RAPPORT DE RECHERCHE

L R I

ALGORITHMIC CONCEPTUALIZATION OF TOOLS FOR PROVING BY INDUCTION « UNWINDING » THEOREMS A CASE STUDY

FRANOVA M / HUTTER D / KODRATOFF Y

Unité Mixte de Recherche 8623
CNRS-Université Paris Sud-LRI

05/2016

Rapport de Recherche N° 1587

CNRS – Université de Paris Sud
Centre d'Orsay
LABORATOIRE DE RECHERCHE EN INFORMATIQUE
Bâtiment 650
91405 ORSAY Cedex (France)

Algorithmic Conceptualization of Tools for Proving by Induction « Unwinding » Theorems – A Case Study

Marta Franova[1], Dieter Hutter[2] and Yves Kodratoff[1]

mf@lri.fr, hutter@dfki.de, yvkod@gmail.com
[1]LRI, UMR8623 du CNRS & INRIA Saclay, Orsay, France
[2]German Research Center for Artificial Intelligence, Bremen, Germany

Abstract: Information flow security is an interesting and challenging problem in which the use of inductive theorem proving is required for dealing with the so-called 'unwinding theorems'. This paper examines whether *Constructive Matching Methodology* (*CMM*) for proving theorems by induction can be used also in this industry oriented framework. We show here that nothing more than a minor extension is to be considered, namely handling non recursive functions and predicates. But many solutions for this problem are already available. However dealing with possibly incomplete recursive theories and their formulation is still a problem. This paper illustrates that *CMM* is well suited in both automated and non-automated versions in the context of information flow security.

I. Introduction

Our goal is to provide an algorithmic conceptualization of tools for proving by induction the so-called 'unwinding theorems' needed in the field of information flow security. The methodology we start from is *Constructive Matching Methodology* (*CMM*) ([franova64], [franova10]) and we shall specify the tools needed to reach our goal. In this paper, we focus on a particular example in the domain of information flow security. We thus do not need to explain the semantic of the security problem: To this purpose the reader is referred to [rushby01]. We just recall the basic facts that are used for our formal development from the point of view of inductive theorem proving.

In the first part of this paper we recall the basic definitions and results related to security of an information system presented by Rushby in [rushby01]. Then, in the second part, we study an inductive proof for Rushby's formulation of a particular 'unwinding' theorem and we consider which tools are required to extend *CMM* in order to perform such a proof. The presented 'non-clever' proof differs from Rushby's 'clever' proof and we point out the main differences as well as the advantage of considering automating 'non-clever' proofs.

II. State-based information flow control – basic knowledge

Definition 1.

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 × A → S
- *output*: S × A → 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 ∘ denotes a concatenation.

We shall consider an extension of the function *step* to sequence of actions in the form of a function *run*

- *run*: S × A^* → S,

defined by

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

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

Note that this definition defines *run* in terms of *run*(*step*(s,*a*),α) and not *step*(*run*(s,*a*),α). This is required by the semantic model of more complex development in which such a use is essential. Of course, such a definition complicates recursive proofs using this definition. In [franova62] we qualify this kind of function *mutilating*.

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 → *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 behaviour 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 ⤍ on *d*. (i.e. u ⤍ u for each domain u.)

A *security policy* will be specified by this relation on *d*. We use ⤍/ to denote the complement relation (i.e. a 'closed' negation of ⤍ on *d* × *d*), that is

$$⤍/ = (d \times d) \setminus ⤍$$

where \ denotes set difference. We speak of ⤍ and ⤍/ 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 ⤍ 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*) ⤍ v.

An action a is said to be required *noninterfering* with domain v if *dom*(*a*) ⤍/ v for all action sequences that contain a.

In order to go from an action sequence to a sequence that is purged of actions noninterfering for a domain, the following function is defined.

Definition 2.

- *purge*: $A^* \times d \rightarrow A^*$

is defined as follows

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

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

(ax5) $\quad purge(a \circ \alpha,v) = purge(\alpha,v),$ 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

$$output(run(s_0,\alpha),a) = output(run(s_0,purge(\alpha,dom(a))),a).$$

For convenience, we introduce the functions *do* and *test* to abbreviate the expressions in the last requirement.

- *do*: $A^* \rightarrow S$
- *test*: $A^* \times A \rightarrow O$

where

$$do(\alpha) = run(s_0,\alpha), \quad \text{and}$$

$$test(\alpha,a) = output(do(\alpha),a).$$

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.

As Rushby says [rushby01], the intuition behind the security of a system is that, starting in the initial state s_0, the system performs an action sequence α and reaches state $do(\alpha)$. At that point, the action *a* and the corresponding test(α,*a*) are observed. We can think of action *a* and its output observation as an experiment performed by *dom*(*a*) in order to learn something about this action sequence α. If dom(*a*) can distinguish between the outputs of the action sequences α and *purge*(α,*dom*(*a*)) then an action by some domain u has "interfered" with *dom*(*a*), i.e. u $\rightarrowtail$ *dom*(*a*) and the system is not secure with respect to policies that specify u $\not\rightarrowtail$ *dom*(*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.

