Cartesian Handling Informal Specifications in Incomplete Frameworks

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

Yves Kodratoff
LRI, UMR8623 du CNRS & INRIA Saclay
Bât. 660, Orsay, France
email: yvkod@gmail.com

Abstract— **This paper introduces and illustrates a fundamental notion, namely *informal specification*, for creating tools developed as symbiotic recursive pulsating systems (SRPS), in the framework of Inductive Theorem Proving and Intelligent Systems. It illustrates the use of this fundamental notion in scientific systemic creativity relative to theorem proving. We deal simultaneously with the meta-level design of a system that proves theorems automatically.**

Keywords-informal specification; intelligence by design; inductive theorem proving; Cartesian Intuitionism; symbiotic recursive systems; Constructive Matching Methodology.

I. Introduction

Often, a direct way to achieve a proof requiring the use of the induction principle is not obvious. It might even be impossible to prove a formula within a given framework while an appropriate detour or a switch in interpretation and in the method of thinking may lead to a success. This problem of changing the framework is also relevant to the design of an intelligent inductive theorem proving system.

There is a largely adopted management approach based on so-called “SMART goals,” where ‘S’ in SMART stands for ‘specific’ and it implicitly means that there is a kind of formal framework and a reproducible or nearly obvious available know-how to reach such a goal. ‘T’ stands for time-bounded and it means that there is a limit date before which the result is expected. ‘M’ stands for measurable, ‘A’ stands for achievable, ‘R’ stands for realistic.

We shall however see that a systematic use of SMART goals should not always be the way by which invention comes to life. Symbiotic recursive pulsating systems (SRPS) and their corresponding Cartesian paradigm [14] defy this SMART approach, though it is established and today more or less required in Science. Handling or even creating SRPS requires a detour from purely formal thinking expected by exact sciences, a switch from linear synergic interpretations advocated by analysis and synthesis, a switch from observations of ‘pure’ facts to *creative interpretations*, a switch from use of established know-how to *creating* on-purpose know-how. Handling or even creating SRPS require their own method of thinking that is not as much reproducible as the usual trend of sciences would require. From an immediate-gratification perspective this might lead to the option that the best way to handle them is to ignore them. Hopefully, the rich potential of these systems illustrated in this paper might suggest to adopt a more positive attitude towards these systems.

The problem lies in the fact that SRPS do not rely on a linearly ordered sequence of notions that could be taught in isolated or progressive manner. In a sense, they can be understood only by already using them, which obviously sounds contradictory. In order to dissolve this contradiction, we need accepting to work with loosely specified tools, followed by a patient work of successive try-fail and recover steps. This has to take place until the whole process holds together and leads to the desired solution. When this process is completed, the former loosely defined specifications are transformed into exact ones. We call *symbiotic recursive pulsating thinking* (srp-thinking) this way of thinking.

We have previously introduced the term Cartesian thinking for srp-thinking and Newtonian thinking for the other, more usual, one [14]. Both are useful because they apply to solving different problems. Newtonian thinking allows, by words of Newton himself, “standing upon the shoulders of giants”. Newtonian thinking is certainly a SMART approach suitable for solving problems that can be tackled on a modular basis, by analysis and synthesis. Because of the presence of recursion, what we call Cartesian thinking is based on self-justification and self-reference. In terms of deductive systems we can say that Newtonian thinking is a derivation of a knowledge contained implicitly in the given axioms (i.e., these giants’ shoulders) while Cartesian thinking focuses on *creating* a relevant axiomatic system for solving a particular problem. In terms of technological development, Newtonian thinking is concerned with innovation (*building* the intended products relying on already existing knowledge; *handling* ‘truth’) while Cartesian thinking is applied to the cases when the intended product seems utopian or even impossible with respect to the existing available knowledge. It consists in developing new custom-made technologies that were not present so far. Cartesian thinking represents an attempt to *create* a (or *re-create*) ‘truth’ and a ‘desired truth’.

The goal of this paper is to bring introductory insights (via an illustration) concerning one fundamental notion of the SRPS context, namely *informal specification*. This task is not simple in face of prevailing Newtonian paradigm and its particular criteria that are used while evaluating either the process of a scientific research (i.e., how the research should be done) or the final product (i.e., how the result should look-like). Indeed, Newtonian approach requires that research progresses linearly (or modularly) and the results are – in Computer Science – either programs understood in their usual modular sense or proofs about standard properties of these programs. SRPS fail to verify these criteria since they are recursive and symbiotic. Moreover, the results are system-procedures made with sub-procedures (we call them "constructors") that are symbiotic in the same manner as it is the case for Natural Numbers (NAT). NAT are determined (and computed) via the symbiotic constructors 0, successor (*suc*) and NAT themselves. Newtonian thinking often ignores this symbiotic and recursive character of NAT since their representation is possible in seemingly modular form. The same can be said about SRPS. They can be represented in an apparently modular form, but the process of their creation is purely symbiotic and recursive. In NAT the constructors are created via a process that can be described by the informal specifications: Creation(0) = Creation(0,*suc*,NAT), Creation(*suc*) = Creation(0,*suc*,NAT) and Creation(NAT) = Creation(0,*suc*,NAT). This is a symbiotic creation which is analogous to the well-known egg-hen problem: what is created first? Obviously this problem has no known solution. However, a symbiotic solution for this problem is expressed as the simultaneous presence of both hen and egg from the start of the implementation. Let us now describe a visual problem which does have an obvious symbiotic solution. Consider the following well-known picture:

