progress and René Descartes by his development of Cartesian Intuitionism. Cartesian Intuitionism is counter-intuitive in usual thinking and this is also the reason why philosophical commentators of Descartes' work explained it in terms of the linear systems. This makes our task of transmitting Cartesian Intuitionism more difficult since we lack contemporary supporters. This means that the access to Cartesian Intuitionism is not an easy one and in the next

sections we give the reader an opportunity to understand why it is so. This means also that we need to present the basic notions of Cartesian Intuitionism intertwined with examples and only then, in Section V, we give a recollection of the basic notions used.

## III. NEWTONIAN AND CARTESIAN WAY OF CONCEPTION NEW SYSTEMS

The main difference between Newtonian and Cartesian paradigms is easily perceptible from comments pronounced by Newton and Descartes themselves.

Newton wrote: "If I have seen further (than you and Descartes) it is by standing upon the shoulders of Giants."

Newtonian science is thus established on logic of sequential research. In a little more formalized way, we can thus describe the Newtonian way by the sequence

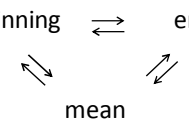
beginning … advancement-1 … advancement-2 … advancement-n … end.

Descartes wrote his first rule in the *Discourse on the Method of Rightly Conducting the Reason, and Seeking Truth in the Sciences* [27] in a following way: "The first was never to accept anything for true which I did not obviously know to be such; that is to say, carefully to avoid precipitancy and prejudice, and to comprise nothing more in my judgement than what was presented to my mind so clearly and distinctly as to exclude all ground of doubt."

Descartes speaks about the obvious truth. As says Descartes' commentator Ferdinand Alquié in [26], the act of thought that seizes the obvious truth is the intuition defined by Descartes in his *Rules for the direction of the mind* (*Regulae ad directionem ingenii* [29]). So, the study of Descartes' intuition, as presented in the book *Formal Creativity* [41] enables to notice that Cartesian science is based on logic of recursive research.

The same thing is expressed by Descartes in a little more complicated way by saying that "beginnings … can be persuaded well only by the knowledge of all the things that follow later; and that these things which follow cannot be understood well, if we do not remember all those that precede them." [26], p. 797. Thus, the Cartesian paradigm takes into account that the demarcation of a notion is not the initial stage but the final stage of its formation.

The Cartesian way can be described by the loop




where the arrow $\rightarrow$ means "leads to". This recursive loop will be illustrated in Section IV by description of the process of the creation of Peano's axioms defining natural numbers.

Thus, there are two basic styles to approach the problem of PS.

### A. Newtonian paradigm for Program Synthesis

Newtonian paradigm in conceiving a system means its linear development. As far as PS is concerned it means that the reference system of the conception of a program synthesizer, that is, the axioms, the rules of inference and the mechanism of control of the program synthesizer, as well as the reference system of a given PS problem, that is the theory in which the PS problem has to be solved, are given at the beginning by the past history of scientific research. The Newtonian paradigm in PS takes as foundation the standard knowledge of the mathematical formal framework, which inevitably inherits the negative results of Kurt Gödel. By consulting the first paragraph of Gödel's article *On formally undecidable propositions of Principia Mathematica and related systems I* [51], we can observe that the keywords of this standard knowledge are

- exactness
- formal system justified in a logical way
- methods of demonstration reduced to some axioms and rules of inference
- decision and undecidability

Previously, we have described the Newtonian style by the sequence

beginning … advancement-1 … advancement-2 … advancement-n … end.

Gödel's results are called negative because they show that the aim of synthesis of programs formulated as the "beginning" in the classic framework cannot lead to a successful 'end' of the task. In other words, they show the impossibility to define a formal logical framework containing the natural numbers allowing to approach the resolution (confirm or counter) of specifications given in a general way. Nevertheless, there are approaches to PS in the Newtonian style and they are very interesting from the short term perspective as well as from the point of view of developing long term Cartesian evolving systems.

The best-known paradigms are presented in [57], [64], [11], [10], [25], [59]. Since the problem of proving by induction specification formulas, i.e., formulas containing existential quantifiers is very difficult, researchers focused on the problem of proving purely universally quantified formulas and on treating formulas with existential quantifiers by assisting the users in developing their own proofs. The best known are the system ACL2 [12], the system RRL [54], the system NuPRL [20], the Oyster-Clam system [14], the extensions of ISABELLE [60], [30], the system COQ [9], Analytica [bauer01], KeY [7], HipSpec [19], Zeno [65] and Matita Proof Assistant [3]. All the mentioned approaches have done a very good work in modelling *human reasoning* by exploring possibilities of *transformational* methods to inductive theorem proving and PS. The construction calculus of [21], that is the basis of the system COQ, is a constructive way of *representing* transformational methods. The paradigm presented in the next section attempts to find a *constructive* way of solving an 'almost' same problem by modelling *human creativity* based on Cartesian style of research.

### B. Cartesian paradigm for Program Synthesis

Cartesian paradigm for PS is based on a logic of recursive research, where the reference system of the ITPPS

system as well as the reference system of PS problem are formulated hand in hand with the development of the solution, and where the exact demarcation of the both reference systems is the final stage of the process, and is too a part of the solution.

Recall that the Cartesian paradigm takes into account that the demarcation of a notion is not the initial stage but the final stage of its formation. The Cartesian paradigm thus specifies at the beginning the reference system in an informal way only. It is much like a hypothetico-deductive method.

The hypothetico-deductive method is a procedure of construction of a theory that consists in putting, at the start, a certain number of loosely defined concepts or proposals that are obtained by a study of experiments undertaken to specify these starting concepts or hypotheses. Then, by deductive reasoning, are obtained postulates that, when they are true, confirm the effectiveness of chosen concepts and hypotheses. If they are not true, the problem, because of the loose definitions of concepts, allows their new reformulation and the process is thus repeated on these new still loosely defined reformulations.

In contrast to hypothetico-deductive method that proceeds by deductive reasoning to access the 'truth', Cartesian paradigm uses Cartesian Intuition to access to 'truth', i.e., to the final description and justification.

Furthermore, in contrast to Newtonian paradigm and hypothetico-deductive method, in Cartesian style one can specify even the goal in a rather 'vague' manner. This is why we introduced the term of 'quite precise' purpose to indicate that this formulation, though informal, must describe a reasonable project.

For the construction of recursive programs from formal specifications, it is possible to give a 'quite precise' purpose by considering PS as a problem of realization or creation, rather than a decision-making problem. We adopted this paradigm when starting to develop the *Constructive Matching Methodology* (*CMM*) for Program Synthesis in 1983 [32]. In contrast with the Newtonian paradigm, the keywords of our particular Cartesian paradigm are

- realization and creativity
- system justified in an epistemological way
- methodology of construction
- realization of a program or sufficient conditions for the realization of such a program.

The most suitable way is thus to consider *CMM* as a technology (in a general sense) rather than a theory. The next section explains the main differences between a mathematical theory and an epistemological technology from the point of view of Newtonian construction and Cartesian creation.

## IV. Newtonian construction versus Cartesian creation

In this section, in order to underline the main differences between a Newtonian mathematical theory and an epistemological Cartesian technology, we shall be interested in the set of natural numbers N, seen here as a creation model for particular complex systems. More precisely, we shall point out the difference between the use (Newtonian) and the creation (Cartesian) of Peano's axioms.