The first step in this development is to partition the states of the system into equivalence classes that all "appear identical" to a given domain. The verification technique will then have to prove that each domain's view of the system is unaffected by the actions of domains that are required to be noninterfering with it.

The following part expresses this formally.

Definition 3.

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

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

Then M is secure for $\dashrightarrow$.

Proof:
It is sufficient to put u = *dom*(*a*). Then, since M is output consistent, (2) holds for any state s and t. Let us put now s = *do*(α) and t = *do*(*purge*(α,*dom*(*a*)).
Since

$$do(\alpha) \overset{dom(a)}{\sim} do(purge(\alpha,dom(a)) \tag{4}$$

by modus ponens form (2) we have

$$output(do(\alpha),a) = output(do(purge(\alpha,dom(a)),a).$$

But, in our notation, this is nothing else than (1).

Definition 4.
Let M be a view-partitioned system and $\dashrightarrow$ a policy.
We say that M *locally respects* $\dashrightarrow$ if

$$dom(a) \not\dashrightarrow u \Rightarrow s \overset{u}{\sim} step(s,a) \tag{5}$$

and that M is *step consistent* if

$$s \overset{u}{\sim} t \Rightarrow step(s,a) \overset{u}{\sim} step(t,a). \tag{6}$$

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

Theorem 1: (Unwinding Theorem)
Let $\dashrightarrow$ be a policy and M a view-partitioned system that is

1. output consistent,
2. step consistent, and
3. locally respects $\dashrightarrow$.

Then M is secure for $\dashrightarrow$.

We have thus recalled the basic knowledge formalizing the information needed by an automated theorem prover. The next section focuses first on a by-hand version of an inductive proof for **Theorem 1** based on use of **Lemma 1**. This proof is then studied from the point of view of its automation. The proof presented in [rushby01] is a clever human generalization of proving (3) taking into account output consistency, step consistency and the fact that M locally respects $\dashrightarrow$.

We do not start with this generalization in order to show how a 'non-clever' proof of Unwinding Theorem leads to detecting the need of a generalization[1]. It is so because we have realized decades ago that Cartesian method of solving problems generating experiences (i.e. hints to the problems) and providing hints for solving problems thus

[1] The need for generalization is a known problem in automation of inductive theorem proving. In many cases it is related to use of mutilating functions.

generated

- gives rise to questions specific to inductive theorem proving and not to the semantic of a particular problem as it is in the case of solving problems based on the intelligence of a clever computer scientist. Even more importantly
- hints at missing information of which such a computer scientist might even unaware.

We are therefore convinced that it is exactly this way of 'non-clever' approach to proofs that will bring, in the long term, most of knowledge useful for automation of inductive theorem proving. This will allow to isolate the algorithmic concepts that can, in some cases, replace complex semantic considerations of a clever computer scientist and even help this computer scientist with refining or completing the theory he is relying on or trying to formulate in a formal way. In other words, instead of aiming at the direct intelligence of a system, we follow one of the main approaches to AI, that is building a 'prosthesis' for intelligence.

III. By-hand proof for Unwinding Theorem

As we said above, we shall suppose that system M is

1. output consistent,
2. step consistent, and
3. locally respects $\not\rightarrow$.

We shall study what operations have to be performed in order to prove the above presented formula

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

for arbitrary domain $dom(b)$ and state α.
By definition of d, $do(\alpha)$ is $run(s_0,\alpha)$ and similarly for $do(purge(\alpha,dom(b)))$. We thus obtain:

$$run(s_0,\alpha) \overset{dom(b)}{\sim} run(s_0,purge(\alpha,dom(b))) . \tag{7}$$

Let us consider a proof by induction on α. This means to consider the base step for $\alpha = \lambda$ and the induction step for $\alpha = a \circ \alpha'$, 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

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

The goal is to prove

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

using the induction hypothesis and the properties of M.
By definition,

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

and the value of $run(s_0,purge(a \circ \alpha',dom(b))$ depends on whether $dom(a) \not\rightarrow dom(b)$ or $dom(a) \not\not\rightarrow dom(b)$ holds.

Let us consider the case

$$dom(a) \dashrightarrow dom(b). \tag{10}$$

then $purge(a \circ \alpha', dom(b)) = a \circ purge(\alpha', dom(b))$ and so $run(s_0, a \circ purge(\alpha', dom(b))) = run(step(s_0,a), purge(\alpha', dom(b)))$.
This means that, in this case we have to prove

$$run(step(s_0,a),\alpha') \overset{dom(b)}{\sim} run(step(s_0,a),purge(\alpha',dom(b))) . \tag{11}$$

The induction hypothesis (8) cannot be applied and so this last formula becomes a new lemma to be proved taking into account the condition (10). A possible answer to classical problem can be to refine the case analysis in order to prove the theorem without resort to induction. Our proposal, in this paper, is to stick to structural induction and to treat the formula (11) as new lemma to be proved taking into account condition (10). This behavior will lead us to isolate a generalization condition, as we shall now illustrate.