On internet this picture is known as 'young lady and old woman illusion'.

It represents, at the same time, a young lady and an old woman. For some people this is hard to see at once and thus we shall use the following two pictures that help to perceive both women in the above picture:

young woman (yw)

old woman (ow)

The creation of such a picture can be described as a symbiotic creation by which the author before drawing has foreseen the final picture. In other words, the author started with an informal specification "I wish to draw an ambiguous figure and I am almost sure that this will be possible." This is the first step of the creation process. The second step is a *creative preprocessing* by which he foresees the final picture expressed by a formalized specification: Creation(yw) = Creation(yw,ow) and Creation(ow) = Creation(yw,ow). The drawing (implementation) process is the third and last stage of creation. The *creative preprocessing* contains thus research work coming from the informal idea of a symbiotic picture to the presence of the effective tools to give a concrete implementable form to this idea.

We can complete now what has been said above. This paper concerns the introduction of the notion of *informal specification* and brings introductory insights on the *preprocessing stage* in Cartesian design of a particular SRPS. Therefore, it is necessary to be prepared to a non linear symbiotic presentation and to simultaneously consider

- design and meta-design
- action and meta-action
- inseparability of
    - concrete proof
    - abstract overall design.

The paper is organized in the following manner. Section II presents the notion of informal specification and an example of the use of this notion in the domain of Inductive Theorem Proving. Section III presents our fundamental procedure (a *design constructor*) used to reach this goal at the design level. Section IV illustrates this procedure by applying it to the specification of a theorem that will show itself to be, in a particular way, incomplete and thus informal. Section V presents our future research projects.

II. Informal Specifications in Incomplete Frameworks

A. *Informal Specification as a Way of Expressing a Goal*

Due to incompleteness results of Gödel [17], to automate proving theorems by induction (ITP), *ITP-goal*, is unachievable in the context of contemporary mathematics. However, the fact that many apparently unsolvable problems are actually solved when their context or representation are changed has become public knowledge several decades ago when Smullyan [30] started to make game of such transformations. With respect to practical use of ITP for Program Synthesis (via deductive approach pioneered by Manna and Waldinger [23]) and for search of missing axioms in incomplete theories [16], a reasonable implementation of ITP is highly desirable goal (or technological vision). Therefore, in early eighties, building on our previous experience with creating deductive systems and conceptual switches in history of mathematics, we have moved this goal from purely Gödel's mathematical context into 'technological' context by reformulating the above ITP-goal into the following *ITP-system goal*: "Automate ITP as much as possible." The former goal takes into account the underlying Gödel's formal context. Our ITP-system goal leaves some freedom for interpreting and addressing in a non-standard manner not only the problem "How this should be done?" but also the problem "How the solution should look like?" We call *informal specification* a description of any reasonable goal that has this particular property of allowing non-standard criteria for the above two problems (or a description that is in a sense incomplete). We call these problems *pragmatic bias* in contrast to academic bias that expresses, even though implicitly, the necessity for the use of standard criteria for solving these problems.

It must be noted that there can be no ‘Impossible!’ for an informally specified reasonable goal. The only negative statement can be pronounced and that is “I do not know.” It depends only on us if we add “But I will,” or not. With respect to the intended applications of our resulting system (Program Synthesis (PS) and completing incomplete theories), we expressed our intention to add this answer when we realized that the best way to tackle the ITP-system goal is to not search for a some clever decision procedure which will in principle face Gödel’s incompleteness, but a procedure or system that, in failure cases, will provide sufficient conditions for recovery, i.e., increasing the possibility of proving the intended formula. Our approach is thus very different from academic (or Newtonian) approaches to ITP that consider strictly the framework limited by Gödel’s results, such as the system ACL2 [3], the system RRL [22], the system NuPRL [6], the Oyster-Clam system [4], the extensions of ISABELLE [27], the system COQ [26] and Matita Proof Assistant [1].

B. Systemic Background and Guiding Principles for ITP-system goal

We have found that Descartes’ method is perfectly suitable for our task. A short systemic description of Descartes’ method is given in [14]. More precisions are given in [12].

