A bottum-up semantics for constructive negation

A. Bossi, M. Fabris, M.C. Meo

Computer Science/Department of Software Technology

Report CS-R9439 July 1994

A Bottom-up Semantics for Constructive Negation

Annalisa Bossi
Dipartimento di Matematica Pura ed Applicata
Università di Padova, Via Belzoni 7, 35131 Padova, Italy
isa@zenone.unipd.it

Massimo Fabris
Dipartimento di Matematica Pura ed Applicata
Università di Padova, Via Belzoni 7, 35131 Padova, Italy
massimo@hilbert.math.unipd.it

Maria Chiara Meo
Dipartimento di Informatica
Università di Pisa, Corso Italia 40, 56125 Pisa, Italy.
meo@di.unipi.it

Abstract

The constructive negation rule has been introduced by Chan [5, 6] to overcome the main drawbacks of the negation-as-failure rule: the unsoundness of floundering programs and, consequently, the inability of providing answers for non-ground negative queries. In this paper we define a bottom-up semantics for constructive negation which we prove sound and complete with respect to the three-valued completion of the program. The semantics describes answers as well as undefined computations for both positive and negative queries. Its construction closely follows the basic idea of constructive negation whereby answers to a negative query are obtained by negating a *frontier* of the computation tree for the corresponding positive query. Therefore, the proposed semantics can be considered as a natural base for reasoning on the operational semantics for constructive negation defined in the literature. Moreover, we show how the semantics can be effectively used to perform a bottom-up computation of the answers of a normal query.

AMS Subject Classification (1991): 68N17, 68Q55.
CR Subject Classification (1991): D.1.6, D.3.1, F.3.2, F.4.1, I.2.3.
Keywords and Phrases: logic programming, semantics.
Note: A preliminary short version of this paper appeared in [2]. Part of this work has been carried out while the third author was visiting CWI, Amsterdam.

Report CS-R9439
ISSN 0169-118X
CWI
P.O. Box 94079, 1090 GB Amsterdam, The Netherlands

1 Introduction

The standard rule for dealing with negation in logic programming is the "negation as failure" rule. Its main drawback is that *computing* by negation as failure, actually means *testing* by negation as failure, that is no answers are produced except for the yes/no ones.

To overcome this limitation Chan [5, 6] introduced the *constructive negation* rule which subsumes negation as failure and extends it by allowing non-ground negative subgoals to bind variables in the same way as positive ones. The basic idea which is formalized by SLD-CNF resolution is that, to handle a negative subgoal, one first considers its positive version and then negates all its answers. Answers to negative goals are described by first-order formulas which are interpreted in CET, the equality theory defined by Clark.

Stuckey [25] pointed out how constraint logic programming provides a much more natural setting for describing constructive negation and described a setting for constructive negation for constraint logic programming over arbitrary structures. He gave a new operational schema and proved its soundness and completeness with respect to the three-valued completion of the program.

Recently, Drabent [11, 10] proposed a method for deriving (constrained) answers for (constrained) normal queries in normal programs called SLDFA-resolution, an extension of SLDNF-resolution. The basic notions of SLDFA-resolution are SLDFA-refutation and finitely failed SLDFA-tree; they are mutually defined as the corresponding notions of SLDNF-resolution. Comparisons between Drabent's method and those defined by Chan and Stuckey suggest that the first may have practical advantages over the other two.

In this paper we follow a semantical approach to constructive negation. We develop a semantics in the style of [13]: we construct a denotation which directly characterizes the program behavior on all most general atomic queries (positive or negative) and contains enough information to represent the program behavior on all queries.

We consider CLP normal programs as defined by [16]. Our denotations (*uncertain interpretations*) extend both those defined in [8, 9, 1] for the semantics of definite programs, and those in [14] for modeling answer constraints in CLP normal programs. They contain four kinds of objects which represent the four different computational aspects of a program we are interested in. Three of them have a logical reading in the three-valued completion of the program. They are: *positive constrained atoms*, which represent success answers for positive general queries; *negative constrained atoms*, which represent success answers for negative general queries or, symmetrically, finite failures of positive general queries; *uncertain negative atoms*, which represent undefined answers on both negative and positive queries, they entail undefined values in three-valued logic. Besides a fourth kind of object (*uncertain positive atom*) is used to represent divergent computations of positive queries. Since a positive query may have both successful and divergent computations, uncertain positive atoms do not necessary entail undefined values.

To illustrate this point consider the definite program $\{p(a).,\ p(X) \leftarrow p(X).\}$. The success constraint for the query $p(X)$ is $X = a$, but its negation, $(X \neq a)$, is not a success for the query $\neg p(X)$. Note that the three valued model is $\{p(a)\}$. Our semantic denotation will contain: the positive constrained atom $X = a \,\Box\, p(X)$, the uncertain negative atom $X \neq a \,\Box\, ?p(X)$ and the uncertain positive atom $true \,\Box\, \widehat{p(X)}$. The last one models the divergent computation for the query $p(X)$.

Actually, the positive components in an uncertain interpretation are sufficient to determine the negative ones. For instance, observe that the absence of success answers for negative general queries in the above program can be derived by the fact that there is no way of negating both the constraint in the positive constrained atom $(X = a)$ and that in the uncertain positive atom $(true)$. On the other hand, the uncertain negative atom can be obtained by negating the success and asserting the divergence.

The semantics is obtained by iterating an operator Ψ_P which maps uncertain interpretations to uncertain interpretations. Our construction agrees with Fitting's Φ_P operator [12] in the sense that at each iteration step there is a one-to-one correspondence between the solutions represented by the uncertain interpretation and the three-valued interpretation constructed by Fitting's operator. There are obvious analogies between Ψ_P and Φ_P^A, the non-ground (constrained) version of Φ_P defined by Stuckey in [25]. The main difference is that instead of deriving information on success and failure we derive information on success and divergent computations. Note that our information is richer than the former. In fact, by negating both successes and failures we obtain only a subset of divergent computations, while by negating both successes and divergent computations all failures can be obtained. Moreover the second construction conforms much better with the basic idea of constructive negation whereby an answer to a negative query is obtained by negating a *frontier* of the computation tree for the corresponding positive query.

The paper is organized as follows. In Section 2 we give the basic notations on *CLP* normal programs. In Section 3 we introduce our semantic domain and the notions of uncertain base and uncertain interpretations. The bottom-up semantics is defined in Section 4. In Section 5 we discuss and compare some other approaches to constructive negation. We dedicate special attention to Drabent's work and those properties of SLDFA-resolution captured by our semantics. Some proofs are deferred to the appendix.

