A Cartesian Methodology for an Autonomous Program Synthesis System

Marta Franova

A Cartesian Methodology for an Autonomous Program Synthesis System

Marta Franova
LRI, UMR8623 du CNRS & INRIA Saclay
Bât. 660, Orsay, France
mf@lri.fr

***Abstract*— In this paper, we present the main difference between Newtonian and Cartesian approaches to scientific creativity when related to Program Synthesis (PS). The main contribution of the paper is a thorough discussion on the creative building of a theorem prover. 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. This analysis is then applied to the more complex case of the general framework for our own 'Constructive Matching Methodology' (*CMM*) as a Cartesian approach to the creation of an autonomous theorem prover for PS.**

Keywords-program synthesis systems; methodology; Constructive Matching methodology; creativity, symbiosis

I. INTRODUCTION

Automatic construction of programs is obviously a desirable goal. There are two main approaches to tackle with this task, namely inductive and deductive. In this paper, we are interested in the deductive approach to Program Synthesis (PS) introduced by Manna and Waldinger in the eighties [22] and followed by many authors, for instance [4], [5], [8], [10], [13], [21], [23], [26]. This problem is however undecidable as a consequence of Gödel's Theorems [19]. In this paper, we shall present an attempt to, as much as possible, approximate the automatization of the deductive approach to PS by introducing the conceptual switch of 'Cartesian Intuitionism', defined by Franova [16] and informally described in the book Franova [15].

This approach is, from an epistemological point of view, an interesting alternative and a complement to the more formal Newtonian approaches because it enables to handle informal specifications. Nevertheless, it is still too soon to compare these approaches on the basis of their relative performance. From a practical point of view, in building what we call Cartesian Intuitionism, we try to open the way to a creative approach that provides a frame of thought to the user of a theorem prover in the process of recovering from a failure.

Before going into the details of the structure of the paper, let us stress the role of Section IV of this paper. This Section contains an example illustrating and underlining the deep gap between creating a set of axioms, that is to say, Cartesian *creation* of these axioms and making use of a given set of axioms, that is to say Newtonian *construction* of a proof.

In Section IV, we illustrate Cartesian creation of the Newtonian theory of the non negative integers build using Peano's axiom. The idea is that each of its 5 axioms depends on the other ones to be justified. Besides, modifying one axiom modifies the others as we shall then illustrate. Another obvious example of Cartesian creativity, though at a much higher level, is provided by Lobachevski's geometry. Euclidian geometry is a very efficient Newtonian system. It becomes Cartesian when you try to play with the axiom relative to the parallels, where you 'create' universes where parallels in the same plane can cross once, several times, all cross at the same point or not etc. The creative aspects we deal with here are akin to these two examples: Our axiomatic system has to invent new axioms each time it meets a failure.

The paper is structured as follows. In Section II, we recall the formulation of the deductive approach to PS. In Section III, we recall the main features of Newtonian and Cartesian approaches to scientific creativity related to PS. In particular, we shall recall the basic notions of Cartesian intuitionism. Section IV has been already summarized. We shall devote Section V to the description of our Constructive Matching Methodology (*CMM*) in the light of Cartesian Intuitionism. In Section VI, we present a few epistemological remarks.

II. PROGRAM SYNTHESIS – DEFINITION OF THE PROBLEM

By program synthesis we call here the deductive approach to automatic construction of recursive programs introduced by Manna and Waldinger [22]. This approach 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. A proof by recursion of this 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, program synthesis 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. There are two main problems with respect to this role:

1. Treatment of strategic aspects of inductive theorem proving system specialized for specification formulae.

We have illustrated this first problem on a simple specification theorem (a computation of the last element of a list) in [16] and a complex example (synthesis of

 ISBN: 978-1-61208-319-3

Ackermann's function defined with respect to the second argument) is presented in [12].