This systemic Cartesian background allows us to summarize the action-guiding principles used in order to conceive an ITP-system:

(*GP1*) We welcome incompleteness for practical reasons since it guarantees progress and evolution. Moreover, each missing axiom discovered hints at a more relevant interpretation. Therefore, a unified method of discovery of missing axioms is an asset.

(*GP2*) We conceive a procedure-system finding sufficient conditions in case of failure and not a decision procedure (see Section III).

(*GP3*) We conceive an ITP-system as a symbiotic recursive pulsative system (see [14]).

(*GP4*) Due to the symbiotic character of tools, we develop first a methodology for ITP, which, when its ‘practical completeness’ will be achieved, a first version of ITP-system will be implemented (this refers to the implemented experimental version 0.3 we presented in [10]).

(*GP5*) The same methodology is conceived for ITP and PS which implies that we adopt no efficiency criteria for synthesized programs.

(GP6) Consequence of the previous principles: We create a database of solutions to non-trivial examples obtained with our methodology. These examples often provide informal specifications of tools that must become part of our methodology. The price to pay is that, in contrast to Newtonian approaches, the real implementation may start only when a plateau is reached in building this database. Such a plateau happens when no new informal specifications of missing tools are discovered (see more in Section VI).

Considering the task formulated in (GP2), and using Beth’s method of deductive tableaux [2] we understood that the first problem to be dealt with is specified informally as finding a method to prove atomic formulas in such a way that it provides, in case of failure, sufficient conditions for provability of this formula. Our solution to this particular tool-specification is our *CM*-formula construction (recalled in the next Section), where *CM* stands for *Constructive Matching*. Since *CM*-formula construction is the basis (or basic symbiotic constructor) of our methodology (see (GP5)), we call it *Constructive Matching methodology* (*CMM*). *CMM* is thus our intended solution for the ITP-system goal.

In the next Section, we recall *CM*-formula construction so that, in Section IV, we can illustrate its use for solving another example (see (GP6)). This example is interesting since it concerns the proof of the so-called Unwinding Theorem met in the domain of information flow security and developed in [29]. In [15] we present the methodological aspects of this problem. In the present paper we want to illustrate how *CMM*, via *CM*-formula construction, handles informal specification of the given Rushby’s theorem. Indeed, as it will become clear, the initial formulation of Rushby’s Unwinding Theorem is, in some sense, incomplete.

III. *CM*-FORMULA CONSTRUCTION IN ITP

In this Section we are going to present the basic mechanism for the *CM*-formula construction originally introduced in [7].

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 a predicate and t_1, t_2 are terms of the axiomatic theory in use. We introduce a new type of arguments in the atomic formula that has to be proven true. We call them **pivotal arguments**, since focusing on them enables to reduce what is usually called the search space of the proof, and to decompose complex problems (such as strategic aspects of a proof) on conceptually simpler problems (such as a transformation of a term into another, possibly finding a sufficient conditions etc.). These pivotal arguments are denoted by ξ (or ξ’ etc.) in the following.

In the first step, the pivotal argument replaces, in a purely syntactical way, one of the arguments of the given formula. The first problem is thus to choose which of the arguments will be replaced by a pivotal argument ξ. A complete algorithmic solution to this problem is not yet proposed, since it will be part of a complete implementation of *CMM* [8]. Nevertheless, a simple informal algorithm is easily obtained in performing a rough analysis of the terms: the most complex one should become the pivot – precisely defining complexity is still left to a human person. Its automation will be tackled with in the final phasis of our research.

In this presentation, let us suppose that we have chosen to work with $F(t_1,\xi)$, the second argument being chosen as the pivotal one. In an artificial, but custom-made manner, we state $C = \{\xi \mid F(t_1,\xi) \text{ is true}\}$. 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 $F(t_1,t_2)$ while preserving the truth of $F(t_1,\xi)$. We thus propose a detour that will enable us to prove also the theorems that cannot be directly proven by the so-called simplification Newtonian 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 $\{ \xi \}$ 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 $cond(\xi)$ is true, we define C_1 by $C_1 = \{\xi \mid cond(\xi)\}$. We can also say that, with the help of the given axioms, we build a 'cond' such that the formula: $\forall\xi \in C_1, F(t_1,\xi)$ 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 ξ_C the result of applying the induction hypothesis to C_1 resulting in its subset $C_2 = \{ \xi_C \mid cond_2(\xi_C)\}$. C_2 is thus such that $F(t_1,\xi_C)$ is true. We are still left with a work to do: prove that t_2 belongs to C_2. In the case that t_2 does not contain existential quantifiers, this is done by verifying $cond_2(t_2)$. In the that case t_2 contains existentially quantified variables, this is done by a new detour. In the first step, we try to solve the problem $cond_2(\xi_C) \Rightarrow \exists\sigma\ (\xi_C = \sigma t_2)$, where σ has to provide a suitable instantiation for the existentially quantified variables in t_2. With such an obtained σ we have then to prove $F(t_1,\sigma t_2)$. In other words, we have to prove that ξ_C and t_2 can be made identical (modulo substitution) when $cond_2(\xi_C)$ holds.

