The Unfolding Semantics for Constraint Logic Programs

Roberto Giacobazzi

Dipartimento di Informatica
Università di Pisa
Corso Italia 40, 56125 Pisa
giaco@dipisa.di.unipi.it

1 Introduction

Pure logic programming is based on the unification algorithm on the Herbrand universe. The unification of a set of equations consists two tasks: deciding if the equations are solvable and simplifying the equations by transforming them into the solved form. The second aspect of this process is not really essential to computing.

We can be interested in extending the domain of terms in order to deal with more appropriate domains of computation, like the uncountable domain of reals. The framework of CLP(X) [5] was proposed to cope with this problem. It leads to the definition of a class of languages based upon constraint solving and the logic-programming paradigm. Each instance of the CLP(X) scheme is obtained by the specifying the computation structure and corresponding solution algorithm.

One of the most interesting aspects of this framework is that it allows to generate a class of languages which are soundly based within a single semantic structure. This structure depends on the intended domain of the computation. This contrasts with working on the Herbrand universe where every semantic object has to be coded into a Herbrand term.

This semantic feature leads to a closer relation between the semantics of the language and the real world of computation. As a consequence, the CLP(X)-scheme provides languages having a superior expressive power. In fact, since the notion of unifiability of a set of equations is a particular instance of constraint solvability, pure logic programming is an instance of CLP. The outstanding aspect of this class of languages is that they have the model-theoretic and fixpoint semantics which are similar to those of pure logic programs [5]. Moreover, since the operational semantics of CLP(X) is always based on constraint solvability on a given structure, it is possible to define an algebraic semantics too, corresponding to the algebraic structure of X.

From a purely theoretical point of view, as in the case of positive logic programs, the declarative semantics of CLP(X) enjoys nice properties such as the equivalence between the three notions of minimal model, success set and least fixpoint of an immediate consequence operator. However, this equivalence is not fully achieved, since the standard operational model of a CLP program does not mirror its observable properties, i.e. the real program behaviour, given in terms of a set of answer constraints, is not reflected in the minimal model. As argued in [7], the key issue consists in a suitable choice of the notion of model. In [4] we can find a reconstruction of the semantics of CLP, obtained by defining various model notions, each corresponding to a specific operationally observable property, as firstly defined in [3] for the pure logic programming case.

In this paper, we extend some results related to the pure logic programming and presented in [8], dealing with an alternative way to characterize the operational behaviour of logic programs. The main idea is that we can consider a possibly infinite collection of unit clauses as an alternative representation of the model. This set is obtained from the original program, via a program transformation technique. Therefore, the declarative meaning of a CLP program is again a possibly infinite CLP program.

The computed answer constraint can be obtained by executing the goal into this set of clauses. The operational semantics is reflected into a program transformation technique which preserves the intended meaning of the program. As stressed in [1] an unfolding transformation technique, appears as intermediary between procedural and fixpoint semantics. They show that it is possible to give a quasi-algebraic characterization of the relationship between these two semantics, which allows to give more simple proofs than usual, (in particular about soundness and completeness theorems). These thoughts increase the importance of such a transformation, both from a purely technical point of view and for a deeper understanding of the logic programs semantics. In particular the immediate consequence operator T_P can be defined in terms of the unfolding rule.

The unfolding rule presented in this paper is an extension, in the area of constraint logic programs, of the one presented in [8].

2 Preliminaries

Let us recall some basic concepts about CLP. For a more complete treatment on the subject see [6, 5].

In Constraint Logic Programming, the basic operational step is constraint solvability on a given domain. It follows that the algebraic approach to the semantics is natural for CLP.

Let us consider a many sorted first order language. Let $SORT = \bigcup_i SORT_i$ denotes a finite set of sorts. The signature of an n-ary function, predicate or variable is a sequence of $n+1$ elements of SORT.

Let us also denote by:

- Σ, a denumerable collection of function symbols,
- Π, a denumerable collection of predicate symbols, where Π contains the equality symbol.
- V, a denumerable set of variable symbols.

Let $\tau(\Sigma)$ and $\tau(\Sigma \cup V)$ denote, the ground and the possibly non ground class of terms respectively. Then [6, 5]:

- a (Σ, Π)-atom has the form: $p(t_1, ..., t_n)$ where $p \in \Pi$ and $t_i \in \tau(\Sigma \cup V)$, $\forall i = 1, ..., n$.
- a (Σ, Π)-constraint is a possibly empty, finite or infinite conjunction of (Σ, Π)-atoms. The empty constraint is denoted by TRUE.

Let us assume that $\Pi = \Pi_C \cup \Pi_A$ and $\Pi_C \cap \Pi_A = \emptyset$.