Peano's axioms define the arithmetic properties of natural numbers N. These axioms include a constant symbol 0 and unary function symbol S. These axioms are usually used to build formal proofs about natural numbers. This section does not deal with the topic of theorem proving. It deals with the topic of understanding and reasoning about the construction of Peano's axioms, that is the creation process involved in their building.

Supposing that the membership relation "$\in$" and the equality "=" are already defined, the basic Peano's axioms read:

A1. $0 \in N$.
A2. if $n \in N$ then $S(n) \in N$.
A3. for all $n \in N$, $S(n) \neq 0$.
A4. for all $n, m \in N$, if $S(n) = S(m)$, then $n = m$.
A5. if M is a set such that
- $0 \in M$, and
- for every $n \in N$, if $n \in M$ then $S(n) \in M$

then M contains every natural number.

In order to tackle the difference between the use and the creation of these five axioms we need to precisely specify the difference between synergy and symbiosis.

An object is constructed *synergistically* when it can be considered as a result of the application of some specific tools from an existing tool-box. This tool-box represents all the tools that have been developed in all scientific domains beforehand and, usually, for various purposes. These tools are not built in such a way that one calls another tool to solve one of its problems before active tool has completed its computations. That is, tool B can call on tool A in one way only: the input of B contains a part of A computations, once A computations have been all achieved. It follows that these tools must be used and constructed independently of each other. The synergic construction is thus the main feature of Newtonian conception of independent modules for which it is meaningful to consider and prove properties independently of the whole system. For instance, the termination of rippling is proved by the team of Alan Bundy in [6], while the second order unification that is used by rippling (see [15]) is not at all considered.

In contrast to this, an object is conceived *symbiotically* when its parts, maybe seemingly independent (as it is the case for lichen that is a symbiotically living fungus and alga), have, during the conception process, no meaning as isolated entities. It means also that a slight change of one part influences the others and the whole as we illustrate below. The symbiotic composition is the main feature of the intuition defined by Descartes in his *Regulae ad directionem ingenii* [29].

Now, what we can underline about Peano's axioms is that their ***use*** is synergetic, while their ***construction*** process is symbiotic. In other words, when ***using*** them, we can use several axioms as being independent entities and the

constructing elements 0, S, and N can be considered as isolated from each other, though they are interdependent elements as show A1 and A2. The following example will show in which way Peano's axioms construction process is of symbiotic nature.

Let us first consider axiom A1 dealing with 0 and N. However, the full meaning neither of 0 nor of N is explained in this first axiom. (Recall that in hypothetico-deductive method the first notions, at the beginning, may be specified in a vague manner.) In particular, from this axiom we cannot conclude that 0 is a basic element and that N is the final object we want to define. The axiom A1 expresses only an interdependence between two symbols 0 and N. The symbol $\in$ does not tell more than 0 is an "element" and N is one of sets to which this element belongs. There is no difference, except substitution, between A1 and B1: "rose $\in$ garden". This means that the creator of Peano's axioms has already in mind a "vision" or an "informal specification" (or, as we say, a 'quite precise' purpose) of what 0 and N mean for him in this first axiom. This is why, in the cyclic presentation of Cartesian thinking (see Section III), there are two arrows, one linking beginning to the end and one doing the reverse. In other words, writing this first axiom, the axiom's creator intuitively knows what 0 and N will be once their description has been completed, i.e., when all the necessary (in this case five) axioms will be provided. In the creator's mind, the first axiom contains implicitly and intuitively all the remaining axioms and all the axioms are constructed from his/her intuitive vision of the "whole", i.e., N. Therefore, 0 and S do not belong to an already given tool-box and the meaning of 0, S and N in the construction process *is custom-made*. Moreover, 0, S, and N are symbiotic during the construction process and they are not synergetic parts. During the construction process, N steers the realization of 0 and S and vice versa, they cannot be considered as isolated already known elements. In this sense, the Newtonian paradigm is unable to provide and explain the process of creation of N and others systems that rely on Cartesian Paradigm. This is also why we say that N is a complex system, even if its description is short one.

We shall below present an example illustrating this symbiotic feature. However, we need first to introduce some more notions.

N is constructed with the help of three "elements", namely 0, S and N itself. Note that self-reference is already acknowledged as a constructive recursive 'trick'. (Look in Section III for the presence of the 'mean' in Cartesian recursive cyclic thinking). These construction parts are usually named 'the constructors'. We have already mentioned that these parts are symbiotic during the construction process, while when using the Peano's axioms for reasoning, we may consider them synergetic "*par la pensée*" (as Descartes puts it §62 of *The Principles of philosophy* [28]). In the following, instead of 'construction' we shall call this process 'Cartesian creation' in tribute to Descartes.

Now we can illustrate the symbiotic character of the constructors 0, S and N. Let us consider Peano's axioms without A3. In such a case we have the liberty to suppose that there exists $n \in N$ such that $S(n) = 0$. Let us suppose that $S(S(0))$ is such an element. We have then $S(S(S(0))) = 0$. Let us call B3 this hypothesis. Then, A1, A2, B3, A4 and A5 constitute a meaningful definition of the set that contains three elements, namely 0, S(0) and S(S(0)). This new axiomatic definition defines a set, N3, which is finite and thus is different from the infinite set N defined by Peano's axioms. In other words, a *little change* in a property of one constructor (as we have see also in the example of geometry) altered the properties of all the constructors, including N that changed into N3. This is not the case in a synergetic construction, where a change of one construction module may influence the behaviour of the whole but has no direct effect on the other modules. This explains why we so much stress the difference between symbiotic Cartesian creation and synergetic Newtonian construction. Once a symbiotic creation of a whole is completed, we may *think* of the constructors as being "unconnected" synergetic elements. (This is also the reason why Descartes' epistemological work is misunderstood and explained in terms of linear thinking and analysis, see our critics of [56] in [41]). We just have shown that this synergetic thinking is not valid during the creation process. This is why there is also a difference between a creation process and the use of the completed whole created by the same process. Descartes specified this difference in his notions of clear and distinct perception [28]. A clear perception is typical for perception and use of synergetic systems, while clear and distinct perception is imperative for symbiotic systems.

An interesting feature of a symbiotic creation is that one cannot produce a sample or "architectural" miniature before the whole creation process is completed. Moreover, partial results are often incomprehensible outside the creation process, which works mainly with informally specified problems that must be simultaneously solved. The drawbacks we just exposed must be one of the reasons why Cartesian creation is hardly reported in the scientific communications that concentrate on the results of the creation, not on the creative process itself. Researchers (and/or referees) seem to prefer tool-box Newtonian progressive construction that provides the security of familiarity with such linear or modular processes as well as immediate gratifications. This may also explain why our original Cartesian paradigm is not followed in the research on PS.

Summarizing this section, we can say that Cartesian creation focuses on building a system, a whole, by progressively inventing symbiotic constructors. Such a progressive process is possible since the first constructors and the whole are described by a 'mere' informal specification. The standard Newtonian research is not accustomed to such an informal goal specification and it usually gathers already existing mechanisms that have been certainly not custom-designed for the given goal. This choice

leads, during the construction process, to new problems, more often related to the chosen basic tools than to the given goal (we can mention the use of the second order unification in rippling [15]). These new problems ask for a new search of already existing tools and to attempts for adapting them to the given goal, a process that tends to fail when it is completely automated. In other words, in Cartesian creation, the basic tools, i.e., constructors and the whole system are custom-made, while in Newtonian construction, the basic words are "choice" and "adaptation" of already available tools.