In the case of the success, this completes the proof. In the case of a failure, a new lemma $cond_2(\xi_C) \Rightarrow \exists\sigma\ (\xi_C = \sigma t_2)$ with an appropriate quantification of the involved variables is generated. In some cases, an infinite sequence of 'failure formulas', i.e., lemmas or missing axioms, may be generated. *CMM* is conceived in such a way that the obtained sequence is well-behaving (see [7]) so that a human person or an automated tool (in the future) be driven in the choice of a suitable generalization. This formula logically covers the infinite sequence of lemmas or missing axioms and it thus fills the gap that cannot be overcome by a purely deductive formal approach to theorem proving. In the case of generation of missing axioms, the process of completion the initial theory is performed by 'pulsation', i.e., by adding then applying the new axioms to the domain theory. The resulting system is logically coherent by construction. In the future, all the relevant and necessary tools will be designed (with the help of Machine Learning, Big Data and other relevant domains) to eliminate human interaction with the decision of the appropriateness of suggested missing axioms. This is useful for applications where human interaction is impossible. Consider, for instance, space explorations and constructions by robots.

IV. Example : Proof for an Unwinding Theorem

A. *State-based Information Flow Control – Basic Knowledge*

In this Section we present the formal framework for the so-called unwinding theorem presented in [29]. Then, in the next Section, we present the proof of Rushby's unwinding theorem as performed by *CM*-formula construction.

The proof presented is interesting from the man-machine interaction point of view since it illustrates how a human expert of the domain theory is prompted to find a suitable generalization of a potentially infinite sequence of terms without being asked to know the mechanism of *CM*-formula construction. It is known that, in non-trivial cases, academic approaches require from the user to be aware of the proof assistant mechanisms in order to guide it towards success. In our completed system, the generalizations will be performed automatically.

A system M is composed of a set S of states, with an initial state $s_0 \in S$, a set A of actions, and a set O of outputs, together with the functions step and output: *step*: $S \times A \rightarrow S$, *output*: $S \times A \rightarrow O$. We shall use the letters ... s, t, ... to denote states, letters a, b, ... from the front of the alphabet to denote actions, and Greek letters α, β, ... to denote sequences of actions. Actions can be thought of as "inputs" or "instructions" to be performed by the system; *step(s,a)* denotes the state of the system resulting by performing action *a* in state *s*, and *output*(s,*a*) denotes the result returned by the action. In the following, λ denotes an empty sequence and $\circ$ denotes a concatenation. We shall consider an extension of the function *step* to sequence of actions in the form of a function *run*: $S \times A^* \rightarrow S$, defined by

(ax1) $run(s,\lambda) = s$

(ax2) $run(s,a \circ \alpha) = run(step(s,a),\alpha)$

The agents or subjects interacting with the system and observing the results obtained will be grouped into "security domains". Security domains represent clearances in terms of persons and classifications in terms of data. We thus assume a set *d* of security domains, and a function *dom*: $A \rightarrow d$ that associates a security domain with each action. We shall use letters … u, v, w … to denote domains.

Information is said to flow from a domain u to a domain v when some actions submitted by domain u cause the information about the behavior of the system perceived by domain v to be different from that perceived when those actions are not present. We shall consider the flow of information as a reflexive relation $\rightarrowtail$ on *d* (i.e., u $\rightarrowtail$ u for each domain u.)

A *security policy* will be specified by this relation on *d*. We use $\not\rightarrowtail$ to denote the complement relation i.e., a closed negation of $\rightarrowtail$ on $d \times d$, that is $\not\rightarrowtail = (d \times d) \setminus \rightarrowtail$, where \ denotes set difference. We speak of $\rightarrowtail$ and $\not\rightarrowtail$ as the *interference* and *noninterference* relations, respectively. A policy is said to be *transitive* if its interference is transitive.

We say that domain u *interferes* with domain v if u $\rightarrowtail$ v. We say that an action *interferes* with domain v if there is *dom*(*a*) such that *dom*(*a*) interferes with v, i.e., *dom*(*a*) $\rightarrowtail$ v.

An action a is said to be required *noninterfering* with domain v if *dom*(*a*) $\not\rightarrowtail$ v for all action sequences that contain a. The function *purge*: $A^* \times d \rightarrow A^*$ is defined as follows

(ax3) $purge(\lambda,v) = \lambda$

(ax4) $purge(a \circ \alpha,v) = a \circ purge(\alpha,v)$, if $dom(a) \rightarrowtail v$

(ax5) $purge(a \circ \alpha,v) = purge(\alpha,v)$, if $dom(a) \not\rightarrowtail v$.

