Cartesian Intuitionism for Program Synthesis

Marta Franova

Cartesian Intuitionism for Program Synthesis

Marta Franova
Laboratoire de Recherche en Informatique, UMR8623 du CNRS
Orsay, France
mf@lri.fr

***Abstract*—This paper presents two possible approaches to scientific discovery, the Newtonian and the Cartesian one. The paper explains the main difference between these two styles and underlines the importance of each of them. Its specific scientific contribution is presenting the less known Cartesian approach and the main problems that can be solved by it, in the light of our research in Automated Program Synthesis. The paper is thus related to the creative framework of modeling human reasoning mechanisms, cognitive and computational models, as well as modeling brain information processing mechanisms.**

Keywords-creativity; Program Synthesis; Newtonian style; Cartesian style; Constructive Matching Methodology

I. INTRODUCTION

This paper suggests a model of human creativity, inspired by our experience on Program Synthesis. In the following sections we shall describe a way of transforming a formal specification, and possibly a partially incomplete one, into a program computing in accordance with this specification. Our purpose is to show that the solution we propose to this problem strongly suggests a model for mathematical creativity.

A. Newtonian Creativity

Newtonian creativity (represented in program synthesis mainly by ([25], [28], [4], [3], [10]) consists essentially in the choice of a statement of the problem so that it can be handled linearly; the problems met have to be solved at once when met, independently of the other problems that are going to be found 'later'. That does not go without creativity, obviously needed to solve these problems. Thus, the creativity process in Newtonian creativity is considered as a cycle of five distinct stages where every stage must be completed before envisaging the next stage. These stages are the following ones.

1. The first one is that of the preparation, where all the important information is collected to correctly express the problem,
2. the one of incubation that supposes the contribution of the whole personality of the mathematician,
3. the one of discovery where an inspiration occurs and finally
4. the stage of verification.

In the case of failure, the creator is forced to undertake

5. the fifth stage which we can qualify as 'incubation of the failure'.

When the failure has been 'incubated' enough, the creator has to take up again the first four stages. This characterizes a behavior where learning from failures is made only after the observation of the global failure of the process of creation. Stated in more formal way, the Newtonian approach hypothesizes that we believe in the following rigid, 'unfriendly' universe: There exists a universal theory, in which the tools necessary to solve **all** problems are available. This is logically expressed as follows:

[The following problem has a solution: {∃ formal framework (a theory) ∀ problem}]

Failure to find a solution is answered by looking for a new universal theory that will provide the appropriate tools. The reader will recognize here the classical behavior of Physics as a Science.

B. Cartesian Creativity

Cartesian creativity (represented in program synthesis mainly by [14]) consists essentially in a 'less conquering' or more 'cautious' approach in which the five Newtonian stages are not so clearly distinguished. For the sake of simplicity let us distinguish phases that run in parallel without, at first, emphasizing their mutual dependencies.

The first phase is the phase of an informal familiarization with the problem which results in a formulation of an ***informal specification*** of the problem that may seem, and often is, absurd or unfeasible to solve in the standard context.

The second phase is the one where, given an informal specification, we try to estimate what we can possibly attempt in order to solve the problem. It consists in performing two actions, specific for the given problem. The first action is gathering the tools that might be usable in *handling* the problem. These tools will be quite general since we do not restrict ourselves to the tools usable in *solving* the problem. For example, in the example that follows, the tool we chose is the one of mathematical induction and various techniques, such as replacing a term by a parameter. The second action is a search for a set of minimal restrictions that are necessary for the problem to be solved. In other word, we try to define the largest possible context in which we can hope to solve the problem. This is symmetrically defined as the smallest context in which we know that no solution will be possible.

The third phase starts once these choices have been done. They are applied to the informal specification, until we succeed into obtaining a specification in which no contradiction is still observable within the context of the problem, as it has been defined during the second phase. This result is called a ***coherent (or 'reasonable') specification.***

 ISBN: 978-1-61208-273-8

The fourth phase consists in a sequence of attempts at proving the validity of the coherent specification. If this succeeds, then this proof is also a solution to the problem. Failures at obtaining of proof does not lead us to reformulate the problem, or to choose a new theory, as happens in the Newtonian approach. A failure does not lead us further than modifying the coherent specification.