Since α' belongs to A^*, it is either λ or $c \circ \gamma$ for some $c \in A$ and $\gamma \in A^*$. Since the base step for (11) is easy, we shall focus on the induction step.

In the induction step for (11), $\alpha' = c \circ \gamma$ and the induction hypothesis is

$$run(step(s_0,a),\gamma) \overset{dom(b)}{\sim} run(step(s_0,a),purge(\gamma,dom(b))) . \tag{12}$$

The formula to be proved is

$$run(step(s_0,a), c \circ \gamma) \overset{dom(b)}{\sim} run(step(s_0,a),purge(c \circ \gamma,dom(b))). \tag{13}$$

By definition,

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

and, with respect to the condition $dom(a) \dashrightarrow dom(b)$,
$run(step(s_0,a),purge(c \circ \gamma,dom(b)) = run(step(step(s_0,a),c),purge(\gamma,dom(b))$

This means that we have to prove

$$run(step(step(s_0,a),c), \gamma) \overset{dom(b)}{\sim} run(step(step(s_0,a),c),purge(\gamma,dom(b))) \tag{14}$$

Once again, neither the induction hypothesis (12) nor any of the properties of M can be applied.
However, the generated 'failure' formulas (7), (11) and (14) are suggesting that a generalization may be needed.
Let us have a closer look at these formulas. Formula (7), i.e.

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

is the initial formula to prove. We attempt to prove it by induction on α. This proof generates (11), i.e.

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

We are unable to prove (11) because the induction hypothesis cannot be applied. We then again try to prove (11) by induction on α' such that $\alpha = a \circ \alpha'$and generate (14), i.e.

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

We observe a regularly growing sequence on both sides of these formula, namely:

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

on both sides of these three formulas. We shall postpone a consideration of the generalization of these growing sequences when seeking further indices in a proof of the case

$$dom(a) \not\rightarrow dom(b). \tag{15}$$

In this case,

$$purge(a \circ \alpha', dom(b)) = purge(\alpha', dom(b))$$

and so

$$run(s_0, a \circ purge(\alpha', dom(b))) = run(s_0, purge(\alpha', dom(b))).$$

This means that formula to be proved (9) evaluates to

$$run(step(s_0,a),\alpha') \overset{dom(b)}{\sim} run(s_0,purge(\alpha',dom(b))). \tag{16}$$

Since $\overset{dom(b)}{\sim}$ is an equivalence relation, it means that the induction hypothesis (8) can be applied. In other words, the term $run(s_0,purge(\alpha',dom(b)))$ can be replaced by the left side of the induction hypothesis, i.e. the term $run(s_0,\alpha')$. This means that the application of the induction hypothesis (8) to (16) leads to the formula

$$run(step(s_0,a),\alpha') \overset{dom(b)}{\sim} run(s_0,\alpha') \tag{17}$$

which has to be proved. Since the only difference between both terms of (17) are their first arguments, namely $step(s_0,a)$ and s_0, we look among the given assumptions of the Unwinding Theorem for one which would allow to conclude that (17) is true. We can see that since M locally respects $\not\rightarrow$, and $\overset{dom(b)}{\sim}$ is symmetric, $step(s_0,a) \overset{dom(b)}{\sim} s_0$ whenever $dom(a) \not\rightarrow dom(b)$, which is just the case we are considering. $\overset{dom(b)}{\sim}$ is an equivalence relation and this means that $step(s_0,a) \overset{dom(b)}{\sim} s_0$ implies

$$run(step(s_0,a),\mu) \overset{dom(b)}{\sim} run(s_0,\mu)$$

for any μ from A^*. It holds therefore for $\mu = \alpha'$. Thus, (17) is true.
The considered case $dom(a) \not\rightarrow dom(b)$ is thus proved without difficulties. Therefore, let us return to the condition $dom(a) \not\rightarrow dom(b)$ where we had generated the sequence

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

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

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

$$\ldots$$

We now generalize this sequence to

$$run(s,\alpha) \overset{dom(b)}{\sim} run(t,purge(\alpha,dom(b))). \tag{18}$$

A clever reader may be aware that this formula is not true but we shall show just below how our ‘non-clever’ approach finds a new suitable formulation.
The above examples underline the fact that our methodology based on the attempt to capture methodically and algorithmically non-clever by-hand proving of theorems (see [franova14]) has the positive feature that it generates “well-ordered” sequences of

'failure' formulas that can be generalized. The generalization itself, however, is not a part *CMM* for the time being. Generalizing is a problem entirely different from our present goals and can also left in the hands of the user who, now, knows at least what has to be generalized.