The machine is *secure* if a given domain v is unable to distinguish between the state of the machine after it has processed a given action sequence, and the state after processing the same sequence purged of actions required to be noninterfering with v.

Formally, the security is identified with the requirement that $output(run(s_0,\alpha),a) = output(run(s_0,purge(\alpha,\mathrm{dom}(a))),a)$.

For convenience, we introduce the functions *do*: $A^* \rightarrow S$ and *test*: $A^* \times A \rightarrow O$ to abbreviate the expressions in the last requirement: $do(\alpha) = run(s_0,\alpha)$, and $\mathrm{test}(\alpha,a) = output(do(\alpha),a)$. Then we say that system M is secure for the policy $\rightarrowtail$ if

$$test(\alpha,a) = test(purge(\alpha,dom(a)),a) \qquad (1)$$

for all actions sequences α and actions a.

The non-interference definition of security is expressed “globally” in (1) in terms of sequences of actions and state transitions. In order to obtain sufficient “local” conditions for verifying the security of systems, Rushby introduces a set of conditions on individual state transitions.

A system M is *view-partitioned* if, for each domain u from *d*, there is an equivalence relation $\overset{u}{\sim}$ on S. These equivalence relations are said to be *output consistent* if

$$s \overset{dom(a)}{\sim} t \;\Rightarrow output(s,a) = output(t,a). \qquad (2)$$

The following result allows relating the output consistency to security of the system.

Lemma 1:

Let $\rightarrowtail$ be a policy and M a view partitioned, output consistent system such that

$$do(\alpha) \overset{u}{\sim} do(purge(\alpha,u)). \qquad (3)$$

Then M is secure for $\rightarrowtail$.

Proof: see [29].

Let M be a view-partitioned system and $\rightarrowtail$ a policy. We say that M *locally respects* $\rightarrowtail$ if

$$dom(a) \not\rightarrowtail u \;\Rightarrow s \overset{u}{\sim} step(s,a) \qquad (4)$$

and that M is *step consistent* if

$$s \overset{u}{\sim} t \;\Rightarrow\; step(s,a) \overset{u}{\sim} step(t,a). \qquad (5)$$

The following theorem shows that the local conditions formulated are sufficient to guarantee security.

Theorem 1: (Unwinding Theorem)

Let $\rightarrowtail$ be a policy and M a view-partitioned system that is output consistent, step consistent, and locally respects $\rightarrowtail$. Then M is secure for $\rightarrowtail$.

We have thus recalled the basic knowledge formalizing the information needed by an automated theorem prover.

B. *CMM Suggests a Generalization Necessary to Prove the Unwinding Theorem*

As we said above, we shall suppose that system M is output consistent, step consistent, and locally respects $\rightarrowtail$. To prove this theorem it is sufficient to prove that (3) holds.

Using our above *CM*-formula construction algorithm, we shall study what operations have to be performed in order to prove formula (3) introduced above:

$$do(\alpha) \overset{dom(b)}{\sim} do(purge(\alpha,dom(b))), \qquad (3)$$

for arbitrary domain *dom*(b) and state α.

By definition of *do*, *do*(α) is $run(s_0,\alpha)$ and similarly for *do*(*purge*(α,*dom*(b))). We thus obtain that the goal is to prove the formula (original theorem)

$$run(s_0,\alpha) \overset{dom(b)}{\sim} run(s_0,purge(\alpha,dom(b))). \quad \text{(UTh)}$$

Let us consider a proof by induction on α. This means to consider the base step for α = λ and the induction step for α = $a \circ \alpha'$, where *a* is an arbitrary action and α’ is a sequence of actions. As the proof for the base step is easy, we focus on the proof of the induction step.

In the induction step, α is $a \circ \alpha'$. The induction hypothesis is

$$run(s_0,\alpha') \overset{dom(b)}{\sim} run(s_0,purge(\alpha',dom(b))). \qquad (6)$$

The goal is to prove

$$run(s_0,a \circ \alpha') \overset{dom(b)}{\sim} run(s_0,purge(a \circ \alpha',dom(b))). \quad (7)$$

using the induction hypothesis and the properties of M.

The *CM*-formula construction requires that we replace one of arguments of (7) by pivotal argument. Since the term at the right side is more complex than the term on the left side, we chose to replace this complex term by the pivotal argument ξ. This gives

$$run(s_0,a \circ \alpha') \overset{dom(b)}{\sim} \xi. \tag{8}$$

By definition,

$$run(s_0,a \circ \alpha') = run(step(s_0,a), \alpha')$$

This gives

$$run(step(s_0,a), \alpha') \overset{dom(b)}{\sim} \xi. \tag{9}$$

We would like now to apply the induction hypothesis (6). This means to compare $run(s_0,\alpha')$ in (8) and $run(step(s_0,a), \alpha')$ in the last formula (9). This fails. Therefore, *CM*-formula construction generates a new lemma expressed in terms of the failure formula

$$run(step(s_0,a), \alpha') \overset{dom(b)}{\sim} run(s_0,purge(a \circ \alpha',dom(b)))\ .$$

For simplicity of our presentation here we do not evaluate the term $purge(a \circ \alpha',dom(b))$.