2. Treatment of strategic aspects of creativity related to the design of such theorem prover.

The present paper is concerned with this second problem, that is, the one of building a system able to perform program synthesis. Strategic aspect can seldom been efficiently formalized. We moreover deal with creativity, the formal aspects of which start being explored. It is obviously too soon to present a general (Newtonian) theory of (Cartesian) creativity in the usual style of lemmas, theorems etc.; thus the nature of this paper is more epistemological than axiomatic.

III. NEWTONIAN AND CARTESIAN APPROACHES TO PS

In the previous Section, we have mentioned that we are here interested in the creative process of construction of an inductive theorem prover. This prover has to be specialized in specification formulae. There are two main styles in this creative process. For particular reasons presented in [16], we call 'Newtonian' the standard approach and 'Cartesian' the non-classical one. In [16], we have presented in detail the above two styles. In this Section, we shall recall the main features necessary for understanding the present communication.

A. Newtonian Approach

The specialists in the Newtonian approach to PS build their own theorem prover, one based on a logic of sequential research. Classically, the reference system of any theorem proving system consists in a set of axioms, rules of inference and control mechanisms devoted to finding a recursive proof for the specification formula. In the Newtonian approach, the various blocks composing the reference system are a composition of some tools chosen among the whole set of all existing tools. In the case where an author introduces a new block he/she invented him/herself, then this new block must be coherent with the existing tools. In other words, more formally, Newtonian approach considers creativity as a finite linear sequence:

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

In this sense, Newtonian creativity is similar to essentialism within the frame of logics as defined by J.Y. Girard in [18].

Since this approach is based on standard mathematical knowledge, it inevitably inherits the negative results of Kurt Gödel [19]. The results of Gödel are said to be negative because they show that the objective of PS, as it is formulated in *beginning*, cannot lead to a successful end of the task in the classical framework. This happens because the classical approach focalizes on the problem

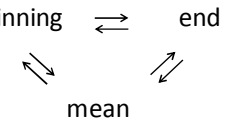
∃ formal framework
in which ∀ specification formula has a solution.

Gödel's results show the impossibility to define a formal logical framework, containing the natural numbers, allowing to deal with the automated resolution (confirm or counter) of specifications given in a general way. This is why Newtonian approaches react by placing themselves inside user-dependent theorem proving assistants, such as ACL2 [6], the system RRL [20], the system NuPRL [9], the Oyster-Clam system [7], the extensions of ISABELLE [24], the system COQ [3], Matita Proof Assistant [1] and Otter-Lambda [2].

B. Cartesian Approach

We have developed an alternative to this classical approach by taking into account the creativity necessary for designing a PS system from the point of view of what we call Cartesian Intuitionism [16]. This non-classical creativity

(a) focalises on the problem: {∀ specification formula ∃ formal 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 [18].

Points (a), (b), and (c) together mean that Cartesian approach to PS is based on a logic of recursive science where the reference system of the problem and the milestones of construction of the solution (i.e., the definitions and the rules of inference of a given specification formula) are formulated hand in hand with the development of the solution. Moreover, the exact demarcation of the reference system and the milestones of construction is the final stage of the process, and it is also a part of the solution. It follows that Cartesian approach specifies in an informal way the purpose to be reached, by a necessarily informal formulation of the reference system. For instance, *CMM* specifies the purpose of PS informally by the sentence: "Create a custom-made mechanism for proving specification formulae that automates PS as much as possible." We shall say more about our application of Cartesian approach to PS later in Section V, because we need first to clarify the model of creation for *CMM* in the following Section.

IV. NEWTONIAN CONSTRUCTION VERSUS CARTESIAN CREATION

In this Section, 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 and the creation of Peano's axioms. Peano's axioms define the arithmetic properties of natural numbers N. These axioms include a constant symbol

 ISBN: 978-1-61208-319-3

0 and unary function symbol S. These axioms are usually used to build formal proofs about natural numbers. Our presentation does not deal with this topic, but with the one of reasoning about the construction of these 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.