2 Preliminaries

We assume the reader is familiar with the basic concept of logic programming. We recall the basic *CLP* concepts as defined in [16] and [20]. A first order language is defined on a function symbols set denoted by Σ, a predicate symbols set denoted by Π and a collection of variables denoted by V. The predicate symbols are partitioned into two sets: Π_C which are pre-defined predicates and Π_B which are the predicates to be defined by the program. We assume that Π_C contains the predicate symbol $=$. $\tau(\Sigma \cup V)$ and $\tau(\Sigma)$ denote the set of terms and ground terms (i.e. terms without variables) built on Σ and V. An *atom* is of the form $p(t_1, \ldots, t_n)$ where p is an n-ary symbol in Π_B and $t_i \in \tau(\Sigma \cup V)$, $i \in [1, n]$. A *literal* is either an atom or the negation of an atom. A *constraint* is a well formed formula over the alphabets Π_C and Σ. In the following, $\tilde{t}$, $\tilde{X}$ will denote tuple of terms and *distinct* variables, while $\tilde{L}$ denotes a (possibly empty) conjunction of literals. If o is a syntactic object, $FV(o)$ is the set of variables which are not explicitly quantified in o. Given a formula f, $\exists f$ and $\forall f$ denote its existential and universal closure respectively. $\exists_{-\tilde{X}} f$ denotes the existential closure of the formula f except for the variables $\tilde{X}$, which remain unquantified. Let F be a set of formulas, we use the notations $\bigvee F$ and $\bigwedge F$ as shorthands for the formulas $(\bigvee_{f \in F} f)$ and $(\bigwedge_{f \in F} f)$ respectively, where, $(\bigvee_{f \in \emptyset} f) = false$ and $(\bigwedge_{f \in \emptyset} f) = true$.

The ordinal powers $f{\uparrow}^\alpha$ of a monotonic function f on a complete lattice are defined as usual, namely $f{\uparrow}^0(x) = x$, $f{\uparrow}^{\alpha+1}(x) = f(f{\uparrow}^\alpha(x))$ for any ordinal α and $f{\uparrow}^\gamma(x) = \bigsqcup_{\alpha<\gamma} f{\uparrow}^\alpha(x)$ for γ limit ordinal. We also use the standard notation $f{\uparrow}^\alpha = f{\uparrow}^\alpha(\bot)$, where $\bot$ is the bottom element of the lattice.

Definition 2.1 (*CLP* normal programs) *[16] A* normal program *is a finite set of clauses of* *the form $H \leftarrow c \,\Box\, \tilde{L}$. where c is a constraint, H (the head) is an atom and $\tilde{L}$ (the body) is a (possibly empty) conjunction of literals. A* normal goal *is a program clause with no head and with a non-empty body.*

In the following we will consider clause heads of the form $p(\tilde{X})$, where $\tilde{X}$ is a sequence of distinct variables and we will denote by P^* the completed definitions of the predicates in a normal program P.

Moreover we will often use *normal queries* instead of normal goals, where given a normal goal $\leftarrow c \,\Box\, \tilde{L}$ the corresponding (normal) query is $c \,\Box\, \tilde{L}$.

Definition 2.2 *A constrained atom is of the form $c \,\Box\, p(\tilde{X})$, where c is a constraint, $FV(c) \subseteq \{\tilde{X}\}$ and $p(\tilde{X})$ is an atom.*

A *structure* $\mathcal{D}$ over the alphabets Π_C and Σ, consists of a non empty set (D) and any interpretation of each function and predicate symbol according to its arity. The structures considered in CLP are the "solution compact" ones as defined in [16, 15]. A *domain theory $\mathcal{T}$ corresponding* to the structure $\mathcal{D}$ is a first-order consistent theory containing only predicate symbols from Π_C, such that for any constraint c with $FV(c) = \emptyset$, $\mathcal{D} \models c$ iff $\mathcal{T} \models c$. Moreover we require that for every constraint c either $\mathcal{T} \models \exists c$ or $\mathcal{T} \models \neg\exists c$ (namely, $\mathcal{T}$ is *satisfaction complete* [20]). A domain theory axiomatizes the particular domain $\mathcal{D}$ on which we wish to compute. We do not make any assumption on Σ. Rather we require that the domain theory $\mathcal{T}$ contains the standard theory CET, given in [7] to axiomatize unification. Moreover if the set of function symbols Σ is finite we assume the (weak) domain closure axiom (DCA) be added to CET, thus achieving the completeness of the theory CET in the case of a language with finite set of function symbols. Informally the axiom DCA [19] ensures that in the interpretation domain of any model of the theory, every object is a value of a non-variable term. A *valuation* is a mapping from the variables to D. The notion of valuation is extended in the obvious way to terms and constraints. A negation of an equation $s = t$ (a *disequation* or *inequality*) will be written as $s \neq t$. If $\tilde{t} = (t_1, \ldots, t_n)$ and $\tilde{s} = (s_1, \ldots, s_n)$ are two sequences of terms, $\tilde{t} = \tilde{s}$ will denote the set of equations $\{t_1 = s_1, \ldots, t_n = s_n\}$.

A constraint c is *satisfiable* in the theory $\mathcal{T}$ iff there exists a valuation θ such that $\mathcal{T} \models c\theta$. θ is called a *solution* of c. If $C = \bigwedge_{i \in I} c_i$ is a possibly infinite conjunction of constraints, $\mathcal{T} \models C\theta$ iff for any $i \in I$, $\mathcal{T} \models c_i\theta$. Analogously, if $C = \bigvee_{i \in I} c_i$ is a possibly infinite disjunction of constraints, $\mathcal{T} \models C\theta$ iff there exists $i \in I$, such that $\mathcal{T} \models c_i\theta$.

We also introduce the following $[\,]$ operator on constrained atoms, which returns the set of "domain instances".

Definition 2.3 *[16] The set of "domain instances" $[c \,\Box\, p(\tilde{X})]$ of a constrained atom $c \,\Box\, p(\tilde{X})$ is defined as $[c \,\Box\, p(\tilde{X})] = \{p(\tilde{X})\theta \mid \theta$ is a solution of $c\}$.*
Let S be a set of constrained atoms. Then $[S] = \bigcup_{A \in S}[A]$.

Next we define a preorder $\preceq$ on constraints, a preorder $\sqsubseteq$ on sets of constraints and the induced equivalence relations.

