Declarative Incorrectness Diagnosis of Constraint Logic Programs

François Le Berre and Alexandre Tessier
LIFO — Université d'Orléans
BP 6759, F-45067 Orléans Cedex 02, France
E-mail: {flb,tessier}@lifo.univ-orleans.fr

Mars 1995
RR-LIFO-95-8

Abstract

Our concern in this paper is the declarative debugging of Constraint Logic Programs in the framework of the CLP formalism. Our diagnosis is declarative, this means that there is no need for the programmer to understand the computational behaviour of the system. What is required is the declarative intended semantics of the program.

It is not possible to merely adapt to CLP diagnosis techniques of LP. So, new theoretical foundations are necessary.

The basic notion is the proof tree skeleton which clearly expresses the relation between declarative and operational semantics.

This paper is only devoted to errors which lead to wrong answers.

The main contribution of this paper is: if there exists a wrong answer then there is an incorrect rule in the program, and this rule occurs in the answer skeleton. Above all, there are new notions adapted to CLP framework. An algorithm is proposed to localize this incorrect rule.

1 Introduction

Our concern in this paper is the declarative debugging of Constraint Logic Programs in the framework of the CLP formalism as defined in [12, 13, 1]. Our diagnosis is declarative. In this context, this means that there is no need for the programmer to understand the computational behaviour of the system. What is required is the declarative intended semantics of the program.

One the one hand, one of the greatest strengths of Constraint Logic Programming is its declarative nature. On the other hand, real programming is not purely declarative and even if this ideal was achieved, it would be an illusion to believe that programs would be always corrects and that debugging tools would be useless.

Tracing techniques are useful, but in addition to their direct link to the computational behaviour, they quickly become extraordinarily difficult to use.

Moreover it is incoherent to use only low level debugging tools, based on an understanding of the operational behaviour of the system, and to give up a high level knowledge of the declarative meaning of the program.

For these reasons, we argue that an indispensable component of a complete CLP system is a declarative debugging system.

Declarative error diagnosis in Logic Programming was introduced by Shapiro [17], and called algorithmic debugging. Other techniques, developped in [15, 14, 7], are inspired from the Shapiro's method.

In this paper, our framework is CLP. In CLP the necessity of declarative debugging is as much significant as in Logic Programming. However, it is not possible to merely adapt LP techniques to CLP.

For example, in CLP, Herbrand interpretations do not represent program semantics. Moreover, in LP each answer which is a logical consequence of the program is covered by a more general computed answer. This is not true in CLP (although a sufficient condition for this covering property is the Independence of Negated Constraints [16]). So, new theoretical foundations are necessary.

Other work were concerned with declarative debugging for constraint programs, but in different frameworks. [9] considers concurrent constraint (cc) programs and is based on the concept of observable and specified behaviours of cc processes. [3] studies abstract debugging which is a combination of algorithmic debugging, s-semantics [2] and abstract interpretation [4].

Our approach of program semantics is based on an extension to CLP of the "Grammatical View" of LP introduced by Deransart and Maluszinski [5]. In fact, the basic notion is the proof tree skeleton. This notion clearly expresses the relation between declarative and operational semantics.

This paper is only devoted to errors which lead to wrong answers. In particular, errors leading to missing answers are not considered.

The main contribution of this paper is: if there exists a wrong answer then there is an incorrect rule in the program, and this rule occurs in the answer skeleton. Above all, there are new notions adapted to CLP framework. An algorithm is proposed to localize this incorrect rule.

The rest of the paper is organized as follows: section 2 gives the motivations for the formal notions. Section 3 formally defines these notions in accordance with an underlying pre-interpretation. Section 4 abstracts the pre-interpretation. In this framework, we give a diagnosis (error localization) algorithm. Section 5 concludes the paper.

2 Motivations

Let P be a CLP program and $\leftarrow g$ be a goal. The answer constraint r for g in P is considered abnormal if there is a valuation v in the underlying pre-interpretation $\mathcal{D}$ (interpretation of the constraints language) such that v satisfies r and $v(g)$ should not evaluate to *true* wrt the expected properties of P. We say that v is an *anomaly*.

The aim of this paper is to provide an assistance in order to localize a reason of the anomaly. The paper is only concentrated on incorrect answers wrt expected properties, and not on missing answers.

Example 1 *Fibonacci*

Let FIB be the program

$$\begin{array}{l} fib(0,0) \leftarrow true \\ fib(1,1) \leftarrow true \\ fib(x+1, y_1+y_2) \leftarrow x > 0 \, \Box \, fib(x,y_1), fib(x,y_2) \end{array}$$

The program language is defined from the symbols which occur in the program in addition to the function symbol $-$ (minus).
The underlying pre-interpretation for the constraints is $\mathcal{N}$, whose domain is $I\!N$, with the usual interpretation for the function symbols and the constraint predicate symbols.
The program defined predicate symbol fib is supposed to define the binary relation over $I\!N$ such that the second argument is the result of the fibonacci mapping (called $fibo$) applied to the first argument.
The answer constraint $x = 1+1 \wedge y = 1$ for the goal $\leftarrow y = 1 \, \Box \, fib(x, y+1)$ is abnormal. A valuation v_0 such that $v_0(x) = 2$ and $v_0(y) = 1$ satisfies the answer constraint but should not satisfy the goal.

We assume that the anomaly is not due to the system. We point out that errors cannot be caused by the possibly incompleteness of the constraint solver. In effect, such solvers possibly provide unsatisfiable answer constraint r, that is to say, always wrong in $\mathcal{D}$, thus, for each valuation v, $v(r) = false$, therefore v cannot be an anomaly for the goal g (r is not an abnormal answer wrt the expected properties).