In the last formula, all the variables are universally quantified. The proof is by induction and the variable α' becomes the induction variable. In the base step, $\alpha' = \lambda$ and the induction step for $\alpha' = c \circ \gamma$, where c is an arbitrary action and γ is a sequence of actions.

The base step for this new goal would lead to discovery of a missing precondition. In this paper we would like to insist more on the discovery of a need for a generalization. Therefore, we shall skip the base step and we shall go directly to the induction step.

In the induction step, since $\alpha' = c \circ \gamma$, the induction hypothesis is the formula

$$run(step(s_0,a), \gamma) \overset{dom(b)}{\sim} run(s_0,purge(a \circ \gamma,dom(b))). \tag{10}$$

and the goal to prove is the formula

$$run(step(s_0,a), c \circ \gamma) \overset{dom(b)}{\sim} run(s_0,purge(a \circ c \circ \gamma,dom(b))).$$

The *CM*-formula construction replaces the right hand term by an abstract argument ξ. This yields

$$run(step(s_0,a), c \circ \gamma) \overset{dom(b)}{\sim} \xi.$$

The evaluation of

$$run(step(s_0,a), c \circ \gamma)$$

is $run(step(step(s_0,a),c), \gamma)$, i.e., we have to consider the formula

$$run(step(step(s_0,a),c), \gamma) \overset{dom(b)}{\sim} \xi.$$

CM-construction tries to apply the induction hypothesis (10), but it fails, since there are no axioms that would put into relation the terms $step(s_0,a)$ and $step(step(s_0,a),c)$. This means that a new lemma expressing this relationship is necessary in order to complete the proof. The use of induction on growing terms will, of course, not solve our problem that recurs at each step:

$$run(s_0,\alpha) \overset{dom(b)}{\sim} run(s_0,purge(\alpha,dom(b)))$$
$$run(step(s_0,a), \alpha') \overset{dom(b)}{\sim} run(s_0,purge(a \circ \alpha',dom(b)))$$
$$run(step(s_0,a), c \circ \gamma) \overset{dom(b)}{\sim} run(s_0,purge(a \circ c \circ \gamma,dom(b)))$$
$$\ldots$$

Nevertheless, this sequence of failures contains an infinite sequence of 'unprovable' lemmas (in the context of the axioms we use). This 'unprovability' is expressed by means of growing terms. A rather obvious solution is thus to suppose that we miss a lemma in which the sequence of these growing terms is generalized by a variable 's'. This new variable s replaces the following sequence of growing terms:

$$s_0$$
$$step(s_0,a)$$
$$step(step(s_0,a),c)$$
$$\ldots$$

Thus the original theorem (UTh) is replaced by the goal to prove a formula into which s_0 is replaced by s in the function 'run' in the non-pivotal argument (i.e., $s_0 \rightarrow s$ in $run(s_0,\alpha)$ of (UTh)). It follows that our task to prove the original theorem (UTh) is replaced by the goal to prove

$$run(s,\alpha) \overset{dom(b)}{\sim} run(s_0,purge(\alpha,dom(b))). \tag{UThG}$$

Again the proof is by induction on α. In the base step, α is λ. The goal is thus prove

$$run(s,\lambda) \overset{dom(b)}{\sim} run(s_0,purge(\lambda,dom(b))). \tag{11}$$

We introduce the pivotal argument here and thus we have to consider

$$run(s,\lambda) \overset{dom(b)}{\sim} \xi. \tag{12}$$

By definition, $run(s,\lambda)$ is s. This means that (12) changes to $s \overset{dom(b)}{\sim} \xi$. We check now whether ξ can be transformed into the right side of (11), that is $run(s_0,purge(\lambda,dom(b)))$. Because of axioms (ax3) and (ax1), the evaluation of $run(s_0,purge(\lambda,dom(b)))$ is s_0. A pivotal argument can be replaced by s_0. This gives that we are left with checking the formula

$$s \overset{dom(b)}{\sim} s_0. \tag{13}$$

We have no way to prove this and thus this formula becomes a missing precondition to (UThG). In other words, we have to prove the formula (Lm1):

$$s \overset{dom(b)}{\sim} s_0 \Rightarrow run(s,\alpha) \overset{dom(b)}{\sim} run(s_0,purge(\alpha,dom(b))).$$

In order to somewhat shorten this example, we can tell that, if we try to prove (Lm1) following the steps described in Section III, we will again fail and generate yet another infinite sequence of lemmas that leads us to the following generalization (Lm2) in which s_0 is generalized to 't' on both sides of the implication. Note that, at the start of our proof, this generalization is by no means intuitively obvious and it does deserve the effort we put in its discovery (Lm2):

$$s \overset{dom(b)}{\sim} t \Rightarrow run(s,\alpha) \overset{dom(b)}{\sim} run(t,purge(\alpha,dom(b))).$$

In order to prove lemma (Lm2), we again use *CM*-formula construction and this will lead to a success as detailed in Appendix of [15]. The initial formula (UTh) is a particular instance of (Lm2) with $s = s_0 = t$.