Definition 2.4 *Let c_1 and c_2 be constraints and let C_1 and C_2 be sets of constraints. Then*

- *$c_1 \preceq c_2$ iff $\mathcal{T} \models \forall(c_1 \rightarrow c_2)$. We denote by $\equiv$ the equivalence relation induced by $\preceq$.*
- *$C_1 \sqsubseteq C_2$ iff for any $c_1 \in C_1$ such that $c_1 \not\equiv false$ there exists $c_2 \in C_2$ such that $c_1 \preceq c_2$. We denote by $\approx$ the equivalence relation induced by $\sqsubseteq$.*

Finally, if $\sim$ is an equivalence relation defined on a set $\mathcal{S}$ and $S \in \mathcal{S}$, we denote by $S_\sim$ the equivalence class in $\mathcal{S}/\!\sim$ which contains S.

3 Uncertain interpretations

We extend the notion of π-interpretation as introduced in [14] in order to provide three-valued models of the completion of a *CLP* normal program. First of all, a *partial* interpretation, as defined in [17], is any total function F from the set of all ground atoms into $\{\mathbf{t},\mathbf{f},\mathbf{u}\}$, where $\{\mathbf{t},\mathbf{f},\mathbf{u}\}$ are interpreted as *true*, *false* and *undefined.* According to our notation, we will represent such a function F as a set of ground literals $F^+ \cup F^-$, where $F^+ = \{p(\tilde{t}) \mid p(\tilde{t})$ is a ground atom and $F(p(\tilde{t})) = \mathbf{t}\}$ and $F^- = \{\neg p(\tilde{t}) \mid p(\tilde{t})$ is a ground atom and $F(p(\tilde{t})) = \mathbf{f}\}$. The extension of a partial interpretation to ground constraints and to ground formulas is defined in [25], by the following rules.

- Let c be a ground constraint. c is *true* in F iff $\mathcal{T} \models c$ and c is *false* in F iff $\mathcal{T} \models \neg c$.
- We assume the usual strong three-valued interpretation of the symbols $\wedge$, $\vee$, $\neg$, $\forall$, $\exists$, $\rightarrow$, and following Kunen, we use Lukasiewicz's truth table for the connective $\leftrightarrow$. Moreover the symbols '$\Box$' and ',' will be interpreted as $\wedge$.

All the following definitions are related to a given $\mathcal{D}$ (and therefore to a given (Π_C, Σ)).

Before giving the definition of uncertain base we extend the preorder on constraints to constrained atoms. It represents the notion of "being more constrained". The equivalence induced by such a preorder is used in the semantic domain in order to abstract from syntactical differences among constrained atoms.

Definition 3.1 *Let* $c_1 \Box p(\tilde{X})$ *and* $c_2 \Box p(\tilde{X})$ *be constrained atoms. Then*
$c_1 \Box p(\tilde{X}) \preceq c_2 \Box p(\tilde{X})$ *iff* $c_1 \preceq c_2$.
The equivalence induced by $\preceq$ *on the set of atoms is still denoted by* $\equiv$. *The quotient set of all the constrained atoms w.r.t. the equivalence relation* $\equiv$ *will be denoted by* $\mathcal{A}$.

It is easy to see that the above equivalence $\equiv$ on constrained atoms corresponds to set equality on domain instances.

Lemma 3.2 *Let* $c_1 \Box p(\tilde{X})$ *and* $c_2 \Box p(\tilde{X})$ *be constrained atoms. Then*
$c_1 \Box p(\tilde{X}) \equiv c_2 \Box p(\tilde{X})$ *iff* $[c_1 \Box p(\tilde{X})] = [c_2 \Box p(\tilde{X})]$.

For the sake of simplicity, we let the constrained atom $c \Box p(\tilde{X})$ denote the equivalence class $(c \Box p(\tilde{X}))_{\equiv}$ in $\mathcal{A}$ and, conversely, any $B \in \mathcal{A}$ will be considered also as a constrained atom obtained by selecting any (arbitrary) representative element in B. It is easy to verify that all our definitions are independent from the choice of such an element. The ordering induced by $\preceq$ on $\mathcal{A}$ will still be denoted by $\preceq$.

We now introduce the uncertain base.

Definition 3.3 (uncertain base) *Let P be a normal program. The uncertain base of interpretations $\mathcal{B}$ is the union of the following sets.*

$$\begin{array}{lcll}
\mathcal{B}^+ & = & \mathcal{A} & \textbf{positive component} \\
\mathcal{B}^- & = & \{c \Box \neg p(\tilde{X}) \mid c \Box p(\tilde{X}) \in \mathcal{A}\} & \textbf{negative component} \\
\mathcal{B}^{\widehat{+}} & = & \{c \Box \widehat{p(\tilde{X})} \mid c \Box p(\tilde{X}) \in \mathcal{A}\} & \textbf{uncertain positive component} \\
\mathcal{B}^{?} & = & \{c \Box ?p(\tilde{X}) \mid c \Box p(\tilde{X}) \in \mathcal{A}\} & \textbf{uncertain negative component.}
\end{array}$$

In the following, given $J \subseteq \mathcal{B}$ and $o \in \{+, -, \widehat{+}, ?\}$ we shall use the notations J^o for $J \cap \mathcal{B}^o$ and $J_{|A}$ for $\{c \mid c \Box A \in J\}$. Intuitively any subset I of $\mathcal{B}$ conveys both *certain* and *uncertain* information. The certain information is contained in I^+ and I^- while the uncertain one is contained in the two other components $I^{\widehat{+}}$ and $I^?$.

We are interested in particular subsets of $\mathcal{B}$, called *consistent*. They have the property that the components $^-$ and $^?$ are completely determined by the two others: $^-$ is the complement of $^+$ and $^{\widehat{+}}$, while $^?$ is the difference between $^{\widehat{+}}$ and $^+$. The following definition formalizes these concepts.

Definition 3.4 (consistent set) *Let $I \subseteq \mathcal{B}$. We say that I is consistent if*

$$I^- = \mathrm{neg}(I) \ \textit{and}\ I^? = \mathrm{unc}(I),$$

where

$$\begin{array}{lrcl} \text{i)} & \mathrm{neg}(I) & = & \{ \bigwedge\{\neg c \mid c \in I_{|p(\tilde{X})} \cup I_{|\widehat{p(\tilde{X})}}\} \,\square\, \neg p(\tilde{X}) \ \mid p \in \Pi_B\} \\ \text{ii)} & \mathrm{unc}(I) & = & \{ \bigwedge\{\neg c \mid c \in I_{|p(\tilde{X})}\} \wedge d \,\square\, ?p(\tilde{X}) \ \mid p \in \Pi_B \ \textit{and}\ d \,\square\, \widehat{p(\tilde{X})} \in I\} \end{array}$$

We denote by $\mathcal{R}$ the set of consistent subsets of $\mathcal{B}$.