Stated in more formal way, the Cartesian approach hypothesizes that we believe in the following flexible, 'friendlier' universe: Given a problem, we will be able to find or build a theory dedicated to solving the **specific** problem at hand. Failure to find a solution will be answered by building a theory in which a solution is possible (instead of looking for a new existing theory). This is logically expressed as follows:

[The following problem has a solution: {∀ problems ∃ appropriate framework (a theory)}].

The reader will recognize here the classical behavior of an artist in front of a task to execute (the specific specification) that he will fulfill either by using existing tools (the existing 'universal' theory of Art, the so-called "academic artists") or by creating a theory specific to his problems (the so-called "new Art").

These two types of creativity cannot, however, in isolation render the richness of mathematician thought. We do not yet have the weapons necessary to describe how may happen a 'pulsation' between the two above approaches. We are nevertheless able to provide a formal specification for it. A formal specification of the pulsating between Newtonian and Cartesian approaches is represented by inversing the ordering of the two quantified terms 'theory' and 'problem'. Newtonian states that ∃ a (universal) theory usable ∀ problems, while Cartesian states that ∀ problems that ∃ a (particular) theory.

The following of this paper will explain how the problem of program synthesis from specification demands such a pulsative approach in order to find a solution. Note that **automatic** program synthesis is difficult enough to be still unable to be applicable to real world problems while the humans called 'programmers' seem to be able to often provide solutions often satisfactory.

The rest of the paper is structured as follows. In section II we shall formalize a bit more the Newtonian and the Cartesian styles of the research. In section III we recall the goal of Program Synthesis and we shall relate the main features of the Newtonian and the Cartesian styles to program synthesis. In section IV we shall present what we call conceptual oscillation in our approach. Sections V and VI are devoted to the main perspectives of these approaches. Section VII recalls the main building strategy of our approach and illustrates it on a simple example.

II. NEWTONIAN AND CARTESIAN STYLES OF RESEARCH

The main difference between these two approaches 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."

Descartes wrote his first rule in the *Discourse on the Method of Rightly Conducting the Reason, and Seeking Truth in the Sciences* in the 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."

Newtonian science is thus established on a logic of sequential research, where the reference system of the problem, that is, the axioms, the rules of inference and the mechanism of control of the system intended to solve the problem, and the milestones (that is, the definitions and the rules of inference of the specified concrete problem) on which we build the solution are given at the beginning by the past history of scientific research. It is in this perspective of work that are situated the results of Gödel of which we will speak later.

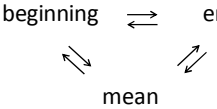
Descartes speaks about the obvious truth. As says the commentator of Descartes Ferdinand Alquié ([11], p. 586), the act of thought which seizes the obvious truth is the intuition defined by Descartes in his *Rules for the direction of the mind*. So, the study of Descartes intuition, as presented in the book *Formal Creativity* [17] enables to notice that Cartesian science is based on a logic of recursive research, where the reference system of the problem and the milestones of construction of the solution are formulated hand in hand with the development of the solution, and where the exact demarcation of the reference system and the milestones of construction is the final stage of the process, and is too a part of the solution. The Cartesian approach thus takes into account that the demarcation of a notion is not the initial stage but the final stage of its formation.

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 which follow later; and that these things which follow cannot be understood well, if we do not remember all those that precede them." [11], p. 797.

In a little more formalized way, we can thus describe the Newtonian way by the sequence

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

The Cartesian way can be described by the loop



where the arrow → means "leads to". Because of the complexity of the Cartesian way and since neither the external observation nor the sequential transmission are suited to the appreciation of the work made in this way, this way presents more obstacles than the Newtonian style.

III. PROGRAM SYNTHESIS AND APPROACHES

By program synthesis we call here the deductive approach to automatic construction of recursive programs introduced in [25]. 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) contains no universal quantifiers 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. 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.

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

A. Newtonian approach to Program Synthesis

The Newtonian approach 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 his article *On formally undecidable propositions of Principia Mathematica and related systems I* [23], 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 that starts by a well-defined beginning and progresses by advancements to a desired end.

The results of Gödel are said negative because they show that the objective 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 that work in the Newtonian style.

In the introduction we have mentioned the most known Newtonian approaches to program synthesis. 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 [5], the system RRL [24], the system NuPRL [8], the Oyster-Clam system [6], the extensions of ISABELLE [27], the system COQ [26], Matita Proof Assistant [1] and Otter-Lambda [2]. 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 program synthesis. The construction calculus of [9], that is the basis of the system COQ, is a constructive way of *representing* transformational methods. The approach presented in the next section attempts to find a *constructive* way of solving an 'almost' same problem by modelling human reasoning based on Cartesian style of research.