We shall tackle here, in this Section, with the difference between the use and the creation of these five axioms. To this purpose, we need to precisely specify the difference between synergy and symbiosis.

A. Synergistical Construction

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**,** that is all the tools that have been developed in all scientific domains beforehand, for various purposes. These tools are not built in such a way that one calls another to solve one of its problems before this one has finished 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 independently of each other for the construction of other objects. During the construction process they do not lose their properties.

B. Symbiotical Construction

In contrast to this, an object is constructed *symbiotically* when its parts, maybe seemingly independent, have, during the construction process, no meaning as isolated entities and a slight change of one part influences the others and the whole as we illustrate later.

The main point we want to 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), which deals with 0 and N. This first axiom, however, does not say what is the full meaning neither of 0 nor of N. 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, apart 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" of what 0 and N mean for him in this first axiom. In other words, writing this first axiom, the axiom's creator intuitively knows what 0 and N will be once their description will be 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. We shall present later an example illustrating this symbiotic character; but we now need at first to introduce some more notions.

C. Cartesian Creation

N is constructed with the help of three "elements", namely 0, S and N itself. Note that N self-reference is already acknowledged as a constructive recursive 'trick'. 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). In the following, instead of 'construction' we shall call this process 'Cartesian creation' in tribute to Descartes' §62 of *The Principles of philosophy* [11]. We shall use the following notation:

$$< A + B > = C,$$

where A, B are constructors and C is the created "whole" for this kind of a symbiotic Cartesian creation. This defines N in the following way:

$$< 0 + S + N > = N.$$

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, that 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 altered the properties of all the constructors, including N which changed into N3. This is not the case in a synergic 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.

 ISBN: 978-1-61208-319-3

We just have shown that this 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.

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 result of the creation, not on their creative process itself. Researchers seem to prefer tool-box Newtonian progressive construction which provides the security of familiarity with such linear or modular processes. This may also explain why our original Cartesian approach is not used in the research on Program Synthesis.

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, as we shall show in the next Section. 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. These new problems ask for a new search for 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. *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 upon which the system is built. 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(0)))), the 'synthesis' will fail. In a Newtonian approach, the user would be asked for a lemma specific to S(S(S(0)))) that enables a success. In such a case our approach 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 and would rather suggest to enlarge the solution to introducing N itself. See below a modification that is less easy to find.

Let us now provide a more complex example that illustrates a situation where modifying system of axioms defining PS mechanism is not trivial.

Newtonian system called Otter-Lambda is presented by Beeson [2], together with several examples of its execution. We have chosen among them a formula

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

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

(1) $\exp(u,0) = s(0)$

(2) $\exp(u,S(v)) = u*\exp(u,v)$

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

(3) $0 + u = u$

(4) $S(v) + u = S(v + u)$

(5) $0 * u = 0$

(6) $S(v) * u = (v * u) + u$

The definition of < is also recursive and given as:

(7) $0 < y$, if $y \neq 0$

(8) $S(v) < y$, if $v < y$ & $y \neq S(v)$

Since the Otter-Lambda system fails, it requests some help from its human user. In [2], the user is able to provide the following lemmas that enable Otter-Lambda to complete the proof of (*).

(9) $\text{not}(u<v)$ or $(x*u < x*v)$ or $\text{not}(0 < x)$

(10) $(x < y)$ or $(y \leq x)$

(11) $\text{not}(y \leq x)$ or $\text{not}(x < y)$

(12) $\text{not}(u < v)$ or not $(v \leq w)$ or $(u < v)$

(13) $\text{not}(S(0) < z)$ or $\text{not}(0 < y)$ or $(S(y) \leq z*y)$

(14) $0 + x = x$

We applied our Cartesian approach to the same problem, which does not suggest to get 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 system notices at once a probable source of trouble: the predicate < is recursively defined on its first argument, while, in (*), the induction variable n occurs also in second position of the predicate <. At this stage, the system could suggest the user to provide a definition of < with respect to both argument (this would actually fail), or to the second argument (this would fail as well), or else, a non recursive definition (that would succeed). As already claimed, our system does not call on its user, and it will proceed by calling a custom-designed constructor module we named "Synthesis of Formal Specifications of Predicates" described by Franova and Popelinsky [17]. The symbiotic