This means that initial Rushby's Unwinding Theorem, i.e., the formula (UTh), is in a sense incomplete from the theorem proving point of view. Indeed, the available axioms are not sufficient to prove the given theorem in its original form. It has to be generalized. Note that Rushby's goes directly to proving a generalized formula (Lm2) without explaining the reasons and motivations for this generalization. The above presentation shows that, in our approach, the motivations for this generalization are expressed as a (possibly infinite) sequence of failure formulas that contain a sequence of terms and these terms increase regularly.

A proof of (Lm2) using *CMM* as well as its comparison with Rushby's proof can be found in [15].

Summarizing, our example here shows that the *CM*-formula construction is particularly suited to finding missing preconditions (as we found in formula (13)) and suggesting a need for useful generalizations leading to (Lm2), as we have just shown. In other words, it is particularly effective in recovery from failures. Our paper [16] shows that our approach is able to suggest even missing axioms. As said above, in the future, integration of the suggested axioms will also be automatically handled.

V. FUTURE WORK

Despite our previous success with solving also non-standard problems such as n-queens [9], manipulation of blocks in robotics [16] and unusual reformulation of Ackermann's function [13], the practical completeness of *CMM* is not yet obtained. We need to extend our investigations also to non-atomic formulas, *etc*. This is why we continue in our research in the domain of information flow security that is nowadays important and challenging. As a complementary work to [21] and [20], we plan to help in the search of the necessary extensions of *CMM* in this field by the attempts for mechanized proofs of Unwinding Theorems presented in Mantel's thesis [24] and, among others [18] [28] [19] and [25]. To our best knowledge there is no other existing work related to the automation of the inductive proofs of these theorems. The challenge is here the execution of proofs for theorems that contain non-transitive relations. Our research question is: Do non-transitive relations require specific tools that are not yet present in *CMM*? Are there other problems that we did not meet yet?

We are quite sure that our future investigations concerning also the use of *CMM* for formalizing deductive theories requiring recursion will be very fruitful and will suggest new problems to be handled and new tools to be developed in the field of Machine Learning and Computational Creativity. Mantel's work [24] is our first objective in this direction. Our research question is: Can Mantel's work be enhanced by use of an ITP-system well suited also for proving theorems containing existential quantifiers?

VI. CONCLUSION

Research shows (see [5]) that even a team of super-gifted people is unable to work together if they do not develop, in what we call "research's preliminary phase", a common vocabulary for their already known particular personal tools so that they become able – together – to develop a new custom-made vocabulary for their intended technological vision.

The main contribution of this paper is the introduction of one of fundamental notions for such research preliminary phases of any technological vision made accessible in the framework of SRPS, namely informal specification.

On our example of progressive building the fundamentals of *CMM* for ITP we have illustrated that, due to symbiotic and recursive character of SRPS, the missing tools of *CMM* are informally specified while *by-hand* experimenting challenging examples. The difficulty of this by-hand experimentation lies in the following points:

- All the experiences are performed strictly following *CM*-formula construction and relying on previously informally *on-purpose specified* tools (in our example here: evaluation, generation of induction hypotheses, application of induction hypotheses, terms transformation and generalization), i.e., these experiences are not led by the personal talent of an experimenter.
- Each observation concerns not only specifying (informally) missing tools (in our example here: handling non-recursive formulas as explained in [15]) but also refining informal specifications of already introduced tools (in our example here: some new features of generalization were found; their presentation is out of scope of this paper). This explains and justifies our by-hand research instead of automated experiences in which subtle patterns may be lost.
- The talent (if any) of experimenter is strictly reduced to looking for patterns that either have nothing to do with the semantic of the domain in which ITP is performed or, if this is not the case, the experimenter

must justify their adequate introduction into our Theory of Constructible Domains [11] (this is a particular theory of representation of definitions of recursive functions and predicates suitable for *CMM*); in other words, no domain or problem dependent heuristics are allowed in by-hand experiments.

This means that persistent and relatively humble systemic creativity and goal awareness are the main features of srp-thinking.

This paper explains that there should be no conflict between Newtonian and Cartesian srp-thinking. Both apply to different problems, they are complementary. The problem arises only when the Newtonian criteria are applied to the evaluation of the research on SRPS. This manifests namely by Newtonians rejecting the necessary long term by-hand experimenting and informally specified notion of 'practical completeness'.