## V. CARTESIAN INTUITIONISM

Cartesian Intuitionism is specified by Descartes in his work mainly by four disruptive notions and the rules of his method. Namely, we have:

- a form of constructive symbiotic creation called **intuition**, in the Latin version of his *Rules for the direction of the mind* [29];
- the ability of thinking as isolated, one of many mutually dependant features (**division *'par la pensée'***) in §62 of *The principles of the philosophy* [28];
- **clear** and **distinct perception** in §45 and §46 of *The principles of the philosophy*[28];
- the **four rules of his method**, in his *Discourse on the method* [27].

These notions and rules are disruptive since they differ from linear, analytical, rigid and unemotional thinking that is usually attributed to Descartes (see, for instance, [56], [22], [23]).

The thinking of Descartes is not linear as we have illustrated by the quotation of Descartes before the recursive loop in Section III. However, the fact that his thinking is recursive is illustrated best by his method. Namely, one should ask the question: "How is his method obtained?" And the (not so) obvious answer is that *his method is conceived by his method*. This contradicts Popper who claims, in [62], that there can be no logical description of inventing new ideas. If one accepts that Descartes' notion of intuition is a logical way of inventing new ideas and that the Descartes' method describes this way, then Popper's opinion is challenged.

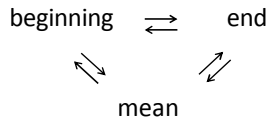
While the Descartes' thinking comprises also analysis (synergy), it is highly symbiotic. This manifests in his recursive creation, the notions of intuition (the symbiotic creation), division '*par la pensée*' and distinct perception.

Descartes' thinking is not rigid since the idea of evolving systems is comprised in the possibility of 'divine revelations' (in the rule II of his *Rules for the direction of the mind*) that have to be 'assimilated' to existing knowledge by Cartesian Intuition and deduction.

Descartes' thinking is not unemotional, as the rule XII of his *Rules for the direction of the mind* insists on employing all possible human resources in conceiving an exploitable evolving system. From a pragmatic point of view, the emotions are hidden in our technological context in the notion of 'trust' and 'faith'. With respect to its large use, Newtonian conception is highly trusted since partial results are measurable in usual ways. However, Cartesian creation cannot be easily understood and measured (thus trusted) by an external observer requesting simple explanations in Newtonian terms and measures. Partial results in Cartesian creation are more-less informal 'chunks' possibly intertwined with other 'chunks' to be yet specified as it is written in XII rule of *Rules for the direction of the mind*. On the other hand, the notion of 'faith' is, in recursive Cartesian thinking, a *technical* term that expresses the conviction about the reasonable and realisable character of the goal and about the soundness and the appropriateness of the method employed for accessing to the goal.

We can here summarize Cartesian creativity representing the Cartesian Intuitionism in three points. Cartesian creativity

(a) focalises on the problem: {∀ specification formula ∃ framework in which the given specification formula has a solution}
(b) oscillates between the problems {∃ framework ∀ specification} and {∀ specification ∃ framework}
(c) considers the creativity process in its recursive cyclic version given by the scheme




where the arrow means "steers".

These three points give to Cartesian Intuitionism the feature of a combination of what is called essentialism and existentialism within the frame of logics by Girard in [50].

## VI. CONSTRUCTIVE MATCHING METHODOLOGY

In this section we are going to

- illustrate some consequences of adopting Cartesian Intuitionism as epistemological justification of the conception of a recursive system and the difference between a Newtonian decision and Cartesian construction procedure;
- explain how the idea of evolving systems is actually performed in *CMM*;
- present an informal description of the basic constructor of *CMM*;
- present assessment and perspectives of *CMM*.

### *A.* CMM *in the light of Cartesian Intuitionism*

The basic principle of Newtonian PS system is the use of a fixed set of specific strategies in order to solve the problems that are submitted to it. In case of failure, the user is requested to provide lemmas or axioms that lead to success.

The basic principle of Cartesian PS system is also the use of a specific strategy defined by the axioms, which themselves represent the whole system. But this is true only as long as the system meets no failure. In case of failure, we build a new PS system possibly with a new solving strategy.

We already illustrated such behaviour by building the pseudo-Peano system by replacing A3 by B3 and N by N3. If this kind of incomplete natural numbers is used to prove a theorem containing the term, say S(S(S(S(0)))), the 'synthesis' will fail. In a Newtonian paradigm, the user would be asked for a lemma specific to S(S(S(S(0)))) that enables a success. In such a case our paradigm would propose to modify the system of axioms by changing B3 and N3. We fully agree that, in this particular case, a human feels the needed modification as being trivial. In consequence, let us provide a more complex example that illustrates a situation where modifying system of axioms defining PS mechanism is not trivial.

In [8], a Newtonian system called Otter-Lambda is presented, together with several examples of its execution. We have chosen among them a formula

$$\forall\, a\, \forall n\, \{\, (S(0) < a \Rightarrow n < \exp(a,n))\, \} \qquad (*)$$

The Otter-Lambda system fails when the basic information relative to (*) is given as a recursive definition of the exponentiation function exp with respect to the second argument:

(A1) $\exp(u,0) = s(0)$
(A2) $\exp(u,S(v)) = \exp(u,v)*u$

of the addition and of the multiplication with respect to the first argument:

(A3) $0 + u = u$
(A4) $S(v) + u = S(v + u)$
(A5) $0 * u = 0$
(A6) $S(v) * u = (v * u) + u$

The definition of < is also recursive and given as:

(A7) $0 < y$, if $y \neq 0$
(A8) $S(v) < y$, if $v < y \;\&\; y \neq S(v)$

Since the Otter-Lambda system fails, it requests some help from its human user. In [8], the user is able to provide the following lemmas that enable Otter-Lambda to complete the proof of (*).

(A9) $\text{not}(u<v)$ or $(x*u < x*v)$ or $\text{not}(0 < x)$
(A10) $(x < y)$ or $(y \leq x)$
(A11) $\text{not}(y \leq x)$ or $\text{not}(x < y)$
(A12) $\text{not}(u < v)$ or not $(v \leq w)$ or $(u < v)$
(A13) $\text{not}(S(0) < z)$ or $\text{not}(0 < y)$ or $(S(y) \leq z*y)$
(A14) $0 + x = x$

We applied to the same problem our Cartesian paradigm, which does not suggest getting any user's help. The system determines n as the induction variable, since it occurs in recursive arguments of all the functions and predicates and the other possible candidate variable a occurs in the non-recursive first argument of the function exp, which would stop the evaluation process in an inductive proof. Nevertheless, our method notices at once a probable source of trouble: the predicate < is defined recursively with respect to its first argument, while, in (*), the induction variable n occurs also in second position of the predicate <. At this stage, the method could suggest the user to provide a definition of < with respect to both argument (this would actually fail), or with respect to the second argument (this would fail as well), or else, a non recursive definition (that would succeed). As already mentioned, our method is not expected to call on its user, and thus it will proceed by calling a custom-designed constructor named "Synthesis of Formal Specifications of Predicates". The initial results in developing this constructor are described in [49]. The symbiotic system *CMM* with this constructor included generates the following formal specification for predicate <:

$$x < y \Leftrightarrow \{\, \exists z\; y = S(x + z)\, \}.$$

With this new definition (*) is transformed into

$$\forall a\, \forall n\, \exists z\, \{\, (S(0) < a) \Rightarrow (\exp(a,n) = S(n + z))\, \}. \qquad (**)$$