Let us now show how we may be able to 'prove' (18)**:**

Base step :
In the base step, $\alpha = \lambda$. The goal is to prove

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

Using the definitions of *run* and *purge* we obtain that the goal is to prove

$$s \overset{dom(b)}{\sim} t. \tag{19}$$

Since there is no assumption corresponding to this formula, (19) becomes a precondition for the formula (18). In other words the failure to prove (18) leads us to discovery of a missing precondition. The failure to prove the Unwinding Theorem directly led us from (7) to a discovery of (18) and now of the following formula (still to be proven):

$$s \overset{dom(b)}{\sim} t \Rightarrow run(s,\alpha) \overset{dom(b)}{\sim} run(t,purge(\alpha,dom(b))) \tag{20}$$

This formula is a generalization of (7) since we can put $s_0 = s = t$ and thus (20) can be considered in its particular form (still to be proven):

$$s_0 \overset{dom(b)}{\sim} s_0 \Rightarrow run(s_0,\alpha) \overset{dom(b)}{\sim} run(s_0,purge(\alpha,dom(b))) \tag{21}$$

We go on, now trying to prove (20):

Base step :
In the base step, $\alpha = \lambda$. Supposing $s \overset{dom(b)}{\sim} t$, the goal is to prove

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

Using the definitions of *run* and *purge* we obtain that the goal is to prove

$$s \overset{dom(b)}{\sim} t. \tag{22}$$

This is our assumption and so the base step is resolved.

Induction step:
In the induction step, $\alpha = d \circ \delta$ and we have the induction hypothesis

$$s' \overset{dom(b)}{\sim} t' \Rightarrow run(s',\delta) \overset{dom(b)}{\sim} run(t',purge(\delta,dom(b))) \tag{23}$$

that holds for each s' and t' (i.e. s' and t' are universally quantified in (23)).
Le us suppose that

$$s \overset{dom(b)}{\sim} t \tag{24}$$

holds. The goal is to prove

$$run(s, d \circ \delta) \overset{dom(b)}{\sim} run(t, purge(d \circ \delta, dom(b))) \tag{25}$$

The evaluation of $run(s, d \circ \delta)$ gives $run(step(s,d),\delta)$. With respect to the definition of *purge*, for the term $run(t,purge(d \circ \delta,dom(b))$ two cases have to be considered:

case A: $dom(d) \dashrightarrow dom(b)$, and
case B: $dom(d) \not\dashrightarrow dom(b)$.

Let us consider the case A, i.e. we assume that $dom(d) \dashrightarrow dom(b)$ holds. In this case,

$run(t,purge(d \circ \delta,dom(b))) =$
$run(t, d \circ purge(\delta,dom(b))) =$
$run(step(t, d),purge(\delta,dom(b)))$.

Thus, in the case A we have to prove

$$run(step(s,d),\delta) \overset{dom(b)}{\sim} run(step(t, d),purge(\delta,dom(b))). \tag{26}$$

Since no evaluations can be performed, we look at the possibility to apply the induction hypothesis. We can see that the instantiation of s' by $step(s,a)$ and of t' by $step(t, d)$ could be done if

$$step(s,a) \overset{dom(b)}{\sim} step(t, d) \tag{27}$$

holds, i.e. we have that this particular induction hypothesis reads

$$run(step(s,d),\delta) \overset{dom(b)}{\sim} run(step(t, d),purge(\delta,dom(b))), \tag{28}$$

$$\text{if} \quad step(s,d) \overset{dom(b)}{\sim} step(t,d)$$

We look at the properties of M to see whether there is one relative to *step* function and the terms $step(s,a)$ and $step(t,d)$. We have only *step consistency* of M. The *step consistency* of M means that

$$s \overset{dom(b)}{\sim} t \Rightarrow step(s,d) \overset{dom(b)}{\sim} step(t,d) \tag{29}$$

holds. The antecedent of (29) is nothing but our assumption (24) and thus

$$step(s,d) \overset{dom(b)}{\sim} step(t,d)$$

holds. In consequence, the induction hypothesis (28) can be applied and (26) is thus proved.
This completes the case A.