B. Cartesian approach to Program Synthesis

The Cartesian approach specifies at the beginning the reference system in an informal way only, by a necessarily informal formulation of the purpose to be reached. It is much like a hypothetico-deductive method. The hypothetico-deductive method is a procedure of construction of a theory which consists in putting, at the start, a certain number of loosely defined concepts or proposals that are obtained by an analysis 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 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 real well-known situation.

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

- rigor, 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 general sense) rather than a theory.

IV. CONCEPTUAL OSCILLATION OF *CMM*

Our approach oscillates between a Newtonian formulation of Program Synthesis and a Cartesian formulation of the same problem. It is clear that this purpose seems thus very ambitious when one forgets the preliminary restrictions (not considering efficiency of synthesised programs and proofs by structural induction only).

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 '$\exists$solution $\forall$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 putting focus on the problem '∀problem ∃solution' and then by a suitable process of conceptualization based on 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 approach 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 [13] and the most complete presentation of which can be found in [16]. In contrast to the basic methods in Newtonian approaches that rely on simplification and rewriting, our *CM-formula construction* is a constructive method and thus it is very suitable for generating missing lemmas and even axioms when the given data are incomplete as it is illustrated in [22].

V. NEWTONIAN AND CARTESIAN PERSPECTIVES

In many cases, including Program Construction, researchers and engineers look at their problems in a goal driven perspective, that is, they try to select beforehand axioms useful for obtaining a particular proof. This approach becomes however less and less successful when it is applied to goals of increasing complexity requiring less and less trivial lemmas, especially when it becomes necessary to simultaneously take into account all the axioms, together with the set of their consequences. In the specific case of Program Synthesis, we have seen, in section III, that the Newtonian approach has been very successful in producing systems that request human help as soon as some 'creativity' is needed in order to provide a lemma or a heuristic not already included in the system library. Since one of ultimate goals is modeling some form of mathematical creativity by building a computer simulation of these creative steps, we had to adopt a new perspective, the one of Cartesian intuitionism.

When non-trivial lemmas are generated in an autonomous way by the computer system itself, and when it is required that we simultaneously take into account axioms and the set of their consequences, our *CMM* fits into Cartesian intuitionism as Descartes himself specified it by defining:

- a form of constructive intuition, in the Latin version of his *Rules for the direction of the mind*
- the ability of thinking as isolated, one of many mutually dependant features in §62 of *The principles of the philosophy* ;
- clear and distinct perception in §45 and §46 of *The principles of the philosophy*;
- the four rules of his method, in his *Discourse on the method*.

Thus, the research program of *CMM* approaches the construction of axioms and intermediate lemmas and the theorem proving system in dependence on the specifications of program synthesis. It is important to note that these three stages (generating missing axioms – in case of undecidability, generating missing lemmas, developing the procedure of demonstration or of control) are interdependent and that the advances of one of three stages can modify the internal objective of the two others.

VI. ASSESSMENT AND PERSPECTIVES OF *CMM*

The stage relative to the procedure of demonstration was elaborated in all our publications until 2000 [12]. An experimental system called Proofs Educed by Constructive Matching for Synthesis (PRECOMAS) showing the well-founded character of the *CM-formula construction* that is the basis for *CMM* was developed in the 90s [15].

The stage relative to the specification of the intermediate lemmas advances well and concerns also the scientific domain known as 'computational creativity' [20], [21].

The stage which concerns the clear and distinct perception (in the Cartesian sense) of the targeted strategic recursive axiomatization has begun in the article [19]. It must be improved and pursued by an adequate formalization of different fundamental interrelated problems which 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. 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" 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 a formalized shape and an informal shape of a given mechanism. As we said above, the definitive demarcation which consists of fixing a final version of the mechanism is only made at the end of development of the whole system.

The problem of the regulation of the finite and the infinite appears in program synthesis 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 Program Synthesis 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 which we envisage for this problem.

 ISBN: 978-1-61208-273-8

The problem of the oscillation between completeness and incompleteness is described in an informal way by the notion of pulsation which 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.

The power of *CMM* was illustrated on a number of interesting problems such as n-queens, the quotient and the rest of two numbers, a problem in robotics and more recently the construction of a definition of Ackermann's function with respect to the second variable [18]. This last illustration is important because it shows the capacity of the system to find another form of defining axioms, the final version of which is not known beforehand.