Example 3.5 *The set $I = \{(X{=}a) \,\square\, p(\tilde{X}),\ (X{=}b) \,\square\, \widehat{p(\tilde{X})},\ (X{\neq}a) \wedge (X{\neq}b) \,\square\, \neg p(\tilde{X}),\ (X{=}b) \,\square\, ?p(\tilde{X})\}$ is consistent.*

Note that, for any consistent set I, $[I^+] \cup [I^-]$ is a standard three-valued interpretation. The following lemma shows that $[I^?]$ are exactly the undefined elements in $[I^+] \cup [I^-]$.

Lemma 3.6 *Let I be a consistent set. Then,*

$$\bigvee I_{|?p(\tilde{X})} \equiv \neg \bigvee (I_{|p(\tilde{X})} \cup I_{|\neg p(\tilde{X})})$$

Proof.
The proof is immediate by definition of neg(I) and unc(I). ■

We propagate and extend the information contained in a consistent set I through the normal program P by means of an unfolding operation. The result of such an unfold is a consistent set too.

Definition 3.7 (I-unfolding) *Let P be a normal program and $I \in \mathcal{R}$ a consistent set. Moreover, let $C:\ p(\tilde{X}) \leftarrow c_0 \,\square\, q_1(\tilde{t}_1), \ldots, q_n(\tilde{t}_n), \neg r_1(\tilde{s}_1), \ldots, \neg r_m(\tilde{s}_m)$. be a clause in P and*

$$\begin{array}{ll} \psi_C^{+,\widehat{+}}(I) = \{ & \exists_{-\tilde{X}}\, c \,\square\, A \mid \textit{for any } i \in [1,n] \textit{ and for any } j \in [1,m], \textit{ there exist} \\ & c_i \,\square\, Q_i \in I^+ \cup I^{\widehat{+}}, \textit{ such that either } Q_i = q_i(\tilde{X}_i) \textit{ or } Q_i = \widehat{q_i(\tilde{X}_i)} \textit{ and} \\ & d_j \,\square\, R_j \in I^- \cup I^?, \textit{ such that either } R_j = \neg r_j(\tilde{Y}_j) \textit{ or } R_j = ?r_j(\tilde{Y}_j), \\ & \qquad \textit{renamed apart,} \\ & c = (c_0 \wedge \bigwedge_{i=1}^{n}(c_i \wedge \tilde{X}_i = \tilde{t}_i) \wedge \bigwedge_{j=1}^{m}(d_j \wedge \tilde{Y}_j = \tilde{s}_j)), \\ & A = p(\tilde{X}) \textit{ if for } i \in [1,n],\ j \in [1,m],\ Q_i = q_i(\tilde{X}_i) \textit{ and } R_j = \neg r_j(\tilde{Y}_j) \\ & A = \widehat{p(\tilde{X})} \textit{ otherwise} \qquad \} \end{array}$$

The I-unfolding of P is the consistent set $\psi_P(I)$ whose positive components are

$$\psi_P(I)^{+,\widehat{+}} = \bigcup_{C \in P} \psi_C^{+,\widehat{+}}(I).$$

Observe that the previous unfolding operation returns certain information if the only information it uses is certain, otherwise it returns uncertain information.

Example 3.8 *Let P be the following normal program:*

$$\begin{array}{llll} C_1: & p(X,Y) \leftarrow\ X = f(V) \Box q(V). & C_4: & r(X) \leftarrow\ X = b \Box r(X). \\ C_2: & p(X,Y) \leftarrow\ Y = g(W) \Box \neg r(W). & C_5: & r(X) \leftarrow\ X = c \Box r(X). \\ C_3: & q(X) \leftarrow\ X = a. & C_6: & r(X) \leftarrow\ X = c. \end{array}$$

Consider the consistent set I_0 whose positive components are

$$I_0^{+} = \emptyset \qquad \textit{and} \qquad I_0^{\widehat{+}} = \{true \Box p(\widehat{X,Y}), true \Box r(\widehat{X}), true \Box q(\widehat{X})\}$$

and consequently,

$$\begin{array}{lll} I_0^{-} & = & \{false \Box \neg p(X,Y), false \Box \neg r(X), false \Box \neg q(X)\} \ \textit{and} \\ I_0^{?} & = & \{true \Box ?p(X,Y), true \Box ?r(X), true \Box ?q(X)\} \end{array}$$

Then

$$\begin{array}{ll} \psi_{C_1}^{+,\widehat{+}}(I_0) = \{\exists V.X = f(V) \Box p(\widehat{X,Y})\} & \psi_{C_4}^{+,\widehat{+}}(I_0) = \{(X = b) \Box r(\widehat{X})\} \\ \psi_{C_2}^{+,\widehat{+}}(I_0) = \{\exists W.Y = g(W) \Box p(\widehat{X,Y})\} & \psi_{C_5}^{+,\widehat{+}}(I_0) = \{(X = c) \Box r(\widehat{X})\} \\ \psi_{C_3}^{+,\widehat{+}}(I_0) = \{(X = a) \Box q(X)\} & \psi_{C_6}^{+,\widehat{+}}(I_0) = \{(X = c) \Box r(X)\} \end{array}$$

Hence the I_0-unfolding of P is composed by:

$$\begin{array}{ll} \psi_P(I_0)^{+,\widehat{+}} & = \{\exists V.X = f(V) \Box p(\widehat{X,Y}),\ \exists W.Y = g(W) \Box p(\widehat{X,Y}), \\ & \quad (X = a) \Box q(X),\ (X = c) \Box r(X),\ (X = c) \Box r(\widehat{X}),\ (X = b) \Box r(\widehat{X})\ \} \\ & \\ \psi_P(I_0)^{-} & = \{\forall VW.\ X \neq f(V) \wedge Y \neq g(W) \Box \neg p(X,Y), \\ & \quad (X \neq a) \Box \neg q(X),\ (X \neq c \wedge X \neq b) \Box \neg r(X)\ \} \\ & \\ \psi_P(I_0)^{?} & = \{\exists V.X = f(V) \Box ?p(X,Y),\ \exists W.Y = g(W) \Box ?p(X,Y), \\ & \quad false \Box ?r(X),\ (X = b) \Box ?r(X)\ \} \end{array}$$

Next we introduce a preorder $\leq$ on $\mathcal{R}$. It is intended to represents the increasing of the amount of information relative to the two components $^{+}$ (which contains certain information) and $\widehat{+}$ (which contains uncertain information). The certain information should not decrease while the uncertain should not increase. Our third requirement in the definition of $\leq$ expresses also the fact that the increase of certain information should result from the relaxation of previous uncertain information.