Let us consider now case B, i.e. we assume that $dom(d) \not\dashrightarrow dom(b)$ holds. In this case,

$run(t,purge(d \circ \delta,dom(b))) =$
$run(t, purge(\delta,dom(b)))$

We have to prove

$$run(step(s,d),\delta) \overset{dom(b)}{\sim} run(t, purge(\delta,dom(b))). \tag{30}$$

We can see that a particular case of the induction hypothesis (23), namely

$$run(step(s,d),\delta) \overset{dom(b)}{\sim} run(t,purge(\delta,dom(b))), \tag{31}$$

$$\text{if} \quad step(s,d) \overset{dom(b)}{\sim} t.$$

could be applied with the instantiation s' = $step(s,d)$ and t' = t, provided that

$$step(s,d) \overset{dom(b)}{\sim} t \tag{32}$$

holds. We have at our disposal the condition $s \overset{dom(b)}{\sim} t$ and we would like that (32) holds. Since M locally respects $\not\rightarrow$,

$$dom(d) \not\rightarrow dom(b) \Rightarrow s \overset{dom(b)}{\sim} step(s,d) \tag{33}$$

holds. $\overset{dom(b)}{\sim}$ is an equivalence relation and this means that from

$$s \overset{dom(b)}{\sim} t$$

and

$$s \overset{dom(b)}{\sim} step(s,d)$$

we have the truth of (32).

This completes a by-hand proof for the Unwinding Theorem.

The next section recalls the method for proving by induction atomic formulas called *Constructive Matching* (*CM*) introduced in [franova10]. This method is based on Cartesian Intuitionism presented in [franova64]. In Appendix we present the development of the proof for (20) by this method. Taking into account this machine directed proof we analyze below the necessary tools to enlarge *CMM* and compare it to the 'human-brain' proof presented in [rushby01].

IV. *CM*-formula construction

CM-formula construction guides the course of the inductive proofs of atomic formulas. Its particularity relies on the fact that it has been custom-designed to deal with Gödel's incompleteness results so that missing knowledge is suggested during the attempt to prove a formula in an incomplete environment. The same holds for suggesting missing lemmas, i.e. the knowledge explicitly missing to be able to complete the proof whereas it is implicitly present in the given axioms of the studied problem.

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 argument in the atomic formula that has to be proven true. We call them **pivotal arguments**, since the focus on them allows

- reducing what is usually called the search space of the proof, and
- decomposing 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 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. We do not propose yet a complete algorithmic solution to this problem, as it will be one of the last problems to be completely solved before implementing the whole *CMM*. This is due not only to the symbiotic character of developed tools but also because in our research we focus too on searching for problems that rise when an unsuitable argument is chosen. This way allows us to discover tools that are useful in synergistic approaches (see [franova08] and [bundy13]).

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 $F(t_1,t_2)$. We thus propose a 'detour' that will enable us to prove also the theorems that cannot be directly proven by the so-called simplification methods, i.e., without this 'detour'.

In the second step, via the definition of F and those involved in the formulation of the term t_1, we look for the features shown by all the ξ such that $F(t_1,\xi)$ is true. Given the axioms defining F and the functions occurring in t_1, we are able to obtain a set C_1 expressing the conditions on the set $\{ \xi \}$ 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 $C_2 = \{ \xi_C \mid cond_2(\xi_C)\}$. 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 case t_2 does not contain existential quantifiers, this is done by verifying $cond_2(t_2)$. In case t_2 contains existentially quantified variables, this is done by a 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 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, may be generated. *CMM* is conceived in such a way that the obtained sequence is well-behaving (see [franova10]) so that one can apply a generalization technique to obtain a more general formula that has to be proved. This formula covers logically the infinite sequence of lemmas and thus it fills the gap that cannot be overcome by purely deductive formal approach to theorem proving.

A detailed description of handling the pivotal argument in a rigorous framework can be found in [franova24] and [franova34]. A list of the most important achievements and experiences performed by *CMM* can be found in [franova64].

V. Necessary extension for *CMM*

The proof presented in Appendix shows that, in order to be able to be execute a proof for the Unwinding Theorem in a purely mechanized way, *CMM* needs to be extended by a non-inductive theorem proving mechanism, i.e. a theorem proving mechanism handling the predicates and the functions defined non-recursively. We intend to choose one from the already available ones. The Beth's tableaux method is one of the first logical mechanization of non-inductive proofs. This method serves as a logical justification (see [franova17]) of the basic non-inductive parts in *CMM*. Therefore, handling the implications in which the antecedent is a conjunction of atomic formulae with non-recursively defined predicates has already been dealt with (for more details see in [franova17]).

VI. Comparison with the proof presented by Rushby

There is a very important difference of our presentation of a non-clever by-hand proof of the Unwinding Theorem an the Rushby's proof [rushby01]. Taking into account the

assumptions of the Unwinding Theorem, the actual formula to be proved is (7), i.e.

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

However, Rushby switches directly to proving its generalization (20), i.e.

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

leaving thus to the reader to make explicit the reasoning steps leading to this generalization.