Definition 2.1 *A constraint logic program (CLP) is a finite set of constraint rules of the following form:*
$A \leftarrow c\Box.$ *or*
$A \leftarrow c\Box B_1, ..., B_n.$
where c is a finite (Σ, Π_C)-constraint and A (the head), $B_1, ..., B_n$ (the body) are (Σ, Π_A)-atoms.
A goal *is a constraint rule with no head and with a non empty body.*

The structure defined on Π and Σ (the alphabet), will be denoted by $\Re(\Sigma, \Pi)$ and consists of

1. a collection of non-empty sets $D\Re_s$, $s \in SORT$;

2. an assignment of a function $D\Re_{s_1} \times ... \times D\Re_{s_n} \rightarrow D\Re_s$ to each n-ary function symbol $f \in \Sigma$, where $(s_1, ..., s_n, s)$ is the signature of f;

3. an assignment of a function $D\Re_{s_1} \times ... \times D\Re_{s_n} \rightarrow \{TRUE, FALSE\}$ to each n-ary predicate symbol $p \in \Pi$ (apart from $=$ which is naturally interpreted as syntactic equality), where $(s_1, ..., s_n)$ is the signature of p.

A $\Re(\Sigma, \Pi)$-valuation of a $\Re(\Sigma, \Pi)$-expression E, is a mapping ϑ from each distinct variable in E, into $D\Re_s$, where s is the sort associated with each variable symbol. In the following, we will omit the $\Re(\Sigma, \Pi)$ notation in programs, constraints, atoms, structures and valuations, when it is clear from the context.

Let us consider a constraint C, defined as a conjunction of a possibly empty or infinite set of atoms C. $\Re \models C\vartheta$, where ϑ is an $\Re$ -valuation of the expressions in C, iff $\Re \models c\vartheta$ holds (i.e. $c\vartheta$ is $\Re$-equivalent to TRUE).

Let $A = p(t_1, ..., t_n)$ be an atom and ϑ a valuation on $t_1, ..., t_n$. We denote by $A\vartheta$ the atom $p(t_1\vartheta, ..., t_n\vartheta)$.

Definition 2.2 *A constraint c is $\Re$-solvable iff there exists an $\Re$-valuation ϑ (the $\Re$-solution) such that $\Re \models c\vartheta$.*

In this paper we do not consider negation. Thus we can relax one of the conditions on the solution compactness of $\Re$-structures, as given in [5, 6]. We will consider only the following condition, namely

- for every sort s and $d \in D\Re_s$, d is uniquely definable in terms of a finite or infinite set of constraints.

Let us give a semantic framework for CLP, based on the previous algebraic structures [4, 6, 5]. In the following we assume that the alphabet (Σ, Π) and the $\Re$ structure are given. Moreover we will often denote n-tuples of terms $t_1, ..., t_n$ by $\bar{t}$.

Definition 2.3 *The set of instances $[c\Box p(\bar{X})]$ of a constrained atom $c\Box p(\bar{X})$ is defined as: $[c\Box p(\bar{X})] = \{p(\bar{X})\vartheta | \vartheta$ is an $\Re$-solution of $c\}$.*

Definition 2.4 *Let A be a set of constrained atoms. We define a preorder $\sqsubseteq$ on A such that: $c_1\Box p(\bar{X}) \sqsubseteq c_2\Box p(\bar{X})$ iff $[c_1\Box p(\bar{X})] \subseteq [c_2\Box p(\bar{X})]$.*

We denote by $\equiv$ the equivalence relation induced by the preorder $\sqsubseteq$.

Definition 2.5 *Let P be a constraint logic program and A be the set of $\Re$-solvable constrained atoms in P. The $\Re$-base of interpretation $\mathcal{B}_\mathcal{P}$ is the quotient set of A w.r.t. $\equiv$.*

Definition 2.6 *An interpretation is any subset of $\mathcal{B}_\mathcal{P}$.*

We recall now the definition of truth as given in [4].

Definition 2.7 *Let I be an interpretation.*

- *A constrained atom $c\Box p(\bar{X})$ is true in I iff $[c\Box p(\bar{X})] \subseteq [I]$;*
- *A clause $H \leftarrow c\Box B_1, ..., B_n$ is true in I iff for each $\Re$-solution ϑ of c, $\{B_1\vartheta, ..., B_n\vartheta\} \subseteq [I]$ implies $H\vartheta \in [I]$.*

Thus, a model of a CLP program P is any interpretation in which all the clauses of P are true.