Note that this last formula is a specification formula by introducing the existentially quantified variable z. *CMM* is then able to prove it (without interaction with the user). *CMM* generates and proves autonomously the following lemmas (that are formal specifications for six auxiliary sub-routines of the program specified by (**)):

L1. $\forall\, a\, \forall n1\, \forall b\, \exists z1\, \{\, S(0) < a \Rightarrow (n1 + b)*a + a = SS(n1 + z1)\, \}.$
L2. $\forall\, a\, \forall b\, \exists z2\, \{\, S(0) < a \Rightarrow b*a + a = SS(z2)\, \}.$
L3. $\forall\, a\, \exists z7\, \{\, S(0) < a \Rightarrow a = SS(z7)\, \}.$
L4. $\forall\, a\, \forall m\, \forall d\, \exists z5\, \{\, S(0) < a \Rightarrow (m + d) + a = S(m + z5)\, \}.$
L5. $\forall\, a\, \forall d\, \exists z3\, \{\, S(0) < a \Rightarrow d + a = S(z3)\, \}.$
L6. $\forall\, a\, \exists z4\, \{\, S(0) < a \Rightarrow a = S(z4)\, \}.$

This example illustrates all three points (a), (b), (c) of Cartesian Intuitionism in that, when meeting failure, a need for a complementary constructor transforming a recursive definition of a predicate into a non-recursive equivalent is informally specified. Then, the successful formalized design of this constructor enlarges the power of *CMM* and thus modifies the whole *CMM* that is ready, when necessary, to be once again modified.

The basic constructor of *CMM* is presented in [2]. With respect to the notions introduced in this paper, we readapt that presentation in Section VI.C. The other constructors of *CMM* specified so far are described in our publications up to 2001 [38]. Some of these constructors were implemented in the system *Proofs Educed by Constructive Matching for Synthesis* (PRECOMAS) [36], [39]. With respect to the symbiotic character of the constructors and the need of treating the failure analysis by developing further constructors, we have interrupted the implementation of PRECOMAS in 1990 and focused on developing an epistemic justification (see [41]) hand in hand by reformulating our work in terms of this justification.

The example presented in this section helps us to illustrate some consequences of adopting Cartesian Intuitionism as epistemological justification of the conception of a recursive system and the difference between a Newtonian decision and Cartesian construction procedure.

First of all, the development of *CMM* is, in this stage, by-hand made. This is because we seek for a methodology, i.e., a conceptual capture of the all problems that are related to inductive theorem proving viewed as a construction procedure. We seek (by-hand) for all the constructors of the resulting system by the on-purpose justified Cartesian method called *Formal Creativity* and described in our book [41]. Classical Newtonian approaches focus on implementing procedures that are checked out with respect

to some benchmark formulas. The systems are considered as failing when they do not provide a decision in some time constraint. For instance, [53] refers to experiments in which timeout is set to 30 seconds. The failure of the system is in this sense unproductive for further research in inductive theorem proving. This is why the Newtonian research is very quick in producing implementations but slow in providing conceptual descriptions of the problems that could point out the directions in which the research has to be done. As we mention in the next section, our by-hand research allowed us to formulate already several major problems.

Second, instead of a modular system for which the properties of modules are formulated and proved independently of the whole system, the Cartesian approach allows us to consider the whole system as an axiomatic system for which, as for Peano's axioms, there can be only a pragmatic justification expressed somewhat unscientifically by the sentence: 'The justification of the system is obvious as it was conceived in such a way that it works'. However, this justification is scientifically valid when one looks at it from the point of view of Cartesian notion of Intuition obtained by (and representing itself) a 'luminous calculus' (see rule II in *Regulae ad directionem ingenii* [29] and Bacon's 'luminous experiment' referred to in *Novum Organum* [5]). Because of its powerful potential for generating new ideas (similarly to lateral thinking [23]), the term 'luminous' should thus become actual even today in all scientific research.

Third, since Cartesian Intuitionism justifies employing all possible human resources, *CMM* relies heavily on the idea of using machine learning (computational creativity) techniques whenever it will be appropriate.

Fourth, as we shall illustrate below, our approach generates multiple auxiliary procedures. This is not possible with second order unification that is able, as in rippling ([52], [15]), to generate auxiliary procedures on one level only (i.e., during the execution, the unification does not generate further auxiliary procedures) and only with already defined functions.

### *B. Conceptual oscillation of* CMM

As we suggested in Section II, we are interested in conceiving evolving systems. Such systems are conceived in oscillatory way. We call oscillatory a paradigm in which, to find an optimal result of a definition of a theory, we oscillate between both specifications of the problem

$$\{\exists \text{solution } \forall \text{problem}\} \text{ and } \{\forall \text{problem } \exists \text{solution}\}$$

More exactly, our paradigm oscillates between a Newtonian formulation of PS and a Cartesian formulation of the same problem. It is clear that this purpose seems very ambitious when one forgets the preliminary restrictions (not considering efficiency of synthesized programs, proofs by structural induction only, specifications formulae expressed as conjunctions of atomic formulae and even more restrictions that may come out in a further elaboration). These restrictions do not make the problem trivial; they only enable to focus on the core of the problem that we must specify and solve at first.

In practice, this oscillation is performed in the following way. For a given specification formula, we attempt to perform a constructive proof relying on the results already achieved by *CMM*. In other words, we start to solve the problem having in mind the specification '∃solution ∀problem', where the solution is the *CMM* and the problem is the given specification formula. If the power of the *CMM* is not sufficient to prove the given specification formula, by a failure analysis we try to conceptualize the problems met as methods rather than heuristics. In other words, we solve the problem by focusing on the problem '∀problem ∃solution' and then by a suitable process of conceptualization similar to hypothetico-deductive method we try to come back to the specification '∃solution ∀problem', where the solution is now the extended *CMM*. This is why this paradigm is more the one of a mathematician trying to build a new theory-technology rather than that of a programmer focusing on obtaining efficient programs.

In this way, we have conceptualized many new methods in inductive theorem proving for specification formulas, for instance: implicative generalization, predicate synthesis from formal specification, synthesis of formal specifications of predicates, introduction of universally quantified induction hypotheses whenever appropriate, a particular evaluation tool and a particular equation solving tool. We explain this conceptual richness of inspirations of *CMM* proofs by the basic method for constructing atomic formulas '*CM*-formula construction' that has been introduced in [33] and the most complete presentation of which can be found in [40]. At present we are working on a general algorithmic presentation. In contrast to the basic methods in Newtonian paradigms that rely on simplification and rewriting, our *CM*-formula construction is a constructive method and thus it is very suitable for generating missing lemmas (see Section VI.A) and even axioms when the given data are incomplete as it is illustrated in [48]. *CMM* is even suitable for proving purely universally quantified theorems even if the proofs are generally more complicated, since the basic method is construction and not simplification. The advantage lies however in the fact that, during a proof of a universally quantified formula, a formula containing existential quantifiers can be generated, which replaces the problem of unification in the framework of PS and thus it seems to be conceptually more powerful.