CMM is even suitable for proving some purely universally quantified theorems. The advantage lies in the fact that, in contrast for instance to [7], 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 inductive theorem proving and thus it seems to be conceptually more natural.

In the following section we shall recall our *CM-formula construction* and we shall give a very simple example to illustrate it.

VII. *CM-FORMULA CONSTRUCTION*

A. 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 predicates a feature of which has to be proven true, we call **abstract arguments**. They are denoted by ξ (or ξ' etc.) in the following. The abstract argument replaces, in a purely syntactical way, one of the arguments of the given formula. The first step is choosing which of the arguments will be replaced by an abstract argument, ξ. Thus, the value of this argument is looked upon as being known and, in a usual proof, its characteristics are used in order to prove the given formula. In our approach, we 'forget' for some time these characteristics and we concentrate on studying the features ξ should have so as insuring that the theorem with a substituted argument is true.

Suppose that we have chosen to work with $F(t_1,\xi)$. We shall then look for the features shown by all the ξ such that $F(t_1,\xi)$ is true. Given axioms defining F and the functions occurring in t_1, we are able to obtain a set C expressing the conditions on the set $\{ \xi \}$ for which $F(t_1,\xi)$ is true. In other words, calling 'cond' these conditions and C the set of the ξ such that $cond(\xi)$ is true, we define C by $C = \{\xi \mid cond(\xi)\}$. We can also say that, with the help of the given axioms, we build a 'cond' such that the formula: $\forall \xi \in C$, $F(t_1,\xi)$ is true. 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'. Using the characteristics of C and the definition axioms in order to perform evaluations, and also using the induction hypothesis, we shall build a form of ξ such that $F(t_1,\xi)$. Even though it is still 'ξ' and only for the sake of clarity, let us call ξ_C an axiom evaluated form to which, possibly, the induction hypothesis has been applied. It is thus such that $F(t_1,\xi_C)$ is true. We are still left with a hard work to do: modify ξ_C in such a way that ξ_C and t_2 will be made identical, which finally completes the proof. [16] gives a detailed description of handling the abstract argument in a rigorous framework.

B. Example

In this section we shall give a very simple illustration of the *CM-formula construction*. The goal will be to synthesize a recursive program for computing the last element of a non-empty list. The formal specification $\forall x \exists z1 \exists z2\ F(x,z1,z2)$ for this problem writes

$$\forall x \exists z1 \exists z2 \{ x \neq nil \Rightarrow x = app(z1,cons(z2,nil)) \}. \quad (1)$$

We shall name sf1 the Skolem function for z1 and sf2 the Skolem function for z2.The definition for the function append 'app' writes

$$app(nil,v) = v. \quad (2)$$

$$app(cons(c,u),v) = cons(c,app(u,v)). \quad (3)$$

With respect to the input condition $x \neq nil$ the structural induction principle means to prove in the base step $\exists z1 \exists z2$ $cons(a,nil) = app(z1,cons(z2,nil))$, where a is an arbitrary constant. The induction step means to represent x in the form $x = cons(b,l)$, where $l \neq nil$, to suppose the induction hypothesis $\exists e1 \exists e2\ l = app(e1,cons(e2,nil))$ and prove $cons(b,l) = app(z1,cons(z2,nil))$. In this induction hypothesis e1 is sf1(l) and e2 is sf2(l).

The solution of the base step: Since the right hand side of the equation in (1) contains the existentially quantified variables, this side is replaced by the abstract argument ξ. We thus have $C = \{\xi \mid cond(\xi)\} = \{\xi \mid cons(a,nil) = \xi\}$. The goal is now to make ξ and $app(z1,cons(z2,nil))$ identical. In this case, using (2), the *CM-term transformer* presented in [14] returns z1 = nil and z2 = a. The base step is thus solved.

Let us consider the induction step. In this step, we have $C = \{\xi \mid cond(\xi)\} = \{\xi \mid cons(b,l) = \xi\}$. By applying the induction hypothesis to cons(b,l) we obtain $\{\xi_C \mid cons(b,app(e1,cons(e2,nil))) = \xi_C\}$. The goal now is to transform ξ_C i.e., $cons(b,app(e1,cons(e2,nil)))$ into the form $app(z1,cons(z2,nil))$. Using (3), the *CM-term transformer* returns z1 = cons(b,e1), z2 = e2. The proof is thus performed and the program for sf1 and sf2 is trivially extracted.

$$sf1(cons(a,nil)) = nil. \quad (4)$$

$$sf1(cons(b,l)) = cons(b,sf1(l)). \quad (5)$$

$$sf2(cons(a,nil)) = a. \quad (6)$$

$$sf2(cons(b,l)) = sf2(l). \quad (7)$$

 ISBN: 978-1-61208-273-8

This example is interesting not only as a very simple illustration of *CM-formula construction*, but also as a suggestion to use program synthesis as a powerful unification tool.