The anomaly is due to P. P is wrong in the sense that P contains at least an incorrect rule. Our concern is to localize, as fast as possible, the faulty rule and the conditions of its abnormal behaviour. It is not necessary to review every rule of the program, but the matter is to examine the rules which have been used to construct the abnormal answer. These rules are those used in the success derivation which compute the answer from the goal.

We suggest to use a tool named *skeleton* which puts together the rules used by the derivation leaving aside the atom selection rule. The answers (success derivations) provided by a program are in fact independent of this atom selection rule.

An answer is a finite skeleton, rooted by the goal, not rejected in some sense. The skeleton contains the rules used along the derivation. From the skeleton we can infer the answer constraint, as well as, by giving an atom selection rule, the derivation. Skeletons are a good representation of the success derivations equivalence classes modulo the atom selection rule. They give a straightforward and intrinsic definition to the answers provided by the program.

Example 2 *Fibonacci*

The faulty rule is $fib(x+1, y_1+y_2) \leftarrow x > 0 \, \Box \, fib(x,y_1), fib(x,y_2)$. A possible patching is $fib(x+1, y_1+y_2) \leftarrow x > 0 \, \Box \, fib(x,y_1), fib(x-1,y_2)$.
The skeleton S_1 which provides the abnormal answer constraint is:

$$
\begin{array}{ccc}
S_1 = & \leftarrow y = 1 \;\Box\; fib(x, y+1) & \\
 & fib(x+1, y_1 + y_2) \leftarrow x > 0 \;\Box\; fib(x, y_1), fib(x, y_2) & \\
 & fib(1,1) \leftarrow true \qquad\qquad fib(1,1) \leftarrow true &
\end{array}
$$

An abnormal answer constraint comes from an answer skeleton. We have to traverse this skeleton to find the faulty rule. Moreover, we try to make clear the circumstances under which it is faulty. At first, they are formalized by a valuation, as shown in the following section.

3 From incorrectness $\mathcal{D}$-symptom and $\mathcal{D}$-incorrectness to incorrectness symptom and incorrectness

Let P be a program and $\leftarrow c_g \;\Box\; b_{g_1}, \cdots, b_{g_m}$ a goal. The answer constraint r is regarded as abnormal because in the underlying pre-interpretation $\mathcal{D}$ there is a valuation v which satisfies r and v is an anomaly.

Definition 3 *A* $\mathcal{D}$-atom *is a $(n+1)$-tuple $p(d_1, \cdots, d_n)$ where p is a n-ary predicate symbol definable by a program and $d_1, \cdots, d_n$ are elements of the domain D of $\mathcal{D}$.*

Definition 4 *A* $\mathcal{D}$-interpretation *I is a set of $\mathcal{D}$-atoms. It defines an interpretation.*

There is a natural extension of v denoted also by v which maps from terms, constraints and atoms.

Definition 5 *I is a* $\mathcal{D}$-model *of P if for each rule (head $\leftarrow$ c $\Box$ body) of P, for every valuation v,*

$$(v(c) = true \text{ and } v(body) \subseteq I) \Rightarrow (v(head) \in I)$$

Let $M_{\mathcal{D}}^P$ be the least $\mathcal{D}$-model of P.

If v is an anomaly then v satisfies c_g in $\mathcal{D}$ and $v(b_{g_i}) \in M_{\mathcal{D}}^P$, for each $i = 1, \cdots, m$.

Answer constraint r has the form of $\exists_{-\tilde{x}}(r' \wedge c_g)$, where $\tilde{x}$ are the free variables of the goal and $\exists_{-\tilde{x}}$ is the existential closure except on the variables $\tilde{x}$, which remain unquantified. If v satisfies r then v satisfies c_g. The anomaly lies in the fact that one atom b_{g_i} of the goal is such that $v(b_{g_i})$ should not be in $M_{\mathcal{D}}^P$.

When we wrote the program P we wanted that the relations defined by P be true in an intended $\mathcal{D}$-interpretation I. This $\mathcal{D}$-interpretation formalizes the intended semantics of the program P.

The anomaly is that $r \rightarrow c_g \wedge b_{g_1} \wedge \cdots \wedge b_{g_m}$ is not valid in I because $v(b_{g_i}) \notin I$. The anomaly exists because $M_{\mathcal{D}}^P \not\subseteq I$.

This motivates the following definitions.

Definition 6 *(partially correct wrt I, correct wrt I, incorrectness $\mathcal{D}$-symptom)*

1. *P is* partially correct *wrt I if $M^P_{\mathcal{D}} \subseteq I$.*

2. *P is* correct *wrt I if I is a $\mathcal{D}$-model of P, that is to say, every rule of P is valid in I.*

3. *An* incorrectness $\mathcal{D}$-symptom *of P wrt I is a $\mathcal{D}$-atom in $M^P_{\mathcal{D}}$ which is not in I.*

Thus, P is partially correct iff there is no incorrectness $\mathcal{D}$-symptom.

Lemma 7 *P is correct wrt $I \Rightarrow P$ is partially correct wrt I.*

Proof: $M^P_{\mathcal{D}}$ is the least $\mathcal{D}$-model of P. ■

Remark 8 The converse is wrong. (For example, let P be the program $\{p \leftarrow true \,\Box\, q\}$ and $I = \{q\}$. $M^P_{\mathcal{D}} = \emptyset \subseteq I$, thus P is partially correct wrt I. But I is not a $\mathcal{D}$-model of P, consequently P is not correct wrt I.)

If there is an incorrectness $\mathcal{D}$-symptom of P wrt I then P is not partially correct wrt I, thus P is not correct. Then there is a rule of P $head \leftarrow body$ which is not valid in I.