Definition 3.9 *Let $I, J \in \mathcal{R}$ be two consistent sets. $I \leq J$ iff for any predicate symbol $p \in \Pi_B$, the following conditions hold*

1. $I_{|p(\bar{X})} \sqsubseteq J_{|p(\bar{X})}$.
2. $\bigvee J_{|\widehat{p(\bar{X})}} \preceq \bigvee I_{|\widehat{p(\bar{X})}}$.
3. $\bigvee J_{|p(\bar{X})} \preceq \bigvee (I_{|p(\bar{X})} \cup I_{|\widehat{p(\bar{X})}})$.

The equivalence relation induced by $\leq$ will be denoted by $\simeq$.

The following lemma shows how the preorder $\leq$ is reflected to I^{-} and $I^{?}$. This supports our construction which derives negative and uncertain information from positive and divergent one.

Lemma 3.10 *Let $I, J \in \mathcal{R}$. If $I \leq J$ then for any predicate symbol $p \in \Pi_B$,*

1. $I_{|\neg p(\tilde{X})} \sqsubseteq J_{|\neg p(\tilde{X})}$
2. $\bigvee J_{|?p(\tilde{X})} \preceq \bigvee I_{|?p(\tilde{X})}$.
3. $\bigvee J_{|\neg p(\tilde{X})} \preceq \bigvee (I_{|\neg p(\tilde{X})} \cup I_{|?p(\tilde{X})})$.

Example 3.11 *Consider the consistent sets I_0 and $\psi_P(I_0)$ of Example 3.8. We have that $I_0 \leq \psi_P(I_0)$.*

The next proposition shows that the unfold operation ψ_P is monotonic wrt $\leq$.

Proposition 3.12 (monotony of ψ_P) *Let I, J be consistent sets such that $I \leq J$. Then $\psi_P(I) \leq \psi_P(J)$.*

Take two consistent sets I and J such that $I \simeq J$. Then, from $I \leq J$ and $J \leq I$ we derive that for any $p \in \Pi_B$, $I_{|p(\tilde{X})} \approx J_{|p(\tilde{X})}$, $I_{|\neg p(\tilde{X})} \approx J_{|\neg p(\tilde{X})}$, $\bigvee I_{|\widehat{p(\tilde{X})}} \equiv \bigvee J_{|\widehat{p(\tilde{X})}}$, $\bigvee I_{|?p(\tilde{X})} \equiv \bigvee J_{|?p(\tilde{X})}$. Two equivalent consistent sets conveys the same information (modulo $\simeq$). We identify them by taking equivalence classes of consistent sets as elements of our semantic domain.

Definition 3.13 (uncertain interpretation) *The set of uncertain interpretation is the quotient set of $\mathcal{R}$ wrt the equivalence $\simeq$.*

$$\mathcal{I} \ = \ \mathcal{R}/\simeq \ = \ \{I_{\simeq} \mid I \in \mathcal{R}\}$$

The ordering induced by $\leq$ on $\mathcal{I}$ will still be denoted by $\leq$.

All the subsequent definitions, lemmas and propositions will be given by selecting any representative element in equivalence class $I_{\simeq}$.

An uncertain interpretation $I_{\simeq}$ is just a denotation which conveys both certain and uncertain information present in every representative I. Definition 3.9 and Lemma 3.10 show how the partial order $\leq$ defined on $\mathcal{I}$ captures the increasing of the amount of information contained in an uncertain interpretation.
Let $I_\perp$ the consistent set, whose positive components are

$$I_\perp^{+,\widehat{+}} = \{false \,\square\, p(\tilde{X}) \mid p \in \Pi_B\} \ \cup \ \{true \,\square\, \widehat{p(\tilde{X})} \mid p \in \Pi_B\}.$$

It is easy to see that $\mathcal{I}_\perp = (I_\perp)_{\simeq}$ is the the least uncertain interpretation wrt $\leq$.

4 Bottom-up semantics

The bottom-up semantics of a normal program is the the set containing all the uncertain interpretations obtained by finite iterations of an immediate consequence operator Ψ_P. Such an operator is the natural extension to equivalence classes of the unfolding operator ψ_P.

Definition 4.1 (immediate consequence operator) *Let P be a normal program and $I_{\simeq}$ be an uncertain interpretation. Then*

$$\Psi_P(I_{\simeq}) = (\psi_P(I))_{\simeq}$$

Note that, by Proposition 3.12, Ψ_P is well defined and monotonic wrt $\leq$.

Definition 4.2 (semantics) *Let P be a normal program. The semantics $[\![P]\!]^{\mathcal{CN}}$ is defined as*

$$[\![P]\!]^{\mathcal{CN}} = \{\Psi_P{\uparrow}^k \mid k \geq 0\}.$$

Observe that, as in ([1]), this semantics is the fixpoint of the monotonic and continuous operator $\mathcal{CN} : \mathcal{P}(\mathcal{I}) \mapsto \mathcal{P}(\mathcal{I})$ on the complete lattice of sets of uncertain interpretations $\mathcal{P}(\mathcal{I})$, defined as follows: $\mathcal{CN}(S) = \{\Psi_P(I_{\simeq}) \mid I_{\simeq} \in S\} \cup \{\mathcal{I}_\perp\}$.

Example 4.3 *Let P be the normal program of Example 3.8 and I_0 the consistent set there defined. Observe that $\mathcal{I}_\perp = (I_0)_{\simeq}$. Therefore $\psi_P(I_0)$ is a representative of $\Psi_P{\uparrow}^1$. Let us calculate $\Psi_P{\uparrow}^2$. For the sake of simplicity, we consider each predicate separately and let a set of constraints C denote the equivalence class $C_{\approx}$. Observe that $\Psi_P{\uparrow}^2$ projected on $p(X,Y)$ can be calculated using $(\Psi_P{\uparrow}^1)_{|q(X)}$ and $(\Psi_P{\uparrow}^1)_{|\widehat{q(X)}}$ on the first clause, $(\Psi_P{\uparrow}^1)_{|\neg r(X)}$ and $(\Psi_P{\uparrow}^1)_{|?r(X)}$ on the second:*

$$\begin{aligned}(\Psi_P{\uparrow}^2)_{|p(X,Y)} &= \{X\!=\!f(a),\ \exists W.\ Y\!=\!g(W) \wedge W\!\neq\! b \wedge W\!\neq\! c\},\\ (\Psi_P{\uparrow}^2)_{|\widehat{p(X,Y)}} &= \{Y\!=\!g(b)\}.\end{aligned}$$