### *C.* CM-*formula construction*

**Formulation**

In the following, for simplicity, let us suppose that the formula to be proven has two arguments, that is to say that we need to prove that $F(t_1,t_2)$ is true, where F is the given theorem. We introduce a new type of argument in the atomic formula, which has to be proven true. We call them **pivotal arguments**, since the focus on them allows reducing what is usually called the search space of the proof. These arguments are denoted by $\xi$ (or $\xi$' etc.) in the following. The pivotal argument replaces, in the first step, in a purely syntactical way, one of the arguments of the given formula. The first problem is thus choosing which of the arguments will be replaced by a pivotal argument $\xi$.

In the first step, let us suppose that we have chosen to work with $F(t_1,\xi)$. In an artificial, but custom-made manner,

we state C = {ξ | $F(t_1,\xi)$ is true}. Except the syntactical similarity with the formula to be proven, there is no semantic consideration in saying that $F(t_1,\xi)$ is true. It simply represents a 'quite-precise' purpose of trying to go from $F(t_1,\xi)$ to $F(t_1,t_2)$. We thus propose a 'detour' that will enable us to prove also the theorems that cannot be directly proven by the so-called simplification methods, i.e., without this 'detour'. In the second step, via the definition of F and those involved in the formulation of the term $t_1$, we look for the features shown by all the ξ such that $F(t_1,\xi)$ is true. Given the axioms defining F and the functions occurring in $t_1$, we are able to obtain a set $C_1$ expressing the conditions on the set { ξ } for which $F(t_1,\xi)$ is true. In other words, calling 'cond' these conditions and $C_1$ the set of the ξ such that cond(ξ) is true, we define $C_1$ by $C_1$ = {ξ | cond(ξ)}. We can also say that, with the help of the given axioms, we build a 'cond' such that the formula: $\forall \xi \in C_1$, $F(t_1,\xi)$ is true. In the third step, using the characteristics of $C_1$ obtained in the second step, the induction hypothesis is applied. Thus, we build a form of ξ such that $F(t_1,\xi)$ is related to $F(t_1,t_2)$ by using the induction hypothesis. For the sake of clarity, let us call $\xi_C$ the result of applying the induction hypothesis to $C_1$ and $C_2$ so obtained is thus such that $F(t_1,\xi_C)$ is true. We are still left with a hard work to do: prove that $t_2$ belongs to $C_2$, i.e., to prove that $\xi_C$ and $t_2$ can be made identical, i.e., that $t_2$ matches $\xi_C$. In the case of the success, this completes the proof. In the case of a failure, a new lemma $\xi_C = t_2$ with an appropriate quantification of the involved variables is generated. In some cases, an infinite sequence of lemmas may be generated. *CMM* is conceived in such a way that the obtained sequence is well-behaving (see [33]) in the sense that one can apply a generalization technique to obtain a more general formula that has to be proved. This formula covers logically the infinite sequence of lemmas and thus it fills the gap that cannot be overcome by purely deductive formal approach to theorem proving.

The works in [39] and [40] give a detailed description of handling the pivotal argument in a rigorous framework. In [2], we illustrate *CM*-formula construction on a simple synthesis of a program for displaying the last element of a non-empty list. This is why we can afford illustrate an incomplete example, namely how *CM*-formula construction generates L1 for (**) from Section VI.A.

**Example**

The formula (**) reads

$$\forall a\ \forall n\ \exists z\ \{\ (S(0) < a) \Rightarrow (exp(a,n) = S(n + z))\ \}.$$

The lemma L1 is generated in course of the induction step for (**) and we shall thus focus only on this general case of inductive proof. With respect to the recursive analysis of the given definitions (see Section VI.A), the induction variable here is n. It varies over natural numbers, and so, in the induction step, n = s(n1) for some natural number n1. We shall denote by sf the Skolem function corresponding to the existentially quantified variable z in this formula, i.e., z = sf(n,a).

In the induction step for (**), the method assumes a > S(0) and, since n is represented by S(n1), the induction hypothesis is (see [16])

$$\exists\ e\ exp(a,n1) = S(n1+e). \qquad (A)$$

In this induction hypothesis,

$$e = sf(n1,a). \qquad (B)$$

Assuming S(0) < a, the goal is to prove

$$z\ exp(a,S(n1)) = S(S(n1)+z). \qquad (C)$$

Here, z = sf(S(n1),a). Since the term S(S(n1)+z) contains an existentially quantified variable, namely z, this term becomes the pivotal argument ξ. In the first step, ξ syntactically replaces the term S(S(n1)+z). The method gets an artificially built set

$$C = \{\xi \mid exp(a,S(n1)) = \xi \text{ is true }\}.$$

In the second step, the term exp(a,S(n1)) is evaluated using the axiom (A2). C changes to

$$C_1 = \{\xi \mid exp(a,n1)*a = \xi \text{ is true }\}.$$

In the third step, $C_1$ becomes semantically related to (**) by the application of the induction hypothesis. By the application of the induction hypothesis the method obtains

$$C_2 = \{\xi_C \mid S(n1+e)*a = \xi_C \text{ is true }\}.$$

This, by the application of (A6) gives

$$C_3 = \{\xi_C \mid (n1+e)*a + a = \xi_C \text{ is true }\}.$$

In the fourth step, the method has to check whether the second term, i.e., S(S(n1+z)), belongs to $C_3$. This leads to the problem of solving the equation

$$\exists\ z\ (n1+e)*a + a = S(S(n1+z)). \qquad (D)$$

This equation cannot by solved by *CM*-term transformer (presented in [35]) and thus the method generates a new lemma.

Since we reserve the name e for existentially quantified variables coming from induction hypotheses, we rename e to b and thus the lemma noted in Section VI.A as L1 is generated, i.e.,

$$\forall\ a\ \forall n1\ \forall b\ \exists z1\ \{\ S(0) < a \Rightarrow (n1 + b)*a + a = SS(n1 + z1)\ \}.$$

Let us denote by sf1 the Skolem function for z1, i.e., z1 = sf1(n1,b,a). By (D) we thus obtain the relation between sf and sf1, namely z in (D) is sf(S(n1),a) = sf1(n1,e,a), which, by (B), gives the partial program

$$sf(S(n1),a) = sf1(n1,sf(n1,a),a),\ \text{if } a > S(0). \qquad (E)$$

The method is the called recursively to prove L1 and all the lemmas that are generated.

This example illustrates well that *CM*-formula construction is an artificial, custom-made method. It is also useful as a suggestion to use PS in the role of a powerful 'unification' tool. For rather complex problems solved by *CMM* the reader can consult the already mentioned [43] but also [37], [40] and [42].

### *D. Assessment and perspectives of* CMM

The stage relative to the procedure of demonstration was elaborated in all our publications until 2000 [38]. An experimental system called *PRECOMAS* (Proofs Educed by Constructive Matching for Synthesis) showing the soundness of the *CM*-formula construction was implemented in the 90s [36].

The stage relative to the specification of the intermediate lemmas is now in a good shape. It concerns also the scientific domain known as 'computational creativity' [46], [47].