This rule can be viewed as an error which causes the symptom. $head \leftarrow body$ not valid in I signifies that there exists a valuation v which satisfies in I the *body* of the rule but not the *head*.

Definition 9 *A* $\mathcal{D}$-incorrectness *of P wrt I is a pair $\langle h \leftarrow c \,\Box\, b_1, \cdots, b_n \;, v\rangle$ such that $h \leftarrow c \,\Box\, b_1, \cdots, b_n$ is a rule of P, v is a valuation and*

- *$v(c) = true$,*
- *$v(b_i) \in I$, $i = 1, \cdots, n$,*
- *$v(h) \notin I$.*

Lemma 10 *(there is an incorrectness $\mathcal{D}$-symptom) $\Rightarrow$ (there is a $\mathcal{D}$-incorrectness)*

Proof: Corollary of lemma 7. ■

Example 11 *Fibonacci*

The intended semantics of FIB is formalized by the $\mathcal{N}$-interpretation

$$I_{FIB} = \{fib(d_1, d_2) \mid fibo(d_1) = d_2\}.$$

The least $\mathcal{N}$-model of FIB is

$$M^{FIB}_{\mathcal{N}} = \{fib(d_1, d_2) \mid \text{if } d_1 = 0 \text{ then } d_2 = 0 \text{ else } d_2 = 2^{d_1 - 1}\}.$$

FIB is not partially correct wrt I_{FIB} ($M^{FIB}_{\mathcal{N}} \not\subseteq I_{FIB}$), thus FIB is not correct wrt I_{FIB}.
The $\mathcal{N}$-atom $fib(2,2)$ is an incorrectness $\mathcal{N}$-symptom ($fibo(2) \neq 2$).
The pair $\langle fib(x+1, y_1 + y_2) \leftarrow x > 0 \,\Box\, fib(x, y_1), fib(x, y_2), v_1\rangle$, where v_1 is a valuation such that $v_1(x) = 1$, $v_1(y_1) = 1$ and $v_1(y_2) = 1$, is a $\mathcal{N}$-incorrectness of FIB wrt I_{FIB}.

The point of interest in CLP is that, usually, v cannot be expressed in the language because there is no ground term for each element of D.

The notion of $\mathcal{D}$-incorrectness is too strong. To remain in the language we propose to change the valuation for a constraint which approximates to the valuation in a sense. There is, at the moment, a notion able to suit.

Definition 12 *r is a* witness *(constraint) of the invalidity of the rule $h \leftarrow c \,\Box\, b_1, \cdots, b_n$ in the $\mathcal{D}$-interpretation I if there is a valuation v solution of r such that $v(c) = true$, for each $i = 1, \cdots, n$, $v(b_i) \in I$ but $v(h) \notin I$, i.e. $\models_I \tilde{\exists}(r \wedge c \wedge b_1 \wedge \cdots \wedge b_n \wedge \neg h)$, where $\tilde{\exists}$ is the existential closure.*

Remark 13 A rule is not valid in I iff there exists a witness (for example, the constraint *true*).

However some witnesses can be more interesting: if $\models_{\mathcal{D}} c \rightarrow c'$ then (c is a witness of the rule R) $\Rightarrow$ (c' is a witness of the rule R); c provides more information in the sense that c better approximates to v than c'.

There is, of course, a commonplace algorithm which consists in verifying every rule of the program, that is to say, for every rule *head* $\leftarrow$ *body* of the program, if $\models_I body \rightarrow head$.

We can restrict the research by considering the rules which take place in the computation of the answer constraint, that is to say, the rules which label the skeleton (defined below) associated to the answer constraint. In the skeleton there is a rule such that the existential closure, except on the free variables of the rule, of the answer constraint is a witness for this rule. This is more precise but it is just a rule checking again.

From this notion of witness, we can deduce an algorithm of rule checking which tests the validity in I of each rule. But this is too awkward. We would like to boil down to easier problems, no more focused on rules but on *constrained atoms*.

Definition 14 *r is a* strong witness *of the invalidity of the rule $h \leftarrow c \,\Box\, b_1, \cdots, b_n$ in the $\mathcal{D}$-interpretation I if $\models_I r \rightarrow c \wedge b_1 \wedge \cdots \wedge b_n$ and not($\models_I r \rightarrow h$).*

Remark 15 ($\models_I r \rightarrow c \wedge b_1 \wedge \cdots \wedge b_n$) $\Leftrightarrow$ ($\models_{\mathcal{D}} r \rightarrow c$ and for all $i \models_I r \rightarrow b_i$).

Remark 16 r is a strong witness $\Rightarrow$ r is a witness.
The converse is wrong, as it is shown in the following example.

Example 17 *Fibonacci*

The constraint *true* is a witness of the invalidity of the rule

$$fib(x+1, y_1+y_2) \leftarrow x > 0 \,\Box\, fib(x, y_1), fib(x, y_2)$$

because of the valuation v_1 ($v_1(x) = 1, v_1(y_1) = 1, v_1(y_2) = 1$).
But *true* is not a strong witness of the invalidity of the same rule because

$$\text{not}(\models_{I_{FIB}} true \rightarrow x > 0 \wedge fib(x, y_1) \wedge fib(x, y_2)).$$

A strong witness of the invalidity of this rule is $x = 1 \wedge y_1 = 1 \wedge y_2 = 1$ (which is also a witness).

This notion of strong witness motivates the following new definitions of incorrectness symptom and incorrectness. We will show that if there is an incorrectness symptom then there is a strong witness for a rule.