We can observe that the mechanism introduced in [2, 3] to allow non-ground atoms in the Herbrand base may be generalized in CLP, by allowing constrained atoms into the $\Re$-base. Each constrained atom $c\Box p(\bar{X})$, in fact, represents the set of instances $p(\bar{X})\vartheta$, where ϑ is an $\Re$-solution of the constraint c. Thus, it implicitely represents a possibly infinite set of instances of the variables $\bar{X}$ on the solution domain of c [4].

Let us summarize now the *operational* and *fixpoint* semantics as given in [4].

Operational Semantics

Let P be a CLP program. A $(P, \Re)$-derivation step of a goal G: $\leftarrow c\Box B_1, ..., B_n$ returns a goal of the form $\leftarrow \tilde{c}\Box\tilde{B}_1, ..., \tilde{B}_n$ if there exist n variants of clauses in P, $H_i \leftarrow c_i\Box\tilde{B}_i, i = 1, ..., n$, with no shared variables with the goal, such that

$$\tilde{c} = \{c, c_1, ..., c_n, B_1 = H_1, ..., B_n = H_n\} \text{ is } \Re\text{-solvable.}$$

The $(P, \Re)$-derivation of a goal G is then a finite or infinite sequence of goals, returned by $(P, \Re)$-derivation steps, starting from G.

A successful $(P, \Re)$-derivation of a goal G is a finite $(P, \Re)$-derivation whose last element is the empty clause $c\Box$. (c will be the associated answer constraint).

A characterization of the operational behaviour of CLP may be given in terms of the following success set associated with the program [4]:

$$\begin{aligned} SS(P, \Re) = \; & \{c\Box p(\tilde{X}) | c\Box p(\tilde{X}) \in \mathcal{B}_\mathcal{P}, \\ & (true\Box p(\tilde{X})) \xrightarrow{(P,\Re)} c'\Box., \\ & c\, is\, finite\, and\, c\Box p(\tilde{X}) \equiv c'\Box p(\tilde{X})\}. \end{aligned}$$

Fixpoint Semantics.

The following characterization of the semantics in terms of a continuous transformation map, associated with the program P, may be found in [4]. Let $I \subseteq \mathcal{B}_{\mathcal{P}}$,

$$\begin{aligned} T_{(P,\Re)}(I) = \quad & \{c\Box p(\tilde{X}) \in \mathcal{B}_{\mathcal{P}} \mid \\ & \exists\ a\ renamed\ clause\ p(\tilde{t}) \leftarrow c_0\Box p_1(\tilde{t_1}), ..., p_n(\tilde{t_n}) \in P \\ & \exists c_i\Box p_i(\tilde{X}_i) \in I,\ which\ share\ no\ variables, \\ & c' = c_0 \wedge \{\tilde{X}_1 = \tilde{t}_1, ..., \tilde{X}_n = \tilde{t}_n\} \wedge \{c_1, ..., c_n\}, \\ & c'\ is\ finite,\ \Re - solvable\ and\ c = c' \wedge \{\tilde{X} = \tilde{t}\}\}. \end{aligned}$$

$T_{(P,\Re)}$ satisfies the usual properties of the immediate consequence operator as given in [10, 2, 3, 4, 6]. In particular $M_{(P,\Re)} = T_{(P,\Re)} \uparrow \omega$ is a model.

The following soundness and completeness theorem is the main result on the relation between the operational and the declarative semantics of CLP.

Theorem 2.1 *Let P be a CLP program, and G: $\leftarrow c_0\Box A_1, ..., A_n$ be a goal. Then the goal G has a refutation in P with computed answer constraint c_a iff there exist*

$$(c_1\Box B_1), ..., (c_n\Box B_n) \in M_{(P,\Re)},$$

which share no variables, such that $c_0 \wedge c_1 \wedge ... \wedge c_n \wedge A_1 = B_1 \wedge ... \wedge A_n = B_n$ and c_a have the same solutions for the variables in the goal. $\Diamond$

This theorem shows that $M_{(P,\Re)}$ really models the operational behavior of CLP programs [4].

3 Unfolding of Constraint Logic Programs

In this section we present a transformation rule (the unfolding) on CLP programs. As in the pure logic programming case, this rule is defined in order to satisfy at least the following requirements:

- the unfolding of a CLP program is still a CLP program,
- the rule can always be applied to any procedure call,
- the unfolded program has to be equivalent to the original one w.r.t. the computed answer constraint.

Let us first consider the transformation on single clauses.