The stage that concerns the clear and distinct perception (in the Cartesian sense) of the targeted strategic recursive axiomatization has begun in the article [44]. It must be improved and pursued by an adequate formalization of different fundamental interrelated problems that are met in the oscillatory design of the recursive systems, namely

- one - multiple (part - whole)
- static - dynamic (permanence - change)
- finite - infinite (visible - invisible)
- complete - incomplete (rigor - creativity).

In Program Synthesis, the problem between a whole and its parts is expressed as a strong and special interdependence between the diverse parts of the system, because a part or the whole can itself assume the failure cases and the weaknesses of the other parts. For example, the failure of a resolution of an equation can call in a recursive way the system for help (as we have illustrated above). Or, the deductive parts of the system can call inductive parts, and vice versa. This particular interdependence is described by Descartes as "the distinction, which is made by the thought" (distinction '*par la pensée*') presented above as "the ability of thinking as isolated, one of many mutually dependant features."

The problem of the oscillation between a static representation and a dynamic representation appears in the process of search and creation of the structures and the mechanisms of the control of proofs. This process oscillates between an already partially formalized shape and an informal shape of a given mechanism (see rule XII in *Regulae ad directionem ingenii* [29]). As we said above, the definitive demarcation that consists in fixing a final version of the mechanism is only made at the end of development of the whole system (i.e., by the Cartesian Intuition).

The problem of the regulation of the finite and the infinite appears in PS especially by the fact that an infinite visible variety of possible formal specifications must be managed by finite invisible structures. In other words, the final system of PS has to represent a finite solution of the infinite problem 'to think of everything at the same time'. So, for this problem, Ackermann's function in an oscillatory version models in a curiously proper way the solution that we envisage for this problem.

The problem of the oscillation between completeness and incompleteness is described in an informal way by the notion of pulsation that allows a controlled oscillation between rigor and creativity. In a concrete way, the *CM*-formula construction allows such a controlled oscillation and has influences on all the *CMM*.

These four fundamental problems are stemming from our perception of Cartesian Intuitionism. They appear as ideas of directions to be developed and to be formalized. These tasks will continue in our future work.

These problems are not, however, the only topics we shall deal with. In near future we intend to describe how the principles behind *CM*-formula construction apply in the design of evolving systems in general and in the evolving recursive *CMM* in particular. We have tackled this problem in an informal way in our book [41].

The power of *CMM* was illustrated on a number of interesting problems such as n-queens [34], the quotient and the rest of two numbers [32], a problem in robotics [45] and more recently the construction of a definition of Ackermann's function with respect to the second variable [43]. This last illustration is important because it shows the capacity of *CMM* to find another form of defining axioms, the final version of which is not known beforehand.

## VII. Advantages and drawbacks

A Newtonian paradigm has the enormous advantage of being fully accepted and respected in the scientific community. As far as Program Synthesis is concerned, it allows bringing quickly highly user-dependent implementations. Its main drawback is however that it provides no clear future orientations of the research on inductive theorem proving. This manifests by a long pause in Newtonian research starting in ninetieth and followed by resurgence around 2010 [66], [67], [63], [58]. These new approaches deviate from the original PS problem, which is that of a user-independent strategy for proving theorems, by introducing a library of efficient templates suitable for one kind of problems or by identifying interesting classes of algorithms and by capturing as much generic algorithm design knowledge as possible in one place. Their contribution is practically very useful in the short term perspective but, in the long term one, it represents the work on building libraries for semantic classes of programs and a need for big data handling. This is an economically useful orientation. However, from the point of view of scientific curiosity, it misses the (reasonable) ambition of Cartesian Paradigm.

In this paper, we have illustrated that Cartesian paradigm is suited for generating a sequence of missing sub-routines. That is not yet possible in simplifications approaches.

The advantage of Cartesian Paradigm lies in its long-term vision of evolving (though disruptive) theorem proving systems. However, this long-term and disruptive perspective is not easily accessible, and makes it somewhat unattractive for researchers seeking quick gratification.

In short, Cartesian paradigm is an advantageous paradigm since it has

- a solid epistemic justification (this somewhat smoothens up its disruptive character);

and it enables:

- accepting Gödel's results in proactive way;
- considering PS as a problem of a developing a technology rather than a procedure of decision;
- introducing the idea of creating complex evolving systems as a complement to the largely accepted idea of observing and manipulating such systems (e.g., by Machine Learning, Knowledge Discovery, Data Mining and so on);
- allows placing PS in the context of creating evolving, recursive and symbiotic systems;
- allows integrating human creativity directly into the systems to be conceived.

The main drawbacks of Cartesian paradigm are the following:

- consideration of PS problem as a problem of a disruptive technology is not yet widespread;
- lack of availability for formations teaching to think in terms of evolving, recursive and symbiotic systems;
- creation of such systems is slow and difficult to evaluate by external observers;
- people used to linear conception of systems are disturbed by necessity to conceive at first mentally all the 'informal chunks' (i.e., constructors) of such systems before the actual implementation starts;
- necessity of collaborations between PS and several non-deductive methods such as they exist in Machine Learning, Data Mining, Knowledge Discovery and other domains.

The difficulty of PS in general confirms that we cannot expect a rapid development of powerful general purpose oriented industrial systems. Nevertheless, both paradigms have an important place in contemporary research.

## VIII. CONCLUSION

In this paper, we have formulated two fundamental questions, namely whether the logical limits of Gödel's results can be 'overcome' by a pragmatic reformulation of the PS problem and whether there can be a custom-designed theorem prover for PS. The paper justifies our positive answers to these questions by putting forward the foundations for Newtonian and Cartesian systemic paradigms and by indicating the necessity of their synergy.

In contrast to Newtonian theoretical metrics of evaluation of PS systems, the paper suggests the metrics of robustness and conceptual symbiotic expressed by the measure of Cartesian Intuition.

This paper presents Cartesian and Newtonian paradigms in PS to a larger extent than our publications [1] and [2], namely by

- mentioning the main orientation of recent works on PS in Newtonian paradigm;
- comparing this orientation with our Cartesian approach
- thorough describing the epistemological background for the Cartesian Intuitionism;
- illustrating
  - some consequences of adopting Cartesian Intuitionism as epistemological justification of the conception of a recursive system and
  - the difference between a Newtonian decision and Cartesian construction procedure;
- presenting an expansion of the experiment presented in [1];
- illustrating that Cartesian Intuitionism can be looked upon as a 'generator of new ideas' not only in the form of missing axioms and lemmas in theorem proving process but also in the form of notions proper to custom-made creation of evolving symbiotic systems.

So far, the Newtonian paradigm has been very successful in producing systems that request human help as soon as some non-trivial 'creativity' is needed in order to provide a lemma or a heuristic not already included in the system library. Since one of our ultimate goals is modeling some form of mathematical systemic creativity by building a computer simulation of these creative steps, we had to adopt a new perspective, the one of Cartesian Intuitionism.

Cartesian Paradigm is disruptive not only by its evolving, symbiotic and recursive character but also because it brings an unusual action-oriented perspective to interpreting Gödel's results.

The Cartesian Paradigm faces more obstacles than the Newtonian one because of its complexity and because neither a superficial external observation (due to the presence of the symbiotic thinking in Cartesian Intuition) nor the sequential transmission (due to the use of recursion) nor a rigid formal perception (due to its evolving character) are suited to the appreciation of the work made in this recursive way. One of our goals in this paper was a call-to-action for tearing down these artificial obstacles immanent within the realm of the Newtonian paradigm. One of our goals was also to stress out the complementary and highly non-competing character of both paradigms.