Hence

$$\begin{aligned}(\Psi_P{\uparrow}^2)_{|\neg p(X,Y)} &= \{X\!\neq\! f(a) \wedge (Y\!=\!g(c) \vee \forall W.\ Y\!\neq\! g(W))\},\\ (\Psi_P{\uparrow}^2)_{|?p(X,Y)} &= \{X\!\neq\! f(a) \wedge Y\!=\!g(b)\}.\end{aligned}$$

It is easy to see that the projections on q and r are the same in $\Psi_P{\uparrow}^1$ and $\Psi_P{\uparrow}^2$. At the third level of iteration also the projections on p of $\Psi_P{\uparrow}^3$ and $\Psi_P{\uparrow}^2$ are equal.

The next proposition shows soundness and completeness of the Fitting's operator. Let $\langle I_{\simeq}\rangle$ be the partial interpretation associated to $I_{\simeq}$, namely $\langle I_{\simeq}\rangle = [I^+] \cup [I^-]$, and Φ_P be the operator defined by Fitting [12]. Φ_P is a monotonic operator on the set of partial interpretation, ordered by set inclusion. We prove that, for any finite k, there is a one-to-one correspondence between $\langle \Psi_P{\uparrow}^k\rangle$ and $\Phi_P{\uparrow}^k$.

Proposition 4.4 *Let P be a normal program and Φ_P be Fitting's operator [12]. Then*

$$\langle \Psi_P{\uparrow}^k\rangle = \Phi_P{\uparrow}^k, \qquad \textit{for any finite } k.$$

Using our terminology, Theorem 6 of [24] (which is a generalization of Theorem 6.3 in [17] for languages other than those with infinitely many function symbols of all arities) can be stated for CLP as follows. The notation $\mathcal{T} \wedge P^* \models_3 S$ shows that the sentence S is a *three-valued* logical consequence of the theory $\mathcal{T} \wedge P^*$. In the following, given the structure $\mathcal{D}$, we denote by $\mathcal{T}$ the theory corresponding to $\mathcal{D}$.

Theorem 4.5 (correctness and completeness of Φ_P) *[24] Let P be a normal program over the structure $\mathcal{D}$ and let S be a sentence. Then the following are equivalent:* 1) $\mathcal{T} \wedge P^* \models_3 S$; 2) S *has truth value true in $\Phi_P{\uparrow}^k$, for a finite k.*

Corollary 4.6 (correctness and completeness of Ψ_P) *Let P be a normal program over the structure $\mathcal{D}$ and let S be a sentence. $\mathcal{T} \wedge P^* \models_3 S$ iff S has truth value true in $\langle \Psi_P{\uparrow}^k\rangle$ for a finite k.*

4.1 Computing success and failure answers

The aim of this section is to show that our semantics can also be used to perform an effective bottom-up computation of the answers of a normal query. Given a normal query, we can derive both a success set and a failure set of constraints from every element of the chain $\Psi_P\uparrow^k$.

First, we define the success set of a query G wrt a representative I of an uncertain interpretation $I_{\simeq}$.

Definition 4.7 (success set) *Let $I \in \mathcal{R}$ be a representative of an uncertain interpretation, and G be a query. Moreover, let ans be a new predicate symbol, ans $\notin \Pi_B$, and $\{\tilde{X}\} = FV(G)$. Then the success set of G wrt I, $\mathcal{S}_G(I)$, is defined as*

$$\mathcal{S}_G(I) = \{c \mid c \,\Box\, ans(\tilde{X}) \in \psi^{+}_{ans(\tilde{X})\leftarrow G}(I)\}$$

The next lemma shows that we may extend Definition 4.7 to interpretations.

Lemma 4.8 *Let I and J be representatives of uncertain interpretations such that $I \leq J$ and let G be a query. Then, $\mathcal{S}_G(I) \sqsubseteq \mathcal{S}_G(J)$.*

Proof.
Straightforward by Proposition 3.12. ■

Then, by using the equivalence $\approx$ on sets of constraints introduced in Section 2, we define: $\mathcal{S}_G(I_{\simeq}) = \mathcal{S}_G(I)_{\approx}$, and, with the convention that a set of constraints C denotes the equivalence class $C_{\approx}$, we just write $\mathcal{S}_G(I)$.

Example 4.9 *Consider the program of Example 3.8. The constraint $\forall V.\ X \neq f(V) \wedge Y = c$ is in the success set of the query $true \,\Box\, \neg p(X,Y), r(Y)$ wrt the interpretation $\Psi_P\uparrow^1$.*

The failure set of a query G wrt a representative of an uncertain interpretation $I_{\simeq}$ contains the conjunction of the constraint of the query with the constraints in the success set of the negation of one of the literals in G. (Note that we simplify double negations).

Definition 4.10 *Let I be a representative of an uncertain interpretation and $G:\ c_0 \,\Box\, L_1, \ldots, L_n$ be a query. The failure set of G wrt I, $\mathcal{F}_G(I)$ is defined as*
$\mathcal{F}_G(I) = \{c \mid c = c_0 \wedge c_i \text{ where } i \in [1,n], \text{ and } c_i \in \mathcal{S}_{\neg L_i}(I)\}$

The next lemma shows that we may extend Definition 4.10 to interpretations.

Lemma 4.11 *Let I and J be representatives of uncertain interpretations such that $I \leq J$ and let G be a query. Then, $\mathcal{F}_G(I) \sqsubseteq \mathcal{F}_G(J)$.*

Proof.
Straightforward by Lemma 4.8. ■

Then, we define, $\mathcal{F}_G(I_{\simeq}) = \mathcal{F}_G(I)/_{\approx}$.

Example 4.12 *Consider the program of Example 3.8. The constraint $Y \neq c \wedge Y \neq b$ is in the failure set of the query $true \,\Box\, \neg p(X,Y), r(Y)$ wrt the interpretation $\Psi_P\uparrow^1$.*

The following proposition proves soundness and completeness of the previous definitions of success and failure sets wrt the Fitting's operator.