VIII. CONCLUSION

We have been able to express formally the difference between strictly scientific approach (Newtonian) and an 'artistic' one (Cartesian) closer to what is generally understood by creativity (scientific, artistic, etc.). The purely scientific approach expresses the work to perform as ∃ theory ∀ problems. The 'artistic' approach expresses the work to perform as ∀ problems ∃ theory.

This is a new result about the difference between Science and Art. Scientific mind will stick as far as possible to the Newtonian approach and resorts to the Cartesian one in cases of failures only. Inversely, the creative artist sticks to the Cartesian one and the academic one to the Newtonian.

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.

The structure of this paper was improved thanks to the comments of anonymous referees.

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, Volume 39, Issue 2, pp 109-139.

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

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

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

[5] R. S. Boyer and J S. Moore, "A computational logic handbook"; Academic Press, Inc., 1988.

[6] A. Bundy, F. Van Harnelen, C. Horn, and A. Smaill, "The Oyster–Clam system", In Proc. 10th International Conference on Automated Deduction, vol. 449 of LNAI, Springer,1990, pp. 647–648.

[7] A. Bundy, D. Basin, D. Hutter, and A. Ireland, "Rippling: Meta-level guidance for mathematical reasoning", Cambridge University Press, 2005.

[8] R. L. Constable, Implementing Mathematics with the Nuprl Proof Development System, Prentice-Hall, 1986.

[9] T. Coquand, G. Huet, "Constructions: A higher order proof system for mechanizing mathematics", in Proc. EUROCAL'85, Vol. 1, Springer-Verlag, 1985, pp. 151-185.

[10] N. Dershowitz, U.S. Reddy, "Deductive and inductive synthesis of equational programs"; JSC Vol. 15, Nos. 5 and 6, 1993, 463-466.

[11] R. Descartes, "Oeuvres philosophiques" (3 vol.). Edition de F. Alquié. T. 1; Classiques Garnier, Bordas, 1988.

[12] M. Franova, www.lri.fr/~mf/mf.publications.html

[13] M. Franova, "CM-strategy : A methodology for inductive theorem proving or constructive well-generalized proofs"; in Proc. of the Ninth IJCAI; 1985, pp. 1214-1220.

[14] 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.

[15] M. Franova, "Precomas 0.3 user guide", Research Report No.524, L.R.I., October, 1989.

[16] M. Franova, "Constructive Matching methodology and automatic plan-construction revisited", Research Report No.874, L.R.I., 1993.

[17] M. Franova, "Créativité Formelle: méthode et pratique - conception des systèmes 'informatiques' complexes et brevet épistémologique", Publibook, 2008.

[18] 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.

[19] M. Franova, "The role of recursion in and for scientific creativity", in Cybernetics and Systems 2010, Proc. of the 20th EMCSR, 2010, pp. 573-578.

[20] M. Franova and Kodratoff Y., "On computational creativity: 'inventing' theorem proofs", in Proc. 18th ISMIS, LNAI 5722, Springer, 2009, pp.573-581.

[21] 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, J. Kacprzyk, (eds.) Advances in Machine Learning II: Dedicated to the Memory of Professor Ryszard S. Michalski, Springer-Verlag, 2010, pp. 155-174.

[22] M. Franova, Kooli M., "Recursion manipulation for robotics: Why and how?", in: Proc. of the Fourteenth Meeting on Cybernetics and Systems Research, Austrian Society for Cybernetic Studies, Austria, 1998, pp. 836-841.

[23] 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 From Frege to Godel, A source book in mathematical logic, 1879-1931, Harvard University Press, 1967, pp. 592-618.

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

[25] Z. Manna, R.Waldinger, "A deductive approach to program synthesis", ACM Transactions on Programming Languages and Systems, Vol. 2., No.1, January, 1980, pp. 90-121.

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

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

[28] D. R. Smith, "Top-down synthesis of simple divide and conquer Algorithm", Artificial Intelligence, vol. 27, no. 1, 1985, pp. 43-96.

 ISBN: 978-1-61208-273-8