ACKNOWLEDGMENTS

Michèle Sebag and Dieter Hutter contributed ideas to improve this paper. This conference referees' feedback is gratefully acknowledged.

REFERENCES

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

[2] E. Beth,The Foundations of Mathematics; North-Holland,1959.

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

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

[5] R. Chauvin, Les Surdoués (Super-gifted); Stock, 1975.

[6] R. L. Constable, Implementing Mathematics with the Nuprl proof development system; Prentice-Hall, Inc., Englewood Clifs, New Jersey, 1986.

[7] M. Franova, "CM-strategy : A Methodology for Inductive Theorem Proving or Constructive Well-Generalized Proofs," in A. K. Joshi, (ed), Proceedings of the Ninth International Joint Conference on Artificial Intelligence; Los Angeles, pp. 1214-1220, 1985.

[8] M. Franova, "Fundamentals for a new methodology for inductive theorem proving: CM-construction of atomic formulae," in Y. Kodratoff (ed.), Proceedings of the 8th European Conference on Artificial Intelligence; August 1-5, Pitman, London, United Kingdom, pp. 137-141, 1988.

[9] M. Franova, "An Implementation of Program Synthesis from Formal Specifications," in Y. Kodratoff, (ed.), Proceedings of the 8th European Conference on Artificial Intelligence; August 1-5, Pitman, London, United Kingdom, pp. 559-564, 1988.

[10] M. Franova, "Precomas 0.3 User Guide," Rapport de Recherche No.524, L.R.I., Université de Paris-Sud, Orsay, France, October, 1989.

[11] M. Franova, "A Theory of Constructible Domains - a formalization of inductively defined systems of objects for a user-independent automation of inductive theorem proving, Part I," Rapport de Recherche No.970, L.R.I., Université de Paris-Sud, Orsay, France, Mai, 1995.

[12] M. Franova, Créativité Formelle: méthode et pratique - Conception des systèmes "informatiques" complexes et Brevet Épistémologique (Formal Creativity : method and practice – Design of complex "computational" systems and Epistemological Patent), Publibook, 2008.

[13] 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.

[14] M. Franova, "Cartesian versus Newtonian paradigms for recursive program synthesis," International Journal on Advances in Systems and Measurements, vol. 7, no 3&4, pp. 209-222, 2014.

[15] M. Franova, D. Hutter and Y. Kodratoff, "Algorithmic conceptualization of tools for proving by induction « Unwinding » Theorems – A Case Study," Rapport de Recherche N◦ 1587, L.R.I., Université de Paris-Sud, Orsay, France, Mai 2016.

[16] M. Franova and Kooli M., "Recursion Manipulation for Robotics: why and how?"; in R. Trappl, (ed.), 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.

[17] K. Gödel, "The completeness of the axioms of the functional calculus of logic", in: J. van Heijenoort, From Frege to Godel, A source book in mathematical logic, 1879-1931, Harvard University Press, pp. 582-592, 1967.

[18] J. Graham-Cumming and J.W. Sanders, "On the refinement of non-interference," Proc. of the IEEE Symposium on Security and Privacy, pp. 11-20, 1982.

[19] J. T. Haigh and W. D. Young, "Extending the noninterference version of MLS for SAT; IEEE Trans. Software Eng. 13(2), pp. 141-150, 1987.

[20] D. Hutter, H. Mantel, I. Schaefer and A. Schairer, "Security of multi-agent systems: A case study on comparison shopping," Journal of Applied Logic, Volume 5, Issue 2, pp. 303-332, June 2007.

[21] D. Hutter, "Automating Proofs of cnwinding conditions," in S. Autexier, H. Mantel (eds.), Workshop Proceedings VERIFY06 at the International Joint Conference on Automated Reasoning, Seattle, 2006.

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

[23] Z. Manna and R.Waldinger, "A Deductive approach to Program Synthesis," in ACM Transactions on Programming Languages and Systems, Vol. 2., No.1, pp. 90-121, 1980.

[24] H. Mantel, "A uniform framework for the formal specification and verification of information flow security," PhD thesis, Universitty of Saarlandes, 2003.

[25] J. K. Millen, "Unwinding Forward Correctability," in Proc. of the 7th IEEE Computer Security Workshop, pp. 35-54, 1994.

[26] 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, 1993.

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

[28] S. Pinsky, "Absorbing covers and intransitive non-interference," in Proceedings of IEEE Symposium on Security and Privacy, pp. 102 - 113, 1995.

[29] J. Rushby, "Noninterference, transitivity, and channel-control security policies," Technical Report CSL-92-02, Computer Science Laboratory SRI International, December, 1992.

[30] R. M. Smullyan, What is the Name of This Book? - The Riddle of Dracula and Other Logical Puzzles; Penguin, 1981.