Our previous and current development of *CMM* is based exactly on formalizing algorithmically the invention process for the cases of missing knowledge. Indeed, the problem of missing knowledge is deeply linked to the incompleteness results formulated by Gödel [godel02]. We shall go further to this point in a few moments. Before that, let us say that we do not attempt to formalize the invention process of a genius or of a clever mathematician. We do follow Descartes' advice of finding once for all a purpose built systematic way of handling creativity issues related to the inductive theorem proving. Thus, our approach radically differs from the existing approaches to mechanizing inductive proofs. (The best known are the system ACL2 [boyer-moore09], the system RRL [kapur03], the system NuPRL [constable04], the Oyster-Clam system [bundy15], the extensions of ISABELLE [paulson03], the system COQ [paulin-mohring01] and Matita Proof Assistant [asperti01].) These approaches select already existing tools from those available in Computer Science and that were initially developed for other purposes. Then they seek for clever ways of putting these tools synergistically together. As it can be seen from our presentation of the *CM*-formula construction, this way of handling atomic formulas through pivotal argument is an original way of directing inductive proofs as well as of the process of a step-wise construction of missing or relevant knowledge whenever necessary.

Of course, due to the above mentioned Gödel's results, in formalizing the invention process, we cannot go further than suggesting such missing knowledge. It will always have to be a human expert of the problem deciding on relevant character of this suggested information, as we have already illustrated in the framework of robotics [franova-kooli01]. In other words, it is only such a human expert who can know about the intended interpretation of the problem on hand.

The above mentioned step-wise construction of the relevant knowledge can be perceived easily by noting two essential differences between Rushby's proof and that of ours.

First of all, the induction hypothesis (23) contains two new universally quantified variables s' and t', while for Rushby, the induction hypothesis does not contain new variables. In our proof (see going from (26) to (28)), the instantiation of these universally quantified variables leads to the condition (27) which has to be verified, hinting thus at the relevant information to be looked for in the given theory.

Since Rushby's proof is clever (due to the human behind the proof), the going from the formula 2.3. in [rushby01], i.e.

$$run(t, purge(a \circ \alpha,u)) = run(t, a \circ purge(\alpha,u))$$

to the successful conclusion of the case 1 of his proof is mathematically 'beautiful' in contrast to the 'cumbersome' presentation of our proof. In the vocabulary of mechanizing theorem proving it means that Rushby's cleverness reduces the search space while our method switches the focus from a search space to the direct 'hints' about the knowledge to be looked for in a given theory for any inductive theorem

proving problem, even those related to the program synthesis [franova64]. This means that *CMM* and Cartesian *Formal Creativity* [franova53] upon which *CMM* is being built are worth more than a simple inductive theorem proving strategies but they are suitable as a method for finding an axiomatic theory for problems that require recursive thinking and that, at the beginning, are specified only informally. We have illustrated this already in [franova-kooli01], but we would like to point out its importance and usefulness in the context of security of the information flow systems. This is why, in future, we intend to look more closely at the work of Mantel [mantel02] from the point of view examining the efficiency of our work and its eventual promoting as a tool to make the process of building formalized theories much more method oriented than genius dependent.

To our best knowledge there is no other existing work related to the automation of the inductive proofs for a large variety of already formulated Unwinding Theorems [graham-cumming01], [pinsky01], [haigh01], [millen01].

VII. Future work

The problem of information flow security is very interesting, important and challenging. As a complementary work to [hutter05], [hutter04], we plan to continue 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 [mantel02] and, among others, [graham-cumming01], [pinsky01], [haigh01], [millen01]. The challenge is there the execution proofs for theorems that contain existential quantifiers and that are related to the information flow systems security as well as handling non-transitive relations. We are quite sure that our future investigations concerning the use of our method 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. Note that tools we custom-create in *CMM* are symbiotic and thus the implementation can really start only after the research is completed. This is not usual in Computer Science today where the systems are built from synergistically interacting independent modules which of course provides a possibility of implementing partial results and allows linear developing of trust of peers. However, in the long term our approach will enable handling problems that are not possible to solve by synergistic tools. It will also provide inspiration for new real world applications simply by the presence of new powerful creation tools.

VIII. Conclusion

The automation of inductive theorem proving has been studied now for a long time. In contrast to traditional way of accepting Gödel's incompleteness results as a limitation, we used our interpretation of these results as an incentive for changing the "automation game" by

- switching
  - from the idea of once for all formally fixed theorem checking system developed via already existing tools
  - to the idea of an 'oscillating' system ([franova64]) taking into account the particular requirements related to the inductive proof at hand, and
- developing
  - not a toolbox of elements brought from other fields than inductive theorem proving framework
  - but an on-purpose inductive theorem proving framework tools solving

problems discovered in the oscillating evolution of the system.

This direction of development requires that we study inductive proving not only in its 'well-behaving' mathematical framework of natural numbers, lists and trees but also in challenging context of information flow security for proving Unwinding Theorems. In this paper we have illustrated how our approach discovers a suitable generalization of the Unwinding Theorem as it was initially formulated. The study in this paper shows that our *CMM* is relevant also in this context even if we have to consider a minor extension. *CMM* provides experts of information security flow a tool well-suited to handling creativity issues met while proving these complex theorems. In particular, some applications may require a change in the specification of the unwinding-type theorems. The present paper provides a detailed illustration on the way new specifications can be proven or, if necessary, completed or reformulated.