Definition 3.1 *Let P be a CLP program and $C : p(\bar{t}) \leftarrow c_0\Box p_1(\bar{t}_1), ..., p_n(\bar{t}_n)$ a clause of P. The unfolding of the clause C with respect to the program P, Unf(C,P), is the following set of clauses*

- $\{C\}$ *if C is a unit clause* $(n = 0)$
- $p(\bar{X}) \leftarrow c\Box p_{1_1}(\bar{t}_{1_1}), ..., p_{1_{k_1}}(\bar{t}_{1_{k_1}}), ..., p_{n_1}(\bar{t}_{n_1}), ..., p_{n_{k_n}}(\bar{t}_{n_{k_n}})$ *such that* $p_i(\bar{X}_i) \leftarrow c_i\Box p_{i_1}(\bar{t}_{i_1}), ..., p_{i_{k_i}}(\bar{t}_{i_{k_i}})$, $i = 1, ..., n$ *are (renamed apart) versions of clauses in P, and*

  $c' = c_0 \wedge \{\bar{X}_1 = \bar{t}_1, ..., \bar{X}_n = \bar{t}_n\} \wedge \{c_1, ..., c_n\}$ *is $\Re$-solvable and* $c = c' \wedge \{\bar{X} = \bar{t}\}$.

Definition 3.2 *Let P be a CLP program defined as a set of constrained clauses $\{C_1, ..., C_n\}$. The unfolding of P is*

$$Unf(P) = \bigcup_{i=1}^{n} Unf(C_i, P)$$

Example 3.1 *The following constraint logic program [6] can be seen as a simple integration routine for some arithmetic function f. The symbol ϵ stands for some term $1/n$ where n is a numeral. The interpretation structure is the Real Arithmetic, with the usual signature and $\Pi_C = \{=, <\}$.*

$$\begin{array}{l}
integrate(A, B, x) \leftarrow \\
\quad 0 < B - A \leq \epsilon, x = (B - A) * f(A + (B - A)/2)\Box. \\
integrate(A, B, x) \leftarrow \\
\quad B - A > \epsilon, M = A + (B - A)/2, x = x_1 + x_2 \Box integrate(A, M, x_1), integrate(M, B, x_2).
\end{array}$$

The unfolding of such a program is

$$\begin{array}{l}
integrate(A, B, x) \leftarrow \\
\quad 0 < B - A \leq \epsilon, x = (B - A) * f(A + (B - A)/2)\Box. \\
integrate(A, B, x) \leftarrow \\
\quad B - A > \epsilon, M = A + (B - A)/2, x = x_1 + x_2, \\
\quad 0 < M - A \leq \epsilon, x_1 = (M - A) * f(A + (M - A)/2), \\
\quad 0 < B - M \leq \epsilon, x_2 = (B - M) * f(M + (B - M)/2)\Box. \\
integrate(A, B, x) \leftarrow \\
\quad B - A > \epsilon, M = A + (B - A)/2, x = x_1 + x_2, \\
\quad M - A > \epsilon, M' = A + (M - A)/2, x_1 = x_1' + x_2', \\
\quad B - M > \epsilon, M'' = M + (B - M)/2, x_2 = x_1'' + x_2'' \Box \\
\quad integrate(A, M', x_1'), integrate(M', M, x_2'), \\
\quad integrate(M, M'', x_1''), integrate(M'', B, x_2'').
\end{array}$$

Since $M = A + (B - A)/2$, the constraint $...M - A > \epsilon, ..., 0 < B - M \leq \epsilon...$ is inconsistent. Thus, we can not unfold the second clause with respect to the first and the second one in the program, and viceversa.

The following results formally establish the equivalence between the semantics of the unfolding of a program and the semantics of the program itself. It follows from the definition that the unfolding rule replaces each clause of the program with a set of clauses, where the corresponding bodies are the result of a single $(P, \Re)$-derivation step, by selecting the whole atoms in the source body. Thus we can argue that the unfolding of a clause is the empty set if and only if each possible $(P, \Re)$-derivation, by using such a clause, is always failed [8].

Theorem 3.1 *Let P be a CLP program. Then $M_{(Unf(P), \Re)} \subseteq M_{(P, \Re)}$.*

Proof. We prove that for each clause $C \in Unf(P)$, C is true in $M_{(P, \Re)}$. Let n denote the length of the body of C.

- Let $n = 0$. Since, by the definition, $C \in P$ and $M_{(P, \Re)}$ is a model for P, the theorem trivially holds.

- Let $n \geq 0$ and C: $p(\bar{X}) \leftarrow c \Box p_{1_1}(\bar{t}_{1_1}), ..., p_{n_{k_n}}(\bar{t}_{n_{k_n}})$. Then by definition there exists a clause $p(\bar{t}) \leftarrow c_0 \Box p_1(\bar{t}_1), ..., p_n(\bar{t}_n)$ in P such that, by considering the following renamed apart versions of clauses in P,

  $p_1(\bar{X}_1) \leftarrow c_1 \Box p_{1_1}(\bar{t}_{1_1}), ..., p_{1_{k_1}}(\bar{t}_{1_{k_1}})$.