 ISBN: 978-1-61208-319-3

system *CMM* with this constructor module included generates the following formal specification for predicate <:

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

With this new definition (*) is transformed into

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

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:

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* which is ready, when necessary, to be once again modified.

The basic constructor of *CMM* is presented in [16] and the other constructors of *CMM* specified so far are described in our publications up to 2001. Some of these constructors were implemented in the system *Proofs Educed by Constructive Matching for Synthesis* (PRECOMAS) [14].

VI. A FEW EPISTEMOLOGICAL REMARKS

Accepting to use Cartesian Intuitionism as a way of creation of some complex systems (we exemplified here a Program Synthesis system) requires a deep transformation of our attitude together with an inevitable shift in thinking, because of changes, due to the new context, in vocabulary meaning, resonances and connotations. Newtonian theories and systems provide a kind of comfortable environment by the identified boundaries existing between each component of their architecture. Therefore, it is true that losing this comfort by accessing the new context we define here requires from the scientists a large change in their behaviour. In this, a Cartesian system requires from researchers the acceptance of open-ended research with its conceptual switches and a new propensity to deal with completeness and incompleteness. In a sense, such an open-ended 'technological' approach seems to be a natural answer to the open-ended theory of natural numbers and the open-ended 'bunch' of desires expressed as program synthesis problems.

Until now, the main technique used in the direction of such an opening to intuition has been carried out by the brainstorming techniques, in which several subjects relax enough to build unexpected mind connexions that might bring a new idea to the fore. In a sense, brainstorming could be an ideal way to define as precisely as possible what is the starting, informal, specification of the problem. The following steps of our proposal are still based on something similar to brainstorming, but the mind of each subject has to focus on ideas explicitly related to the informal specification of the problem. Ideas to find a path from informal to formal specification, then to solution, are triggered by each new problem arising at each failure to succeed in proving a step towards solution. In that sense, the collaboration between the members of a team working on the problem at hand, is enriched and much more focused by this problem than it is during a brainstorming session.

VII. CONCLUSION

Any design of a new complex system obviously requires, during its creative process**,** that its authors might be able to generate new ideas. In the field of program synthesis, our approach can be looked upon as a 'generator of new ideas'. We thus somewhat try to contradict Karl Popper who claims in [25] that "there is no such a thing as a logical method of having new ideas, or a logical reconstruction of this process." Our opinion is that Popper restricts here logical thinking to the linear one and his claim is perhaps valid in such a framework. On the contrary, our experience shows that Cartesian Intuitionism with its recursive features provides a method for having new ideas (and ones that are 'useful-for-solving-the-problem-at-hand)' as well as a model for a reconstruction of creative process, as we illustrated it in the study of creation of the Peano's axioms and its application to the design of an autonomous PS system.

By this paper, we have progressed in the direction of an adequate formalization of the first fundamental challenge met, as pointed out in [16], in the oscillatory design of the recursive system, namely, the challenge of understanding the symbiotic interrelation between a recursive whole, like N or *CMM*, and its parts (constructors) like S from N or "Synthesis of Formal Specifications of Predicates" from *CMM*. Understanding this first challenge will help to accelerate our future work on the three remaining problems described in [16], namely the 'chameleon' like behaviour of Cartesian systems, which are simultaneously static/dynamic, finite/infinite and complete/incomplete.

The ideas explained in the present paper are an illustration of our methodology that we plan to enlarge to problem-solving in general, not only to program synthesis.

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 referees' comments have been very helpful in improving our presentation.