Definition 18 *An* incorrectness *of P wrt the $\mathcal{D}$-interpretation I is a pair $\langle h \leftarrow c \,\Box\, b_1, \cdots, b_n \;, r\rangle$, where r is a strong witness for the rule.*

Definition 19 *An* incorrectness symptom *of P wrt the $\mathcal{D}$-interpretation I is a pair $\langle \leftarrow c \,\Box\, b_1, \cdots, b_n \;, r\rangle$, such that $\models_{M_{\mathcal{D}}^P} r \rightarrow c \wedge b_1 \wedge \cdots \wedge b_n$ and not($\models_I r \rightarrow c \wedge b_1 \wedge \cdots \wedge b_n$).*

Since $\models_{\mathcal{D}} r \rightarrow c$ then $\models_I r \rightarrow c$, therefore there exists $i \in \{1, \cdots, n\}$ such that not($\models_I r \rightarrow b_i$).

An atomic incorrectness symptom *of P wrt I is a constrained atom $r \rightarrow b$ such that $\models_{M_{\mathcal{D}}^P} r \rightarrow b$ and not($\models_I r \rightarrow b$).*

Remark 20 (there is an incorrectness symptom of P wrt I) $\Leftrightarrow$ (there is an atomic incorrectness symptom of P wrt I).

Lemma 21 *(There is an incorrectness symptom of P wrt I) $\Rightarrow$ (there is an incorrectness of P wrt I).*

To prove this lemma and to define answers and answer constraints we use the skeletons, informally introduced in section 2. Let us define them more precisely.

Definition 22 *Let GL be the set of all goals. A* skeleton *is an oriented tree, labeled by $P \bigcup GL$, such that*

- *the degree of a node is the number of atoms in the body of its label,*
- *the root is the unique node labeled by an element of GL.*

The relation "to be a variant" is an equivalence relation over the set of rules. We assume that the node of a skeleton (except the root) is labeled by an equivalence class of rules denoted by a rule of P.

We define the *choice mapping* CM in the following way: for each node N labeled by C,

- if N is the root $CM(N) = C$,
- otherwise, we choose a rule $CM(N)$ in the equivalence class of C, such that, for each pair of distinct nodes N_1 and N_2, $CM(N_1)$ and $CM(N_2)$ have not shared variables.

A skeleton S and a choice mapping CM are assumed to be fixed.

For every node, if b is the i^{th} atom in the body of its label and N is its i^{th} child, we define $atom(N) = b$.

Let $p(\tilde{t}), q(\tilde{t}')$ be two atoms, we define the constraint $p(\tilde{t}) = q(\tilde{t}')$, where $\tilde{t}$ and $\tilde{t}'$ denote sequences of terms, as

- $false$, if $p \neq q$,
- $t_1 = t'_1 \wedge \cdots \wedge t_n = t'_n$, if $p = q$.

We associate to each node N of S the constraint defined by:

- if N is the root: the constraint of the goal,
- otherwise $CM(N) = h \leftarrow c \,\square\, \tilde{b}$: the constraint $c \wedge (h = atom(N))$.

We associate to the skeleton S the constraint system $const(S)$ defined by the collection of the constraints associated to every node of S. When S is finite, we denote by $C(S)$ the conjunction of the constraints of $const(S)$ and by $AC(S)$ the constraint $\exists_{-\tilde{x}} C(S)$, where $\tilde{x}$ are the free variables of the root of S.

An *answer* for the goal $\leftarrow g$ is a finite skeleton rooted by $\leftarrow g$ such that $\models_{\mathcal{D}} \tilde{\exists} C(S)$, where $\tilde{\exists}$ denotes the existential closure. $AC(S)$ is an *answer constraint* for the goal $\leftarrow g$.

Example 23 *Fibonacci*

The following skeleton S_2 is an answer for the goal $\leftarrow true \,\square\, fib(x,y)$ (for further simplicity, nodes are labeled by renamed rules according to CM).

$$
\begin{array}{lccc}
S2 = & & \leftarrow true \,\square\, fib(x,y) & \\
 & & | & \\
 & & fib(x_1+1, y_1+y_2) \leftarrow x_1 > 0 \,\square\, fib(x_1,y_1), fib(x_1,y_2) & \\
 & \swarrow & & \searrow \\
 & fib(1,1) \leftarrow true & & fib(1,1) \leftarrow true
\end{array}
$$

The constraint associated to S_2 is $C(S_2) =$
$true \wedge x_1 > 0 \wedge x_1 + 1 = x \wedge y_1 + y_2 = y \wedge true \wedge 1 = x_1 \wedge 1 = y_1 \wedge true \wedge 1 = x_1 \wedge 1 = y_2$
which is satisfiable in $\mathcal{N}$.
The answer constraint is $AC(S_2) =$
$\exists x_1 \exists y_1 \exists y_2 (true \wedge x_1 > 0 \wedge x_1 + 1 = x \wedge y_1 + y_2 = y \wedge true \wedge 1 = x_1 \wedge 1 = y_1 \wedge true \wedge 1 = x_1 \wedge 1 = y_2)$
which is equivalent to $x = 1 + 1 \wedge y = 1 + 1$.

The pair $\langle fib(x+1, y_1+y_2) \leftarrow x > 0 \,\square\, fib(x_1,y_1), fib(x_2,y_2)\ , x = 1 \wedge y_1 = 1 \wedge y_2 = 1\rangle$ is an incorrectness of FIB wrt I_{FIB}.
The pair $\langle \leftarrow true \,\square\, fib(x,y)\ , x = 1+1 \wedge y = 1+1\rangle$ is an incorrectness symptom of FIB wrt I_{FIB}.
The constrained atom $(x = 1+1 \wedge y = 1+1) \rightarrow fib(x,y)$ is an atomic incorrectness symptom of FIB wrt I_{FIB}.