  ....

  $p_n(\bar{X}_n) \leftarrow c_n \Box p_{n_1}(\bar{t}_{n_1})$,..., $p_{n_{k_n}}(\bar{t}_{n_{k_n}})$.

  the associated constraint:

$$c'_i = c_0 \wedge \bigwedge_{j=1}^{i} c_j \wedge \{\bar{X}_1 = \bar{t}_1, ..., \bar{X}_i = \bar{t}_i\}$$

  is $\Re$-solvable, for each $i = 1, ..., n$, and $c = c'_n \wedge \{\bar{X} = \bar{t}\}$.

  Let us consider the following clause

  C'_i: $p(\bar{X}) \leftarrow c'_i \wedge \{\bar{X} = \bar{t}\} \Box p_{1_1}(\bar{t}_{1_1})$,..., $p_{i_{k_i}}(\bar{t}_{i_{k_i}}), p_{i+1}(\bar{t}_{i+1})$,...,$p_n(\bar{t}_n)$.

  It is a resolvent of the clauses C'_{i-1} and C_i. Thus, for each $i = 1, ..., n$, C'_i is true in $M_{(P,\Re)}$. $\diamond$

The next theorem shows that each minimal model of $Unf(P)$ is also a model of P. The equivalence between the program P and its unfolding is then proved in conjunction with the previous result.

Theorem 3.2 *Let P be a CLP program, then:* $M_{(P,\Re)} \subseteq M_{(Unf(P),\Re)}$.

Proof. We prove by induction on n that: $T_{(P,\Re)} \uparrow n \subseteq M_{(Unf(P),\Re)}$.

- Base step: let $(c \Box p(\bar{X})) \in T_{(P,\Re)} \uparrow 1$. Thus, by definition

  $p(\bar{t}) \leftarrow c_0. \in P$ and $c = c_0 \wedge \{\bar{X} = \bar{t}\}$. By the unfolding rule definition we have that $p(\bar{t}) \leftarrow c_0. \in Unf(P)$, and then the unit clause is true in $M_{(Unf(P),\Re)}$.

- Inductive step: let $(c \Box p(\bar{X})) \in T_{(P,\Re)} \uparrow (m+1) = T_{(P,\Re)}(T_{(P,\Re)} \uparrow m)$. Thus, by the $T_{(P,\Re)}$ definition,

  $\exists\, p(\bar{t}) \leftarrow c_0 \Box p_1(\bar{t}_1), ..., p_n(\bar{t}_n) \in P$, $\exists\, (c_i \Box p_i(\bar{X}_i)) \in T_{(P,\Re)} \uparrow m$ such that

$$c' = \bigwedge_{i=0}^{n} c_i \wedge \{\bar{X}_1 = \bar{t}_1, ..., \bar{X}_n = \bar{t}_n\}$$

  is $\Re$-solvable and $c = c' \wedge \{\bar{X} = \bar{t}\}$. Moreover, by the hypothesis

$$c = c_0 \wedge \bigwedge_{i=1}^{n} c_i \wedge \{\bar{X}_1 = \bar{t}_1, ..., \bar{X}_n = \bar{t}_n\} \wedge \{\bar{X} = \bar{t}\}$$

  is $\Re$-solvable.

  Since $\exists p(\bar{t}) \leftarrow c_0 \Box p_1(\bar{t}_1), ..., p_n(\bar{t}_n) \in P$ and $(c_i \Box p_i(\bar{X}_i)) \in T_{(P,\Re)} \uparrow m$ for each $i = 1, ..., n$, there exist:

  $p_1(\bar{g}_1) \leftarrow c_{0_1} \Box p_{1_1}(\bar{g}_{1_1}), ..., p_{1_{k_1}}(\bar{g}_{1_{k_1}}) \in P$

....