Proposition 4.13 *Let P be a normal program, Φ_P be Fitting's operator and $G: c_0 \Box \tilde{L}$ be a query. Then*

- *if $c \in \mathcal{S}_G(\Psi_P\uparrow^k)$ (resp. $c \in \mathcal{F}_G(\Psi_P\uparrow^k)$) then for any $G' \in [c \Box \tilde{L}]$, $\Phi_P\uparrow^k(G') = true$ (resp. $\Phi_P\uparrow^k(G') = false$),*
- *if $G' \in [c_0 \Box \tilde{L}]$ and $\Phi_P\uparrow^k(G') = true$ (resp. $\Phi_P\uparrow^k(G') = false$) then there exists $c \in \mathcal{S}_G(\Psi_P\uparrow^k)$ (resp. $c \in \mathcal{F}_G(\Psi_P\uparrow^k)$) such that $G' \in [c \Box \tilde{L}]$,*

where $[c \Box \tilde{L}]$ denotes the set of formulas $\{\tilde{L}\vartheta \mid \vartheta$ is a solution of $c\}$.

The following theorems show the correctness and the completeness of Ψ_P with respect to the 3-valued logical consequences of $\mathcal{T} \wedge P^*$. Similar theorems were proved for constructive negation in [25].

Theorem 4.14 (correctness) *Let P be a normal program over the structure $\mathcal{D}$ and $G : c_0 \Box L_1, \ldots, L_m$ be a query. If there exists a finite k such that $c \in \mathcal{S}_G(\Psi_P\uparrow^k)$ then $\mathcal{T} \wedge P^* \models_3 \forall(c \rightarrow L_1 \wedge \ldots \wedge L_m)$.*

Theorem 4.15 (completeness) *Let P be a normal program over the structure $\mathcal{D}$, and $G : c_0 \Box L_1, \ldots, L_m$ be a query. If $\mathcal{T} \wedge P^* \models_3 \forall(c_0 \rightarrow L_1 \wedge \ldots \wedge L_m)$ then there exists a finite k such that $\mathcal{T} \models \forall(c_0 \leftrightarrow \bigvee \mathcal{S}_G(\Psi_P\uparrow^k))$.*

After proving that our semantics captures all the correct success and failure constraints, we show how this bottom-up computation can be made effective. The idea is that every element of the chain conveys correct information that is refined by the next element of the chain. This is proved by the next corollary of Lemmas 4.8 and 4.11:

Corollary 4.16 *Let P be a normal program and G be a query. Then for any $k \geq n$,*

$$\mathcal{S}_G(\Psi_P\uparrow^n) \sqsubseteq \mathcal{S}_G(\Psi_P\uparrow^k) \text{ and } \mathcal{F}_G(\Psi_P\uparrow^n) \sqsubseteq \mathcal{F}_G(\Psi_P\uparrow^k)$$

The last point is to show how we can recognize that we have reached full information and hence stop. Let G be a query and let C be the clause $ans(\tilde{X}) \leftarrow G$, where $\tilde{X}$ are all the free variables occurring in G and ans is a new predicate symbol. Then, for any uncertain interpretation $I_{\simeq}$, $\Psi_{\{C\}}(I_{\simeq})_{|ans(\tilde{X})} = \mathcal{S}_G(I_{\simeq})$ and $\Psi_{\{C\}}(I_{\simeq})_{|\neg ans(\tilde{X})} = \bigvee \mathcal{F}_G(I_{\simeq})$. Therefore, by Lemma 3.6, the bottom-up computation stops as soon as: $\Psi_{\{C\}}(I_{\simeq})_{|?ans(\tilde{X})} = \{false\}$.

Let us apply the previous ideas to a couple of examples.

Example 4.17 *Consider the program of Example 3.8 and the query*

$$G: \neg p(g(Z), f(Z)), q(Z).$$

Then $\mathcal{S}_G(\Psi_P\uparrow^1) = \{Z = a\}$, $\mathcal{F}_G(\Psi_P\uparrow^1) = \{Z \neq a\}$ and $(\Psi_{\{ans(Z) \leftarrow G\}}(\Psi_P\uparrow^1))_{|?ans(Z)} = \{false\}$. Hence the first step is completely informative and then $Z = a$ is the only answer to the query G while $Z \neq a$ is the only answer to the query $\neg G$.

Example 4.18 *Consider again the program of Example 3.8 and the query*

$$G': p(S, g(T)), q(T).$$

The first steps gives $\mathcal{S}_{G'}(\Psi_P\uparrow^1) = \{false\}$ and $(\Psi_{\{ans(S,\, T) \leftarrow G'\}}(\Psi_P\uparrow^1))_{|?ans(S,\, T)} = \{\exists V. S = f(V) \wedge T = a;\ T = a\}$. Then we exploit also the second step that gives us $\mathcal{S}_{G'}(\Psi_P\uparrow^2) = \{S = f(a) \wedge T = a;\ T = a\}$ and $(\Psi_{\{ans(S,\, T) \leftarrow G'\}}(\Psi_P\uparrow^2))_{|?ans(S,\, T)} = \{false\}$. This terminates the computation.

5 Comparison with other approaches

We comment the information present in our semantics by relating it to some other approaches to constructive negation. We shall give the intuition that $[\![P]\!]^{\mathcal{CN}}$ captures and models computational features (like termination properties or answer substitutions as in [8, 1]) of some operational semantics for constructive negation recently proposed.

First we consider the operational semantics defined in [5, 22]. These semantics exploit a generalization to normal programs of the following idea: in a definite program, if the atomic query A has a finite set of answers $\delta_1, \ldots, \delta_n$ then $A \leftrightarrow \delta_1 \vee \ldots \vee \delta_n$ is a consequence of Clark's completion. Thus also $\neg A \leftrightarrow \neg\delta_1 \wedge \ldots \wedge \neg\delta_n$ is a consequence of Clark's completion, and if we "normalize" $\neg\delta_1 \wedge \ldots \wedge \neg\delta_n$ into an equivalent disjunction $\sigma_1 \vee \ldots \vee \sigma_m$ then each σ_i is an answer for the query $\neg A$. This method is correct with respect to the completion of the program, it is a natural generalization of negation as failure and its implementation is very simple: every time a negated atom $\neg A$ is found, the complete tree for A is built and the answers are negated. Unfortunately it is incomplete: if the tree is infinite then the program enters an infinite loop and correct solutions are missed. To see that, consider the query $\mathit{true} \,\Box\, \neg r(X)$ in the program of Example 3.8. The SLD-tree of the query $\mathit{true} \,\Box\, r(X)$ is infinite with an infinite set of answers $X = c$. The answer $X \neq b \wedge X \neq c$ (which is captured by $(\Psi_P \uparrow^1)_{|\neg r(X)}$) is a consequence of the program completion but is missed by the previous method. Notice that $\Psi_P \uparrow^3{}_{|\widehat{r(X)}} = \{X = c,\ X = b\}$ captures the divergent computations of the predicate r: a query $\sigma \,\Box\, r(Y)$ has an infinite derivation in every SLD-tree if and only if the constraint $\sigma \wedge (X = c \vee X = b)$ is satisfiable [2].