The following skeleton S_3 is not an answer for the goal $\leftarrow true \,\square\, fib(x,y)$ because $C(S_3) =$
$true \wedge x_1 > 0 \wedge x_1 + 1 = x \wedge y_1 + y_2 = y \wedge true \wedge 1 = x_1 \wedge 1 = y_1 \wedge true \wedge 0 = x_1 \wedge 0 = y_2$
is such that not$(\models_{\mathcal{N}} \exists x \exists y \exists x_1 \exists y_1 \exists y_2 C(S_3))$.

$$S3 = \qquad \leftarrow true \;\Box\; fib(x,y)$$

$$fib(x_1+1, y_1+y_2) \leftarrow x_1 > 0 \;\Box\; fib(x_1,y_1), fib(x_1,y_2)$$

$$fib(1,1) \leftarrow true \qquad\qquad fib(0,0) \leftarrow true$$

The notion of answer constraint is equivalent to the one of [13], as it is shown by the following lemma:

Lemma 24 *(correctness and completeness)*

1. *If S is an answer for the goal $\leftarrow g$ then $P \models_{\mathcal{D}} AC(S) \rightarrow g$.*

2. *If $P \models_{\mathcal{D}} c \rightarrow g$ then $\models_{\mathcal{D}} c \rightarrow \bigvee_{S \in R} AC(S)$, where R is the set of answers for the goal $\leftarrow g$.*

Proof: See [1]. ■

Remark 25 R may be infinite. It is not always possible to replace it by a finite subset. This is possible if $\mathcal{D}$ is replaced by a theory $\mathcal{T}$, a model of which is $\mathcal{D}$, by using the finitness theorem of first order logic.

Definition 26 *A* computed incorrectness symptom *of P wrt the $\mathcal{D}$-interpretation I is a pair $\langle \leftarrow c \;\Box\; b_1, \cdots, b_n \;, r\rangle$, such that r is an answer constraint for $\leftarrow c \;\Box\; b_1, \cdots, b_n$ and not($\models_I r \rightarrow c \wedge b_1 \wedge \cdots \wedge b_n$).*

Remark 27 (there is a computed incorrectness symptom of P wrt I) $\Rightarrow$ (there is an atomic incorrectness symptom of P wrt I).

Lemma 28 *(there is a computed incorrectness symptom of P wrt I) $\Leftrightarrow$ (there is an incorrectness symptom of P wrt I).*

Proof:

$\Rightarrow$ if $\langle \leftarrow c \;\Box\; b_1, \cdots, b_n \;, r\rangle$ is a computed incorrectness symptom of P wrt I then r is an answer constraint for the goal $\leftarrow c \;\Box\; b_1, \cdots, b_n$.

The first part of the lemma 24 shows that $P \models_{\mathcal{D}} r \rightarrow c \wedge b_1 \wedge \cdots \wedge b_n$, thus $\models_{M_{\mathcal{D}}^P} r \rightarrow c \wedge b_1 \wedge \cdots \wedge b_n$ (because $M_{\mathcal{D}}^P$ is the least $\mathcal{D}$-model of P), therefore $\langle \leftarrow c \;\Box\; b_1, \cdots, b_n \;, r\rangle$ is an incorrectness symptom of P wrt I.

$\Leftarrow$ if $\langle \leftarrow c \;\Box\; b_1, \cdots, b_n \;, r\rangle$ is an incorrectness symptom of P wrt I then $\models_{M_{\mathcal{D}}^P} r \rightarrow c \wedge b_1 \wedge \cdots \wedge b_n$, i.e. $r \rightarrow c \wedge b_1 \wedge \cdots \wedge b_n$ evaluates to $true$ in every $\mathcal{D}$-model of P, thus $P \models_{\mathcal{D}} r \rightarrow c \wedge b_1 \wedge \cdots \wedge b_n$.

The second part of the lemma 24 shows that $\models_{\mathcal{D}} r \rightarrow \bigvee_{S \in R} AC(S)$, where R is the set of the answers for the goal $\leftarrow c \;\Box\; b_1, \cdots, b_n$. This means that for every valuation v such that $v(c) = true$ there exists $S \in R$ such that $v(AC(S)) = true$.

We know that not($\models_I r \rightarrow c \wedge b_1 \wedge \cdots \wedge b_n$), thus there is a valuation v such that $v(r) = true$, therefore $v(c) = true$, and $\{v(b_1), \cdots, v(b_n)\} \not\subseteq I$.

Let $S \in R$ be a finite skeleton such that $v(AC(S)) = true$ then $\langle \leftarrow c \,\Box\, b_1, \cdots, b_n \ , AC(S)\rangle$ is a computed incorrectness symptom of P wrt I.

We have shown that (there is an incorrectness symptom $\langle \leftarrow g\ , r\rangle$ of P wrt I) $\Leftrightarrow$ (there is an answer S for $\leftarrow g$ such that $\langle \leftarrow g\ , AC(S)\rangle$ is a computed incorrectness symptom of P wrt I). ■

Lemma 29 *(There is a computed incorrectness symptom of P wrt I) $\Rightarrow$ (there is an incorrectness of P wrt I)*