## ACKNOWLEDGMENT

I would like to express my warmest thanks to Michèle Sebag, my research group director at L.R.I., and Yves Kodratoff who helped me to express the ideas presented in this paper. Thanks to Veronique Benzaken for her moral support. The feedback provided by Dieter Hutter and the comments of referees of this Journal as well as of the ones of COGNITIVE 2013 and ICONS 2014 contributed to improve the quality of this paper.

## REFERENCES

[1] M. Franova, "A Cartesian methodology for an autonomous program synthesis system," in M.Jäntti, and G. Weckman, Eds., Proc. of ICONS 2014, The Ninth International Conference on Systems; ISBN: 978-1-61208-319-3, pp. 22-27, 2014.

[2] M. Franova, "Cartesian Intuitionism for program synthesis," in. S. Shimizu, and T. Bosomaier, Eds., Cognitive 2013, The Fifth International Conference on Advanced Cognitive Technologies and Applications; ISBN: 978-1-61208-273-8, pp. 102-107, 2013.

[3] A. Asperti, C. S. Coen, E. Tassi, and S. Zacchiroli, "User interaction with the Matita Proof Assistant," Journal of Automated Reasoning, August 2007, Volume 39, Issue 2, pp. 109-139, August 2007.

[4] G. Bachelard, The Formation of the Scientific Mind: A Contribution to a Psychoanalysis of Objective Knowledge. Clinamen Press Ltd., 2001.

[5] F. Bacon, Novum Organum. P.U.F, 1986.

[6] D. A. Basin and T. Walsh, "A calculus for and termination of rippling," JAR, Volume 16, Issue 1-2, pp. 147-180, March 1996.

[7] B. Beckert, R. Hähnle, and P. H. Schmitt, Eds., Verification of Object-Oriented Software. The KeY Approach. Lecture

Notes in Computer Science, Volume 4334, Springer-Verlag, 2007.

[8] M. Beeson, "Mathematical induction in Otter-Lambda," Journal of Automated Reasoning, Volume 36, Issue 4, pp. 311-344, April 2006.

[9] Y. Bertot and P. Casteran, Interactive Theorem Proving And Program Development - Coq'art: The Calculus Of Inductive Constructions. Springer-Verlag, 2004.

[10] W. Bibel, "On syntax-directed, semantic-supported program synthesis," Artificial Intelligence 14, pp. 243-261, 1980.

[11] S. Biundo and F. Zboray, "Automated induction proofs using methods of program synthesis," Computers and Artificial Intelligence, 3, No. 6, pp. 473-481, 1984.

[12] R. S. Boyer and J S. Moore, A Computational Logic Handbook. Academic Press, Inc., 1988.

[13] A. Bundy, "The automation of proof by mathematical induction," in A. Robinson and A. Voronkov, Eds., Handbook of Automated Reasoning, Volume I; North-Holland, pp. 845-912, 2001.

[14] A. Bundy, F. Van Harnelen, C. Horn, and A. Smaill, "The Oyster–Clam system," in Stickel, M.E. Eds. 10th International Conference on Automated Deduction, Lecture Notes in Artificial Intelligence, Volume 449, Springer, pp. 647–648, 1990.

[15] A. Bundy, D. Basin, D. Hutter, and A. Ireland, Rippling: Meta-Level Guidance for Mathematical Reasoning; Cambridge University Press, 2005.

[16] R. M. Burstall, "Proving properties of programs by structural induction," Computer J., Volume 12, Issue 1, pp. 41-48, 1969.

[17] R. Bodik and B. Jobstmann, "Algorithmic program synthesis: introduction," in International Journal on Software Tools for Technology Transfer, Volume 15, Issue 5-6, pp. 397-411, October 2013.

[18] J. Chazarain and S. Muller, "Automated synthesis of recursive programs from a "forall" "exists" logical specification," Journal of Automated Reasoning, Volume 21, Issue 2, pp. 233-275, October 1998.

[19] K. Claessen, M. Johansson, D. Rosén, and N. Smallbone, "Automating inductive proofs using theory exploration," Automated Deduction – CADE-24, Lecture Notes in Computer Science, Volume 7898, pp. 392-406, 2013.

[20] R. L. Constable, Implementing Mathematics with the Nuprl Proof Development System. Prentice-Hall, Inc., Englewood Clifs, New Jersey, 1986.

[21] T. Coquand and G. Huet, "Constructions: A higher order proof system for mechanizing mathematics," in B. Buchberger, Eds., EUROCAL'85, European Conference on Computer Algebra, Proceedings Volume 1: Invited Lectures; April, Springer-Verlag, Linz, Austria, pp. 151-185, 1985.

[22] A. Damasio, Descartes' Error: Emotion, Reason and the Human Brain. Vintage, 2006.

[23] E. de Bono, Serious Creativity: Using the Power of Lateral Thinking to Create New Ideas. HarperCollins, 1992.

[24] L. De Moura and N. Bjorner, "Satisfiability modulo theories: an appetizer," Formal Methods: Foundations and Applications; Lecture Notes in Computer Science, Volume 5902, pp. 23-36, 2009.

[25] N. Dershowitz and U.S. Reddy, "Deductive and inductive synthesis of equational programs," JSC Volume 15, Nos. 5 and 6, pp. 463-466.

[26] R. Descartes, Oeuvres philosophiques (3 vol.), Ed. F. Alquié, T. 1; Classiques Garnier, Bordas, 1988.

[27] R. Descartes, "Discours de la méthode pour bien conduire sa raison et chercher la vérité dans les sciences," in R. Descartes, Oeuvres philosophiques (3 vol.). Edition de F. Alquié, T. 1; Classiques Garnier, Bordas, pp. 567-650, 1988.

[28] R. Descartes, "Les principes de la philosophie," in R. Descartes, Oeuvres philosophiques (3 vol.). Edition de F. Alquié. T. 3; Classiques Garnier, Bordas, pp. 87-525, 1989.

[29] R. Descartes, "Regulae ad directionem ingenii," in R. Descartes, Oeuvres complètes (11 vol.). Edition Adam et Tannery. T. 10; Vrin, Paris, pp. 359-469, 1996.

[30] L. Dixon and J. Fleuriot, "IsaPlanner: a prototype proof planner in Isabelle," in F. Baader, Eds., CADE-19, LNAI 2741, pp. 279–283, 2003.

[31] J. Endrullis, H. Geuvers, J. G. Simonsen, and H. Zantema, "Levels of undecidability in rewriting," Information and Computation archive, Volume 209, Issue 2, pp. 227-245, February 2011.

[32] M. Franova, "Program synthesis and constructive proofs obtained by Beth's tableaux," in R. Trappl, Eds., Cybernetics and System Research 2; North-Holland, Amsterdam, pp. 715-720, 1984.

[33] M. Franova, "CM-strategy : A methodology for inductive theorem proving or constructive well-generalized proofs," in A. K. Joshi, Eds., Proceedings of the Ninth International Joint Conference on Artificial Intelligence; August, Los Angeles, pp. 1214-1220, 1985.

[34] M. Franova, "An implementation of program synthesis from formal specifications," in Y. Kodratoff, Eds., Proceedings of the 8th European Conference on Artificial Intelligence; August 1-5, Pitman, London, United Kingdom, pp. 559-564, 1988.