We thus aim at enlarging our *CMM* methodology in such a way that it could be used in testing their hypotheses while formulating the domain specification (i.e. the formal theory) for Unwinding Theorems specific to their particular problem. We already have illustrated this particular use of *CMM* in the context of robotics [franova-kooli01].

IX. Appendix

In this part we are going to present the induction step for a proof of (20) as performed following the *CM*-construction formula presented in Section 0. Nevertheless, we shall 'linearize' the previous notation. Namely, the binary relation 'u $\dashrightarrow$ v' will be denoted by intf(u,v) and consequently 'u $\not\dashrightarrow$ v' will be not(intf(u,v)). The equivalence relation 's $\overset{u}{\sim}$ t' will be denoted by EQ(u,s,t). Recall that EQ is an equivalence relation, i.e. it holds for all triple of variables p, q, r from the set of states.

(reflexivity)	EQ(p,p)
(symmetry)	EQ(p,q) $\Leftrightarrow$ EQ(q,p)
(transitivity)	EQ(p,q) & EQ(q,r) $\Rightarrow$ EQ(p,r)

We want to prove (20), i.e.,

$$EQ(u,s,t) \Rightarrow EQ(u,run(s,\alpha),run(t,purge(\alpha,u))). \quad (34)$$

where M locally respects EQ and is step consistent (see **Definition 4.**).

We use the rules of Beth's tableaux

- decompose the proof of this implication as a proof of an atomic formula (in the IVALID column – see [beth03]) and the all conditions in the VALID column)
- to any property of M introduce its particular case considered for any parameter introduced.

In the induction step we have $\alpha = c \circ \gamma$. Thus, s, t, c and γ are parameters for which the properties of M will be instantiated.

The goal is to prove

$$EQ(u,run(s,c \circ \gamma),run(t,purge(c \circ \gamma,u))) \quad (35)$$

assuming the antecedent of (34), i.e.

$$EQ(u,s,t) \quad (36)$$

as well as (since M locally respects EQ)

$$non(intf(dom(c),u)) \Rightarrow EQ(u,r,step(r,c))) \quad (37)$$

for any state r, and (since M is step consistent)

$$EQ(u,s,t) \Rightarrow EQ(u,step(s,c),step(t,c)) \quad (38)$$

and the induction hypothesis

$$EQ(u,s',t') \Rightarrow EQ(u,run(s',\gamma),run(t',purge(\gamma,u))) \quad (39)$$

for any s' and t'.

To be able to follow the steps as given in Section IV, it is enough to state t_1 = run(s,c ∘ γ) and t_2 = run(t,purge(c ∘ γ,u)).

The first step in our procedure is to replace an argument term of (35) by a so-called pivotal argument (see Section IV). Since the third term, i.e. t_2, is the most complex (as a tree), the third argument of (35) becomes the pivotal argument ξ.
We thus consider

$$EQ(u,run(s,c \circ \gamma),\xi). \quad (40)$$

By (ax2), this gives

$$EQ(u,run(step(s,c), \gamma),\xi). \quad (41)$$

The consequent of the induction hypothesis (39) can be applied via instantiation s' = step(s,c) and the transformation of ξ to run(t', purge(γ,u)) if the condition

$$EQ(u,run(step(s,c), \gamma),run(t', purge(\gamma,u))) \quad (42)$$

holds.
The question is whether the term run(t,purge(c ∘ γ,u)) can be transformed into run(t',purge(γ,u)). To check this, we have to evaluate run(t,purge(c ∘ γ,u)) first. The definition of pg gives two cases to be considered.

- (ax4) gives the case condition intf(dom(c),u) and
- (ax5) gives the case condition not(intf(dom(c),u))).

Let us consider case intf(dom(c),u).
Then, run(t,purge(c ∘ γ,u)) = run(t, c ∘ purge(γ,u)) = run(step(t,c),purge(γ,u)).
The question is thus whether run(step(t,c),purge(γ,u)) can be transformed into run(t',purge(γ,u)). This can be done with the instantiation t' = step(t,c) provided the corresponding condition (42) is verified. Thus we have to check

$$EQ(u,run(step(s,c)), run(step(t,c),purge(\gamma,u))). \quad (43)$$

Note that the predicate of this formula is not recursively defined and thus this condition is verified non-recursively in the context of the considered properties (here (38)) of M and the condition (36).
Thus this case is solved applying Modus Ponens to (36) and (38).