Proof: Let $\langle \leftarrow c_g \,\Box\, b_{g_1}, \cdots, b_{g_m} \ , AC(S)\rangle$ be a computed incorrectness symptom of P wrt I.
Remark: S is a finite skeleton.
Remark: $\langle \leftarrow c_g \,\Box\, b_{g_1}, \cdots, b_{g_m} \ , C(S)\rangle$ is an incorrectness symptom of P wrt I.
We will show that $C(S)$ is a strong witness for a rule which labels a node of S.
Let us assume that $C(S)$ is not a strong witness for any rule of S, then we will show by induction on the subtree height of S that for every node N of S, except the root, $\models_I C(S) \rightarrow atom(N)$, i.e. $C(S) \rightarrow atom(N)$ is not an atomic incorrectness symptom of P wrt I, thus $\langle \leftarrow c_g \,\Box\, b_{g_1}, \cdots, b_{g_m} \ , C(S)\rangle$ is not an incorrectness symptom of P wrt I.
We remind that, because of the definition of $C(S)$, $\models_{\mathcal{D}} C(S) \rightarrow c_g$ and for every node N of S, except the root, labeled by a rule $h_N \leftarrow c_N \,\Box\, b_{N_1}, \cdots, b_{N_k}$,

- $\models_{\mathcal{D}} C(S) \rightarrow c_N$,
- ($\models_I C(S) \rightarrow h_N$) $\Leftrightarrow$ ($\models_I C(S) \rightarrow atom(N)$), because $const(S)$ contains $h_N = atom(N)$.

We denote by $S(N)$ the subtree of S whose root is N.

1. If $S(N)$ is 1 high then N is a leaf labeled by the fact $h_N \leftarrow c_N$ and $\models_I C(S) \rightarrow h_N$, therefore $\models_I C(S) \rightarrow atom(N)$,
2. if $S(N)$ is a high, $a > 1$, and for every child N' of N $\models_I C(S) \rightarrow atom(N')$,

   let $h_N \leftarrow c_N \,\Box\, b_{N_1}, \cdots, b_{N_k}$ be the label of N,

   then $\models_I C(S) \rightarrow c_N \wedge b_{N_1} \wedge \cdots \wedge b_{N_k}$, thus $\models_I C(S) \rightarrow h_N$, therefore $\models_I C(S) \rightarrow atom(N)$.

Consequently, every child N' of the root is such that $\models_I C(S) \rightarrow atom(N')$, thus $\models_I C(S) \rightarrow c_g \wedge b_{g_1} \wedge \cdots \wedge b_{g_m}$ therefore $\langle \leftarrow c_g \,\Box\, b_{g_1}, \cdots, b_{g_m} \ , C(S)\rangle$ is not an incorrectness symptom of P wrt I.
We have shown that
($\langle \leftarrow c_g \,\Box\, b_{g_1}, \cdots, b_{g_m} \ , C(S)\rangle$ is an incorrectness symptom of P wrt I) $\Rightarrow$ (there is a rule $h_N \leftarrow c_N \,\Box\, b_{N_1}, \cdots, b_{N_k}$ which labels a node of S such that $\langle h_N \leftarrow c_N \,\Box\, b_{N_1}, \cdots, b_{N_k} \ , C(S)\rangle$ is an incorrectness of P wrt I). ■

Proof of lemma 21 It is a consequence of lemmas 28 and 29. ■

Remark 30 If S is an answer for the goal $\leftarrow g$ and $\langle \leftarrow g, AC(S) \rangle$ is an incorrectness symptom of P wrt I then there is *in* S a rule $head \leftarrow body$, whose free variables are $\tilde{y}$, such that $\langle head \leftarrow body, \exists_{-\tilde{y}} C(S) \rangle$ is an incorrectness of P wrt I.

Remark 31 Particular case of logic programs.
In the domain $\mathcal{H}$ of the herbrand terms, the constraints which represent the substitutions have the Independence of Negated Constraints property [16].
A consequence of the second part of the lemma 24 and the Independence of Negated Constraints is:

if $P \models_{\mathcal{H}} c \rightarrow g$ then $\models_{\mathcal{H}} c \rightarrow r$, where r is an answer constraint for the goal $\leftarrow g$.

This result is known in Logic Programming [15] as:

if $P \models g\theta$ then there is a computed answer σ for $\leftarrow g$ which is more general than θ.

Proof: See [1]. ■

4 A theoretical algorithm for declarative incorrectness diagnosis

In this section, we introduce a novel program semantics which is based on the real constraint solver rather than on the underlying pre-interpretation. It is well-known that, in general, the constraint solvers are not satisfaction complete [12, 13]. Furthermore, they do not have the same behaviour that a theory. For a given constraint solver, there is not always a theory $\mathcal{T}$ such that $\mathcal{T} \models \neg c$ iff the constraint solver answers `no` for c.

Example 32 *CLP($\mathcal{R}$)*

The CLP($\mathcal{R}$) [11] constraint solver provides three kinds of answer: `yes`, `maybe` and `no`. In accordance to the underlying theory or pre-interpretation, if the CLP($\mathcal{R}$) constraint solver answers `yes` then the constraint is satisfiable and if it answers `no` then the constraint is unsatisfiable. But for the answer `maybe` the constraint is sometimes satisfiable, sometimes unsatisfiable.
The CLP($\mathcal{R}$) constraint solver answers `yes` for $x * x = 1 \wedge x = 1$ but `maybe` for $x * x = 1$, nevertheless $\models \exists x(x * x = 1 \wedge x = 1) \rightarrow \exists x(x * x = 1)$.
The CLP($\mathcal{R}$) constraint solver answers `no` for $x = 1 \wedge x = 0$ but `maybe` for $x*x = 1 \wedge x*x = 0$, nevertheless $\models \neg\exists x(x = 1 \wedge x = 0) \rightarrow \neg\exists x(x * x = 1 \wedge x * x = 0)$.

In section 3, the program semantics was formalized by the least $\mathcal{D}$-model of P. A finite skeleton S was rejected if $\tilde{\exists} C(S)$ was unsatisfiable in $\mathcal{D}$. For the system, a finite skeleton S is rejected if the constraint solver answers `no` for $C(S)$.