[35] M. Franova, Fundamentals of a new methodology for program synthesis from formal specifications: CM-construction of atomic formulae. Thesis, Université Paris-Sud, November, Orsay, France, 1988.

[36] M. Franova, "PRECOMAS 0.3 user guide," Rapport de Recherche No.524, L.R.I., Université de Paris-Sud, Orsay, France, October, 1989.

[37] M. Franova, "A constructive proof for prime factorization theorem: A result of putting it together in Constructive Matching methodology," Rapport de Recherche No.780, L.R.I., Université de Paris-Sud, Orsay, France, October, 1992.

[38] https://www.lri.fr/~mf/index.html 2014.11.11

[39] M. Franova, "PRECOMAS - An implementation of Constructive Matching Methodology," Proceedings of ISSAC'90, ACM, New York, 1990, 16-23.

[40] M. Franova, "Constructive Matching methodology and automatic plan-construction revisited," Rapport de Recherche No. 874, L.R.I., Univ. de Paris-Sud, Orsay, France, November, 1993.

[41] M. Franova, Formal Creativity: method and practice – conception of complex 'computerized' systems and epistemological patent (Créativité Formelle: méthode et pratique - conception des systèmes "informatiques" complexes et brevet épistémologique). Publibook, 2008.

[42] M. Franova, "A Construction of several definitions recursive over the variable under the exponent for the exponent function," Rapport de Recherche No.1519, L.R.I., Université de Paris-Sud, Orsay, France, June, 2009.

[43] M. Franova, "A construction of a definition recursive with respect to the second variable for the Ackermann's function," Rapport de Recherche No.1511, L.R.I., Université de Paris-Sud, Orsay, France, 2009.

[44] M. Franova, "The role of recursion in and for scientific creativity," in R. Trappl, Eds., Cybernetics and Systems 2010; Proc. of the Twentieth European Meeting on Cybernetics and System research, Austrian Society for Cybernetic Studies, pp. 573-578, 2010.

[45] M. Franova and Y. Kodratoff, "How to clear a block with Constructive Matching methodology," in J. Mylopoulos, R. Reiter, Eds., Proceedings of the Twelfth International Joint

Conference on Artificial Intelligence, IJCAI'91; Morgan Kaufmann, pp. 232-337, 1991.

[46] M. Franova and Y. Kodratoff, "On Computational Creativity: 'inventing' theorem proofs," Rauch J., Ras Z.W., Berka P., Eloma T. Eds., Foundations of Intelligent Systems, 18th International Symposium, ISMIS 2009, LNAI 5722, September, Springer, pp. 573-581, 2009.

[47] M. Franova and Y. Kodratoff, "Two examples of computational creativity: ILP multiple predicate synthesis and the 'assets' in theorem proving," in J. Koronacki, Z. W. Ras, S.T. Wierzchon and J. Kacprzyk, Eds., Advances in Machine Learning II: Dedicated to the Memory of Professor Ryszard S. Michalski; Springer-Verlag, pp. 155-174, 2010.

[48] M. Franova and M. Kooli, "Recursion manipulation for robotics: why and how?," in R. Trappl, Eds., Cybernetics and Systems '98; proc. of the Fourteenth Meeting on Cybernetics and Systems Research, Austrian Society for Cybernetic Studies, Vienna, Austria, pp. 836-841, 1998.

[49] M. Franova and L. Popelinsky, "Synthesis of formal specifications of predicates: why and how?," in R. Trappl, Eds., Cybernetics and Systems 2000; proc. of the Fifteenth European Meeting on Cybernetics and Systems Research, Volume II, Austrian Society for Cybernetics Studies, pp. 739-744, 2000.

[50] J. Y. Girard, Le Point Aveugle I - Cours de Logique - Vers la Perfection. Hermann, 2006.

[51] K. Gödel, "Some metamathematical results on completeness and consistency, On formally undecidable propositions of Principia Mathematica and related systems I, and On completeness and consistency," in J. van Heijenoort: From Frege to Godel, A source book in mathematical logic, 1879-1931; Harvard University Press, Cambridge, Massachusets, pp. 592-618, 1967.

[52] A. Ireland, "The use of planning critics in mechanizing inductive proofs," in Logic Programming and Automated Reasoning, Lecture Notes in Computer Science, Volume 624, pp. 178-189, 1992.

[53] M. Johansson, L. Dixon, and A. Bundy, "Case-Analysis for rippling and inductive proof," Interactive Theorem Proving, Lecture Notes in Computer Science, Volume 6172, pp. 291-306, 2010.

[54] D. Kapur, "An overview of Rewrite Rule Laboratory (RRL)," J. Comput. Math. Appl. 29(2), pp. 91–114, 1995.

[55] Y. Korukhova, "An approach to automatic deductive synthesis of functional programs," Annals of Mathematics and Artificial Intelligence, Volume 50, Issue 3-4, pp. 255-271, August 2007.

[56] J. L. Le Moigne, La théorie du système général, théorie de la modélisation. P.U.F, 1984.

[57] Z. Manna and R.Waldinger, "A deductive approach to program synthesis," ACM Transactions on Programming Languages and Systems, Volume 2, Issue 1, January, pp. 90-121, 1980.

[58] S. Nedunuri, D.R. Smith, and W.R. Cook, "Theory and techniques for synthesizing efficient breadth-first search algorithms," in FM 2012: Formal Methods, Lecture Notes in Computer Science, Volume 7436, pp. 308-325, 2012.

[59] C. Paulin-Mohring and B. Werner, "Synthesis of ML programs in the system Coq," Journal of Symbolic Computation; Volume 15, Issues 5–6, pp. 607–640, May–June 1993.

[60] L. C. Paulson, "The foundation of a generic theorem prover," Journal of Automated Reasoning, Volume 5, Issue 3, pp. 363-397, September 1989.

[61] B. Pientka and Ch. Kreitz, "Instantiation of existentially quantified variables in inductive specification proofs," Artificial Intelligence and Symbolic Computation, Lecture Notes in Computer Science, Volume 1476, pp. 247-258, 1998.

[62] K. Popper, The logic of scientific discovery. Harper, 1968.

[63] Y. Pu, R. Bodik, and S. Srivastava, "Synthesis of first-order dynamic programming algorithms," in OOPSLA '11: Proceedings of the 2011 ACM international conference on Object oriented programming systems languages and applications, pp. 83-98, 2011.

[64] D. R. Smith, "Top-down synthesis of simple divide and conquer algorithm," Artificial Intelligence, Volume 27, Issue 1, pp. 43-96, 1985.

[65] W. Sonnex, S. Drossopulou, and S. Eisenbach, "Zeno: an automated prover for properties of recursive data structures," in Proceedings of TACAS, Springer, pp. 407-421, 2012.

[66] S. Srivastava, S. Gulwani, and J. S. Foster, "From program verification to program synthesis", in M. V. Hermenegildo and J. Palsberg, Eds., Proceedings of the 37th ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages, POPL 2010, pp. 313-326, 2010.

[67] M. Vechev, E. Yahav, and G. Yorsh, "Abstraction-guided synthesis of synchronization," POPL '10: Proceedings of the 37th annual ACM SIGPLAN-SIGACT Symposium on Principles of programming languages, pp. 327-338, 2010.