$p_n(\bar{g}_n) \leftarrow c_{0_n} \Box p_{1_n}(\bar{g}_{1_n}), ..., p_{n_{k_n}}(\bar{g}_{n_{k_n}}) \in P$

and, by the $T_{(P,\Re)}$ definition, for each $j = 1, ..., k_i$

$\exists\, (\hat{c_{i_j}} \Box p_{i_j}(\bar{X}_{i_j})) \in T_{(P,\Re)} \uparrow (m-1)$, such that

$$c_i = c_{0_i} \wedge \bigwedge_{j=1}^{k_i} \{\bar{X}_{i_j} = \bar{g}_{i_j}\} \wedge \bigwedge_{j=1}^{k_i} \{\hat{c_{i_j}}\} \wedge \{\bar{X}_i = \bar{g}_i\}$$

is $\Re$ solvable, for each $i = 1, ..., n$. Thus, by the inductive hypothesis

$(\hat{c_{i_j}} \Box p_{i_j}(\bar{X}_{i_j})) \in T_{(P,\Re)} \uparrow (m-1) \implies (\hat{c_{i_j}} \Box p_{i_j}(\bar{X}_{i_j})) \in M_{(Unf(P),\Re)}$.

Let us consider the unfolding of $p(\bar{t}) \leftarrow c_0 \Box p_1(\bar{t}_1), ..., p_n(\bar{t}_n)$ in P,

$p(\tilde{X}) \leftarrow c_{unf} \wedge \{\tilde{X} = \bar{t}\} \Box p_{1_1}(\bar{g}_{1_1}),...,p_{n_{k_n}}(\bar{g}_{n_{k_n}}) \in$ Unf(P), where

$$c_{unf} = c_0 \wedge \bigwedge_{i=1}^{n} \{\bar{g}_i = \bar{t}_i\} \wedge \bigwedge_{i=1}^{n} c_{0_i}.$$

Since c is $\Re$-solvable, and

$$c = c_0 \wedge \bigwedge_{i=1}^{n} \{\bar{X}_i = \bar{t}_i\} \wedge$$

$$\wedge \bigwedge_{i=1}^{n} \{ \underbrace{c_{0_i} \wedge \bigwedge_{j=1}^{k_i} \{\bar{X}_{i_j} = \bar{g}_{i_j}\} \wedge \bigwedge_{j=1}^{k_i} \{\hat{c}_{i_j}\} \wedge \{\bar{X}_i = \bar{g}_i\}\}}_{c_i} \wedge \{\bar{X} = \bar{t}\}.$$

we have that

$$c = \underbrace{c_0 \wedge \bigwedge_{i=1}^{n} \{\bar{t}_i = \bar{g}_i\} \wedge \bigwedge_{i=1}^{n} \{c_{0_i}\}}_{c_{unf}} \wedge \bigwedge_{i=1}^{n} \bigwedge_{j=1}^{k_i} \{\bar{X}_{i_j} = \bar{g}_{i_j}\} \wedge \bigwedge_{i=1}^{n} \bigwedge_{j=1}^{k_i} \{\hat{c}_{i_j}\} \wedge \{\bar{X} = \bar{t}\}.$$

is $\Re$-solvable.

Moreover $p(\tilde{X}) \leftarrow c_{unf} \Box p_{1_1}(\bar{g}_{1_1}),..., p_{n_{k_n}}(\bar{g}_{n_{k_n}})$ is true in $M_{(Unf(P),\Re)}$, and, by definition, this is true if and only if for each $\Re$-solution ϑ of c_{unf}, if $\{p_{1_1}(\bar{g}_{1_1})\vartheta,..., p_{n_{k_n}}(\bar{g}_{n_{k_n}})\vartheta\} \subseteq [M_{(Unf(P),\Re)}]$ then $p(\tilde{X})\vartheta \in [M_{(Unf(P),\Re)}]$.

Since, by inductive hypothesis, $(\hat{c}_{i_j} \Box p_{i_j}(\bar{X}_{i_j})) \in M_{(Unf(P),\Re)}$, then for each $\Re$-solution σ of c, $p_{i_j}(\bar{X}_{i_j})\sigma \in [M_{(Unf(P),\Re)}]$.

Thus, $p(\tilde{X})\sigma \in [M_{(Unf(P),\Re)}]$, and therefore $c \Box p(\tilde{X}) \in M_{(Unf(P),\Re)}$. $\diamond$

As a consequence, we have proved the equivalence of P and $Unf(P)$, namely

Corollary 3.3 *Let P be a CLP program. Then $M_{(P,\Re)} = M_{(Unf(P),\Re)}$*

In the next theorem we will focus on the operational behavior of $Unf(P)$. Since $M_{(P,\Re)} = M_{(Unf(P),\Re)}$, we will see that any refutation of a goal in P, with answer constraint c_a, has an equivalent refutation in $Unf(P)$ with answer constraint c'_a, having the same solutions w.r.t. the variables in the goal. Moreover, the refutation in $Unf(P)$ is generally shorter than the corresponding in P.

In the following, as in the pure logic programming case [8], we will assume that the computation rule used in the refutation, always selects all the atoms in the goal.