We want to abstract the pre-interpretation $\mathcal{D}$ by a *Reject Criterion RC* [1]. *RC* can be defined

1. wrt a pre-interpretation $\mathcal{D}$ (denoted by $RC(\mathcal{D})$): the constraint c is rejected iff c is unsatisfiable in $\mathcal{D}$,

2. wrt a theory $\mathcal{T}$ (denoted by $RC(\mathcal{T})$): c is rejected iff $\mathcal{T} \models \neg c$,

3. wrt a constraint solver $\mathcal{A}$ (denoted by $RC(\mathcal{A})$): c is rejected iff $\mathcal{A}$ answers `no` for c.

We denote by $RC_\emptyset$ the reject criterion which accepts all the constraints.

The reject criterion is assumed to be monotonic, i.e. if the collection of constraint C_1 contains the collection C_2 then (C_2 is rejected) $\Rightarrow$ (C_1 is rejected).

A finite skeleton S is not rejected by the reject criterion RC iff $C(S)$ is not rejected by RC.

Definition 33 *An answer according to RC for the goal $\leftarrow g$ is a finite skeleton, rooted by $\leftarrow g$, not rejected by RC.*

Then $AC(S)$ is an answer constraint according to RC for the goal $\leftarrow g$.

Definition 34 *A general answer for a goal $\leftarrow g$ is a finite skeleton rooted by $\leftarrow g$ (that is to say, an answer according to $RC_\emptyset$ for $\leftarrow g$).*

Remark 35 The lemma 24 can be extended to

1. If S is a general answer for the goal $\leftarrow g$ then $P \models AC(S) \rightarrow g$.

2. If $P \models c \rightarrow g$ then $\models c \rightarrow \bigvee_{S \in GR} AC(S)$, where GR is the collection of the general answers for the goal $\leftarrow g$.

3. If $P \models c \rightarrow g$ then $\models c \rightarrow \bigvee_{S \in GR_f} AC(S)$, where GR_f is a finite subset of the general answers for the goal $\leftarrow g$.

Proof: See [1]. ∎

As we abstract the underlying pre-interpretation $\mathcal{D}$, we can abstract the intended $\mathcal{D}$-interpretation I. Indeed, the notions of incorrectness symptom and incorrectness are defined from the validity of a constrained atom in I. That is to say, the interaction with the intended semantics is only based on constrained atoms. Therefore, the intended semantics can be formalized by a set of constrained atoms. We propose to use the notion of oracle $\mathcal{O}$ to formalize the intended semantics [17, 7, 15, 8]. $\mathcal{O}$ accepts or rejects constrained atoms in accordance with the intended semantics.

Remark 36 An oracle $\mathcal{O}$ is such that

$$\begin{array}{c} \mathcal{O} \text{ rejects } c \rightarrow b_1, \cdots, b_n \\ \text{iff} \\ \text{there exists } i \in \{1, \cdots, n\} \text{ such that } \mathcal{O} \text{ rejects } c \rightarrow b_i \\ \text{iff} \\ \text{there exists } i \in \{1, \cdots, n\} \text{ such that } \mathcal{O} \text{ rejects } (\exists_{-\tilde{y}_i} c) \rightarrow b_i, \\ \text{where } \tilde{y}_i \text{ are the free variables of } b_i. \end{array}$$

This section is more general than the previous, in the sense that we make two abstractions. $\mathcal{D}$ is abstracted by RC and I is abstracted by $\mathcal{O}$.

Definition 37 *An* incorrectness *of P wrt $\mathcal{O}$ is a pair $\langle h \leftarrow c \,\square\, b_1, \cdots, b_n \,, r\rangle$ such that for each $i \in \{1, \cdots, n\}$, $(r \wedge c) \rightarrow b_i$ is not rejected by $\mathcal{O}$ and $(r \wedge c) \rightarrow h$ is rejected by $\mathcal{O}$.*

Definition 38 *A* computed incorrectness RC-symptom *of P wrt $\mathcal{O}$ is a pair $\langle \leftarrow c \,\square\, b_1, \cdots, b_n \,, r\rangle$ such that r is an answer constraint according to RC for the goal $\leftarrow c \,\square\, b_1, \cdots, b_n$ and there exists $i \in \{1, \cdots, n\}$ such that $r \rightarrow b_i$ is rejected by $\mathcal{O}$.*

Lemma 39 *(There is a computed incorrectness RC-symptom of P wrt $\mathcal{O}$) $\Rightarrow$ (there is an incorrectness of P wrt $\mathcal{O}$).*

Remark 40 If $\langle \leftarrow c \,\square\, b_1, \cdots, b_n \,, r\rangle$ is a computed incorrectness RC-symptom of P wrt $\mathcal{O}$ then there is a finite skeleton S not rejected by RC, rooted by $\leftarrow c \,\square\, b_1, \cdots, b_n$, such that $AC(S) = r$ and there is a renamed rule according to CM in S $head \leftarrow body$ such that $\langle head \leftarrow body, r\rangle$ is an incorrectness of P wrt $\mathcal{O}$.

Proof: The prooves are similar to those of section 3. ■

We propose now a diagnosis algorithm. There is, of course, many other algorithms. The proposed algorithm travels down the skeleton along a branch from the goal to the incorrect rule, asking questions to the oracle. The oracle can be a human, but also an automatic system based on a partial specification of the intended semantics [6, 17].