REFERENCES

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

[2] M. Beeson, “Mathematical Induction in Otter-Lambda,” Journal of Automated Reasoning, April 2006, vol. 36, Issue 4, pp. 311-344.

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

[4] W. Bibel, “On Syntax-Directed, Semantic-Supported Program Synthesis,” Artificial Intelligence 14, 1980, pp. 243-261.

[5] S. Biundo and F. Zboray, “Automated Induction Proofs using methods of program synthesis,” Computers and Artificial Intelligence, 3, No. 6, 1984, pp. 473-481.

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

[7] A. Bundy, F. Van Harnelen, C. Horn, and A. Smaill, “The Oyster–Clam System,” in M.E. Stickel, (ed.) 10th International Conference on Automated Deduction, vol. 449 of Lecture Notes in Artificial Intelligence, Springer 1990, pp. 647–648.

[8] J. Chazarain and S. Muller, “Automated Synthesis of Recursive Programs from a ‘forall’ ‘exists’ Logical Specification,” Journal of Automated Reasoning, October 1998, vol. 21, Issue 2, pp. 233-275.

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

[10] N. Dershowitz and U.S. Reddy, “Deductive and Inductive Synthesis of Equational Programs,” JSC vol. 15, Nos. 5 and 6, 1993, pp. 463-466.

[11] R. Descartes “Œuvres philosophiques” (3 vol.). Edition de F.Alquié. T. 3; Classiques Garnier, Bordas, 1988.

[12] M. Franova, “A construction of a definition recursive with respect to the second variable for the Ackermann’s function”, Research Report No.1511, L.R.I., 2009.

[13] M. Franova, “Program Synthesis and Constructive proofs Obtained by Beth's tableaux,” in R. Trappl, (ed), Cybernetics and System Research vol. 2, North-Holland, Amsterdam, 1984, pp. 715-720.

[14] M. Franova, “PRECOMAS - An Implementation of Constructive Matching Methodology,” Proceedings of ISSAC'90, ACM, New York, 1990, pp. 16-23.

[15] M. Franova, Créativité Formelle: Méthode et Pratique - Conception des systèmes ‘informatiques’ complexes et Brevet Épistémologique , (Formal Creativity: Method and Pratics – Conception of Complex ‘Informatics’ Systems and Epistemological Patent) Publibook, 2008.

[16] M. Franova, “Cartesian Intuitionism for Program Synthesis,” in S. Shimizu, T. Bosomaier (eds.) , Cognitive 2013, The Fifth International Conference on Advanced Cognitive Technologies and Applications, www.thinkmind.org, ISBN : 978-1-61208-273-8, 2013, pp. 102-107.

[17] M. Franova and L. Popelinsky, “Synthesis of formal specifications of predicates: Why and How?,” in R. Trappl, (ed.), Cybernetics and Systems 2000, proc. of the Fifteenth European Meeting on Cybernetics and Systems Research, vol. 1, vol. 2, Austrian Society for Cybernetics Studies, 2000, pp. 739-744.

[18] J. Y. Girard, Le Point Aveugle I - Cours de Logique - Vers la Perfection, (The Blind Spot I – The Course on Logics – Towards the Perfection) Hermann, 2006.

[19] 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, 1967, pp. 592-618.

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

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

[22] Z. Manna and R.Waldinger, “A Deductive Approach to Program Synthesis,” ACM Transactions on Programming Languages and Systems, vol. 2., No.1, January, 1980, pp. 90-121.

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

[24] L. C. Paulson, “The foundation of a generic theorem prover,” Journal of Automated Reasoning, September 1989, vol. 5, Issue 3, pp. 363-397.

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

[26] D. R. Smith, “Top-Down Synthesis of Simple Divide and Conquer Algorithm,” Artificial Intelligence, vol. 27, no. 1, 1985, pp. 43-96.

 ISBN: 978-1-61208-319-3