Let us consider case not(intf(dom(c),u)).
Then, run(t,purge(c ∘ γ,u)) = run(t,purge(γ,u)).
The question is thus whether run(t,purge(γ,u)) can be transformed into run(t',purge(γ,u)). This can be done with the substitution t' = t, provides the appropriate instantiation of the antecedent of the induction hypothesis (39) holds. I.e. if

$$EQ(u,run(step(s,c)),t) \quad (44)$$

holds. Once again, EQ is a non-recursively defined predicate, thus (44) is proved via traditional theorem proving from the condition (34) and instantiation s for r in (37) which gives EQ(u,s,step(s,c)).
But EQ is symmetric and thus EQ(u,s,t) gives EQ(u,t,s). Transitivity of EQ gives (from EQ(u,t,s) and EQ(u,s,step(s,c))) that EQ(t,step(s,c)) holds and by symmetry of EQ we have (44).

X. References

[asperti01] A. Asperti, C. S. Coen, E. Tassi, S. Zacchiroli: User Interaction with the Matita Proof Assistant; Journal of Automated Reasoning, August 2007, Volume 39, Issue 2, pp 109-139.

[beth03] E. Beth: The Foundations of Mathematics; Amsterdam, 1959.

[boyer-moore09] R. S. Boyer, J S. Moore: A Computational Logic Handbook; Academic Press, Inc., 1988.

[bundy13] A. Bundy : The Automation of Proof by Mathematical Induction, in: A. Robinson, A. Voronkov (eds.): Handbook of Automated Reasoning, vol. I, North-Holland, 2001, pp. 845-912.

[bundy15] A. Bundy, F. Van Harnelen, C. Horn, 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).

[constable04] R. L. Constable: Implementing Mathematics with the Nuprl Proof Development System; Prentice-Hall, Inc., Englewood Clifs, New Jersey, 1986.

[franova08] M. Franova: PRECOMAS Challenge; Rap. de Recherche No.376, L.R.I., Orsay, France, September 1987.

[franova10] 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; August, Los Angeles, 1985, 1214-1220.

[franova14] M. Franova: Why are we (almost always) able to prove inductive theorems "by hand" and how to obtain an automatic system that does it the same way: Introduction to inductive theorem proving for postgraduate students; Rapport de Recherche No. 327, L.R.I., Université de Paris-Sud, Orsay, France, January, 1987.

[franova17] 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.

[franova24] M. Franova: PRECOMAS - An Implementation of Constructive Matching Methodology; Proceedings of ISSAC'90 (Tokyo, Japan, August 20-24, 1990), ACM, New York, 1990, 16-23.

[franova34] M. Franova: Constructive Matching methodology and automatic plan-construction revisited; Rapport de Recherche No.874, L.R.I., Univ. de Paris-Sud, Orsay, France, November, 1993.

[franova53] M. Franova: Créativité Formelle: Méthode et Pratique - Conception des systèmes "informatiques" complexes et Brevet Épistémologique; Publibook, 2008.

[franova62] M. Franova, Y. Kodratoff: Choosing an induction variable in universally quantified atomic formulas; Rapport de Recherche No.1579, L.R.I., Université de Paris-Sud, Orsay, France, February 2015.

[franova64] M. Franova: Cartesian versus Newtonian Paradigms for Recursive Program Synthesis; International Journal on Advances in Systems and Measurements, vol. 7, no 3&4, 2014, pp. 209-222.

[franova-kooli01] M. Franova, 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, 1998, 836-841.

[godel02] 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, 592-618.

[graham-cumming01] J. Graham-Cumming, J.W. Sanders: On the refinement of Non-interference; Proc. of the IEEE Symposium on Security and Privacy, pp. 11-20, 1982.

[haigh01] J. T. Haigh, W. D. Young: Extending the Noninterference Version of MLS for SAT; IEEE Trans. Software Eng. 13(2), pp. 141-150, 1987.

[hutter05] D. Hutter: Automating Proofs of Unwinding Conditions; in: Serge Autexier, Heiko Mantel (eds.): Workshop Proceedings VERIFY06 at the International Joint Conference on Automated Reasoning, Seattle, 2006.

[hutter04] D. Hutter, H. Mantel, I. Schaefer, 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.

[kapur03] D. Kapur: An overview of Rewrite Rule Laboratory (RRL); J. Comput. Math. Appl. 29(2), 91–114 (1995).

[mantel02] H. Mantel: A Uniform Framework for the Formal Specification and Verification of Information Flow Security; PhD thesis, Universitty of Saarlandes, 2003.

[millen01] J. K. Millen: Unwinding Forward Correctability; Proc. of the 7th IEEE Computer Security Workshop, pp. 35-54, 1994.

[paulin-mohring01] 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, p. 607–640..

[paulson03] L. C. Paulson: The foundation of a generic theorem prover; Journal of Automated Reasoning, September 1989, Volume 5, Issue 3, pp 363-397.

[pinsky01] S. Pinsky: Absorbing Covers and Intransitive Non-Interference; Proceedings of IEEE Symposium on Security and Privacy, pp. 102 - 113, 1995.

[rushby01] J. Rushby: Noninterference, Transitivity, and Channel-Control Security Policies; Technical Report CSL-92-02, Computer Science Laboratory SRI International, December 1992.