Theorem 3.4 *Let $\leftarrow c\Box A_1, ..., A_n$ be a goal which has a refutation in P of length l, with answer constraint c_a. Then:*

1. *if $l = 1$, the goal has the same refutation in Unf(P),*
2. *if $l > 1$, the goal has a refutation in Unf(P) of length $l' < l$ with an equivalent (w.r.t. the solutions) answer constraint.*

Proof.

1. is trivial since each unit clause in P is also in $Unf(P)$.
2. the proof is given by induction on the refutation length.
   - Base step: let $l = 2$, and let us suppose that
     $(c\Box A_1, ..., A_n) \stackrel{C_1,..,C_n}{\longrightarrow} (c'\Box G_1, ..., G_n) \stackrel{C_{i_j}}{\longrightarrow} (c'_a\Box.)$
     where
     $C_i :\ H_i \leftarrow c_i\Box G_i = B_{i_1}, ..., B_{i_{k_i}}.$ and
     $C_{i_j} :\ H_{i_j} \leftarrow c_{i_j}\Box.$
     Moreover, by definition
     $c' = c\bigwedge_{i=1}^{n}\{c_i \wedge H_i = A_i\}$ and
     $c'_a = c'\bigwedge_{i=1}^{n}\bigwedge_{j=1}^{k_i}\{c_{i_j} \wedge B_{i_j} = H_{i_j}\}$.
     are $\Re$-solvable.
     Let us consider the unfolding in P of each clause: $H_i \leftarrow c_i\Box G_i$,
     $H_i \leftarrow c_i\bigwedge_{j=1}^{k_i}\{B_{i_j} = H_{i_j} \wedge c_{i_j}\}\Box. \in Unf(P)$.
     Since c'_a is $\Re$-solvable, then there exists in $Unf(P)$ a refutation of the goal of length 1, with an equivalent answer constraint.
   - The inductive step is quite similar to the base one.
     Let us suppose that the thesis holds for $l \geq 2$ and that the goal has the following refutation of length $l + 1$
     $(c\Box A_1, ..., A_n) \stackrel{C_1,..,C_n}{\longrightarrow} (c'\Box G_1, ..., G_n) \stackrel{C_{i_j}}{\longrightarrow} (c''\Box G_{1_1}, ..., G_{n_{k_n}}) \stackrel{*}{\longrightarrow} (c'_a\Box.)$
     Then, for each $i = 1, ..., n$
     $H_i \leftarrow \tilde{c}_i\Box G_{i_1}, ..., G_{i_{k_i}} \in Unf(P)$, and $\tilde{c}_i = c'\bigwedge_{j=1}^{k_i}\{B_{i_j} = H_{i_j}\}$ is $\Re$-solvable.
     Thus, as in the base case
     $(c\Box A_1, ..., A_n) \longrightarrow (\bigwedge_{i=1}^{n}\tilde{c}_i\Box G_{1_1}, ..., G_{n_{k_n}})$
     is a correct refutation step in $Unf(P)$, which, by the induction hypothesis, leads trivially to the thesis. $\Diamond$

4 The Unfolding Semantics

In this section we consider the iterative application of the unfolding rule to a given CLP program. Such an application gives us an alternative way to characterize the semantics of CLP.

Let us consider a CLP program P together with the following inductive definition:

$P^0 = P$

$P^{i+1} = Unf(P^i)$

Since for each $i, j \in \omega$, $M_{P^i} = M_{P^j}$, this is a collection of CLP programs which are all equivalent [8]. Thus, by considering the following set

$U^i = \{(c \wedge \{\bar{X} = \bar{t}\} \Box p(\bar{X}))$ s.t $p(\bar{t}) \leftarrow c\Box.$ is a clause of $P^i\}$, we can define the unfolding semantics of CLP as follows.

Definition 4.1 *The unfolding semantics of a CLP program P, denoted by $\mathcal{M}$, is*

$$\mathcal{M} = \bigcup_{i \in \omega} U^i.$$

Roughly speaking, since the unfolding semantics of a CLP program is a possibly infinite set of unit clauses computed by the repeated application of the unfolding rule, we claim that $\mathcal{M}$ is the same as the success set of P. This result is formally proved in the following theorem.

Theorem 4.1 *Given a CLP program P, $\mathcal{M} = M_{(P,\Re)}$.*

Proof.

- We prove that $\mathcal{M} \subseteq M_{(P,\Re)}$. This holds trivially, by definition. In fact, since $(c\Box p(\bar{X})) \in \mathcal{M}$ if and only if $\exists\ i \in \omega$ such that $p(\bar{t}) \leftarrow c_0\Box. \in P^i$ and $c = c_0 \wedge \{\bar{X} = \bar{t}\}$, then $(c\Box p(\bar{X})) \in M_{(P,\Re)}$.