We emphasize on the fact that our algorithm turns account the constraint solver of the system. Usually, the CLP system presents the answer constraints in a simplified form. We want to question the oracle with simplified constrained atoms. In place of questioning the oracle on the constrained atom $r \rightarrow b$, we question it on $(\exists_{-\tilde{y}} r) \rightarrow b$, where the constraint $\exists_{-\tilde{y}} r$ is simplified as well as the answer constraints. Then determining if r is a correct answer constraint for the goal $\leftarrow c \,\square\, b_1, \cdots, b_n$ is not easier than determining if $(\exists_{-\tilde{y}} r) \rightarrow b$ is in the intended semantics. The constraint solver of the system is invoked in order to simplify the constraint $\exists_{-\tilde{y}} r$.

Let us give the algorithm.

We denote by

`C(S)`	the constraint associated to the skeleton `S`
`root(S)`	the root of the skeleton `S`
`child(S,N,I)`	the `I`th child of node `N` in the skeleton `S`
`label(S,N)`	the label of node `N` in the skeleton `S`
`arity(S,N)`	the number of children of node `N` in the skeleton `S`
`i-atom(S,N,I)`	the `I`th atom in the body of the label of the node `N` in the skeleton `S`
`simplify(C,V)`	the simplification of the constraint $\exists_{-\mathtt{V}}\mathtt{C}$
`var(X)`	the free variables of the rule (or the atom) `X`
`ask(C --> B)`	the questioning of the oracle for the constrained atom $\mathtt{C} \rightarrow \mathtt{B}$

The function `simplify(C,V)` calls the constraint solver of the system.

```
function IncorrectNode(S) return node
begin
    N := root(S)
    loop
       I := 0
       loop
          I := I + 1
          if I > arity(S,N) then return N
          B := i-atom(S,N,I)
          ask(simplify(C(S),var(B)) --> B)
          exit when answer is NO
       endloop
       N := child(S,N,I)
    endloop
end

N := IncorrectNode(S)
write(simplify(C(S),var(label(S,N))) --> label(S,N))
```

5 Conclusion

We have formally defined the notions of incorrectness symptom and incorrectness for constraint programs in terms of constrained atoms. In Logic Programming abnormal valuations are always expressible in the program language. It is not the case in Constraint Logic Programming, where valuated atoms are replaced by constrained atoms.

We have proved that the existence of an incorrectness symptom implies that of an incorrectness. This incorrectness can be localized in the skeleton associated to the incorrectness symptom.

This paper is only devoted to errors which lead to wrong answers. It remains to study in the same way the errors leading to missing answers.

We have shown that there is a diagnosis algorithm. It remains to find efficient algorithms, in particular with respect to the number of questionings to the oracle.

It remains also to implement these algorithms. This raises meta-programming problems in CLP.

References

[1] M. Bergère, G. Ferrand, F. Le Berre, B. Malfon, A. Tessier, La Programmation Logique avec Contraintes revisitée en termes d'Arbres de Preuve et de Squelettes, Technical report, University of Orléans, LIFO 95-06, 1995.

[2] A. Bossi, M. Gabbrielli, G. Levi, M. Martelli, The s-semantics approach : Theory and applications, in Journal of Logic Programming, Vol. 19-20, pp 149-147, 1994.

[3] M. Comini, G. Levi, G .Vitiello, Abstract Debugging of Logic Programs, in L. Fribourg and F. Turini, editors, Proc. Logic Program Synthesis and Transformation and Metaprogramming in Logic 1994, LNCS 883, pp 440-450, Springer-Verlag, 1994.

[4] P. Cousot, R. Cousot, Abstract Interpretation and Applications to Logic Programs, Journal of Logic Programming, 19-20, pp 351-384, 1994.

[5] P. Deransart, J. Maluszinski, A Grammatical View of Logic Programming, MIT Press, 1993.

[6] W. Drabent, S. Nadjm-Tehrani, J. Maluszynski, Algorithmic Debugging with Assertions, META'88, pp 365-378, 1988.

[7] G. Ferrand, Error Diagnosis in Logic Programming: an adaptation of E. Y. Shapiro's method, Journal of Logic Programming 4, pp 177-198, 1987.

[8] G. Ferrand, The Notions of Symptom and Error in Declarative Diagnosis of Logic Programs, AADEBUG'93, LNCS 749, pp 40-57, Springer-Verlag, 1993.

[9] M. Fromherz, Towards Declarative Debugging of Concurrent Contraint Programs, AADEBUG'93, LNCS 749, pp 88-100, Springer-Verlag, 1993.

[10] M. Gabrielli, G. Levi, Modeling answer constraints in constraint logic programming, ICLP'91, pp 238-252, (V. J. Saraswat, K. Ueda ed.) MIT Press, 1991.

[11] N. Heintze, N. Jaffar, S. Michaylov, P. Stuckey, R. Yap, The CLP(R) programmer's manual version 1.2, IBM T. J. Watson Research Center.

[12] J. Jaffar, J. L. Lassez, Constraint Logic Programming, Technical Report 86/73, Department of Computer Science, Monash University, 1986 (and 14^{th} ACM Symposium on Principles of Programming Languages, Munich, 1987).

[13] J. Jaffar, M. Maher, Constraint Logic Programming: a survey, in Journal of Logic Programming, Vol. 19-20, pp 503-581, 1994.

[14] J. W. Lloyd, Foundations of Logic Programming, Springer-Verlag, 2^{nd} ed., 1987.

[15] J. W. Lloyd, Declarative Error Diagnosis, New Generation Computing 5, pp 133-154, 1987.

[16] M. J. Maher, A Logic Programming view of CLP, ICLP'93, pp 737-753, Warren Ed., 1993.

[17] E. Y. Shapiro, Algorithmic Program Debugging, ACM Distinguished Dissertation Series, MIT Press, 1982.