- Let us suppose that $\mathcal{M} \subset M_{(P,\Re)}$. Thus, by definition there exists a constrained atom $(c\Box p(\bar{X})) \in M_{(P,\Re)}$ such that $(c\Box p(\bar{X})) \notin T_{(P^i,\Re)} \uparrow 1$, for each $i \in \omega$ (since for each $i \in \omega$: $U^i = T_{(P^i,\Re)} \uparrow 1$).

  Moreover, by completeness, the goal $\leftarrow c\Box p(\bar{X})$. has a refutation in P and its length has to be $l_0 > 0$ (otherwise $(c\Box p(\bar{X})) \in T_{(P^0,\Re)} \uparrow 1$).

  By theorem 3.4, the same goal has a refutation of length $l_1 < l_0$ in P^1, with an equivalent answer constraint, and so on, for $i = 2, 3, \ldots..$ This leads to a contradiction, from which the thesis follows. $\diamond$

The unfolding rule for the CLP framework, is an extension of a rule originally developed for positive logic programs [8]. Similar results can be obtained by considering other unfolding rules [9] in the framework of CLP.

5 Concluding remarks

We have shown that the operational semantics of CLP leads a suitable transformation rule which allows to define a semantics equivalent to the model-theoretic one. The transformation is strongly related to the immediate consequence operator.

In fact, as in the pure logic programming case, by considering an interpretation I as a set of constrained rules (CLP clauses), we have

$$T_{(C,\Re)} = Unf(C, I),$$

where $T_{(C,\Re)}$ is the immediate consequence operator associated with the clause C [8]. This result, which is inherited from the theory of pure logic programs, stresses the very close relation between the standard fixpoint semantics of CLP and the unfolding one.

Thus, many of the results on the unfolding semantics theory of pure logic programs, as given in [8], hold in the framework of CLP too. In particular, since we deal with constraints instead of substitutions, many of the previous results are much easier to prove. Many of technical lemmas on the unification theory have, in fact, a more natural and general interpretation by using constraints. Thus, we obtain much simpler and more elegant proofs for tedious theorems, such as 3.1, 3.2 and 3.3. The rational behind this remark has to be found in the role played by unification in substitution based proofs. Unification, in fact, as already argued in the introduction, is somewhat redundant, since it embodies the transformation to solved form in addition to solvability checking. In fact, from the proof viewpoint, we only need to test solvability of sets of constraints (or equations, in the case of pure logic programs).

References

[1] F. Denis and J.-P. Delahaye. Unfolding, Procedural and Fixpoint Semantics of Logic Programs. In *Proc. STACS'91*, 1991.

[2] M. Falaschi, G. Levi, M. Martelli, and C. Palamidessi. Declarative Modeling of the Operational Behavior of Logic Languages. *Theoretical Computer Science*, 69(3):289–318, 1989.

[3] M. Falaschi, G. Levi, M. Martelli, and C. Palamidessi. A Model-Theoretic Reconstruction of the Operational Semantics of Logic Programs. Technical Report TR 32/89, Dipartimento di Informatica, Università di Pisa, 1989. To appear in *Information and Computation*.

[4] M. Gabbrielli and G. Levi. Modeling answer constraints in Constraint Logic Programs. Technical Report TR 4/91, Dipartimento di Informatica, Università di Pisa, 1991.

[5] J. Jaffar and J.-L. Lassez. Constraint Logic Programming. In *Proc. Fourteenth Annual ACM Symp. on Principles of Programming Languages*, pages 111–119. ACM, 1987.

[6] J. Jaffar and J.-L. Lassez. Constraint Logic Programming. Technical report, Department of Computer Science, Monash University, June 1986.

[7] G. Levi. Models, Unfolding Rules and Fixpoint Semantics. In R. A. Kowalski and K. A. Bowen, editors, *Proc. Fifth Int'l Conf. on Logic Programming*, pages 1649–1665. The MIT Press, Cambridge, Mass., 1988.

[8] G. Levi and P. Mancarella. The Unfolding Semantics of Logic Programs. Technical Report TR-13/38, Dipartimento di Informatica, Università di Pisa, 1988.

[9] H. Tamaki and T. Sato. Unfold/Fold Transformations of Logic Programs. In Sten-Åke Tarnlund, editor, *Proc. Second Int'l Conf. on Logic Programming*, pages 127–139, 1984.

[10] M. H. van Emden and R. A. Kowalski. The semantics of predicate logic as a programming language. *Journal of the ACM*, 23(4):733–742, 1976.