A second group of methods has been proposed to deal also with infinite trees and it can be understood as using the completion of a program instead of the program itself. The basic concept of [26, 18, 6, 25, 23, 21, 3, 4] is, roughly speaking, that in a derivation step for $\neg p(\tilde{t})$ the completed definition of the predicate p is used: $\neg p(\tilde{t})$ can be replaced by the negated right hand side of the completed definition of the predicate itself (with an appropriate mgu applied). Another extension of [6, 25] is that not only literals can be selected but also some negated formula. Both previous rules are defined in details and in fact Chan has already implemented its rule in the Sepia Prolog compiler. Stuckey proves the completeness of his rule (in the more general framework of constraint logic programming) wrt the three-valued consequences of the completion of the program.

We sketch Stuckey's method, so to explain it and to introduce some motivations to Drabent's subsequent work. Stuckey's rule states that if a negative query $\neg G$ is selected, then the subsidiary tree for G is exploited till a fixed depth (depth 1 if the rule is the Depth Front one). If $F = \{\sigma_1 \,\Box\, \tilde{B}_1, \ldots, \sigma_n \,\Box\, \tilde{B}_n\}$ is the frontier of this tree then $P^* \wedge \mathcal{T} \models_3 G \leftrightarrow \exists_{-\tilde{X}}(\sigma_1 \wedge \tilde{B}_1) \vee \ldots \vee \exists_{-\tilde{X}}(\sigma_n \wedge \tilde{B}_n)$, where $\tilde{X} = FV(G)$ and hence $P^* \wedge \mathcal{T} \models_3 \neg G \leftrightarrow \neg\exists_{-\tilde{X}}(\sigma_1 \wedge \tilde{B}_1) \wedge \ldots \wedge \neg\exists_{-\tilde{X}}(\sigma_n \wedge \tilde{B}_n)$. In order to simplify the right hand side of the previous formula into a normal disjunctive form while retaining completeness, Stuckey propose to separate the constraints in the following way:

$$\neg\exists_{-\tilde{X}}(\sigma \wedge \tilde{B}) \quad \leftrightarrow \quad \neg\exists_{-\tilde{X}}\sigma \ \vee\ \neg\exists_{-\tilde{X}}(\sigma \wedge \tilde{B})$$

If we consider the formula obtained so far

$$\neg G \quad \leftrightarrow \quad (\neg\exists_{-\tilde{X}}\sigma_1 \vee \neg\exists_{-\tilde{X}}(\sigma_1 \wedge \tilde{B}_1)) \ \wedge\ \ldots\ \wedge\ (\neg\exists_{-\tilde{X}}\sigma_n \vee \neg\exists_{-\tilde{X}}(\sigma_n \wedge \tilde{B}_n))$$

[2]This property follows by the fact that if we deal with definite programs, the Ψ functional is equivalent to the one proposed by Delzanno and Martelli in [8].

we can see that computing the disjunctive form of the right hand side and adding every disjunct as a son of the query $\neg G$ in its derivation tree is correct and complete. The problem pointed out by Drabent is that doing so we add (in general) 2^n sons. Even worse, each $\neg\exists_{-\tilde{X}}\,(\sigma_i \wedge \tilde{B}_i)$ occurs in 2^{n-1} of them, producing in general repeated computations. Moreover, subqueries can become really very complex, and the method seem to produce a lot of redundant solutions. The method of Chan shares the same problems.

A solution to these drawbacks might be found in the approach recently proposed by Drabent [11, 10] for arbitrary normal programs. It is based on the construction of failed trees. If finitely failed trees are concerned then it is sound and complete wrt Clark completion in 3-valued logic. His idea is a generalization of Chan's first method: since correct answers for $\mathit{true}\,\Box\,\neg G$ are constraints σ such that $\sigma\,\Box\,G$ has a finitely failed tree, Drabent proposes to "build" such trees. Every time the selected literal is negative, $\neg A$, a constraint γ is searched such that $\gamma\,\Box\,A$ has a (subsidiary) failed SLDFA-tree. Hence γ is an answer for the query $\mathit{true}\,\Box\,\neg A$. The (subsidiary) failed SLDFA-tree for $\gamma\,\Box\,A$ is a finitely failed tree built in this way: if the selected literal is positive, then the usual step of SLD-derivation is applied. If the selected literal in the node $\theta\,\Box\,\neg B, \tilde{D}$ is the negative literal $\neg B$, then this node has children $\sigma_1\,\Box\,\tilde{D}; \ldots; \sigma_m\,\Box\,\tilde{D}$ provided that there exist $\delta_1, \ldots, \delta_n$ SLDFA-computed-answers for the query $\theta\,\Box\,B$, and $CET \models \theta \rightarrow \delta_1 \vee \ldots \vee \delta_n \vee \sigma_1 \vee \ldots \vee \sigma_m$. The justification of this step is the following: since our aim is to falsify $\theta\,\Box\,\neg B, \tilde{D}$, then each δ_i is rejected because it implies $\theta\,\Box\,B$ and hence it already implies falsity of $\theta\,\Box\,\neg B, \tilde{D}$. If we build a failed subtree for each $\sigma_i\,\Box\,\tilde{D}$ then each σ_i implies $\neg\tilde{D}$ which implies falsity of $\theta\,\Box\,\neg B, \tilde{D}$. Drabent does not fully specify how to find the constraint γ, but he informally describes a method based on the construction of a pre-failed tree for A: a tree built using the two derivation steps of a failed SLDFA-tree, but not necessarily failed or fully expanded. Thus γ is the constraint that cuts all successful and unexpanded branches at a finite depth. Let us give the intuition of his idea applying SLDFA-resolution to the query $\mathit{true}\,\Box\,\neg p(X,Y)$. We start the subsidiary pre-failed tree for the query $\mathit{true}\,\Box\,p(X,Y)$. After expanding every branch of one level, we obtain the frontier of depth one:

$$\begin{tikzcd}
 & \mathit{true}\,\Box\,p(X,Y) \arrow[dl] \arrow[dr] & \\
X = f(V)\,\Box\,q(V) & & Y = g(W)\,\Box\,\neg r(W)
\end{tikzcd}$$

Since the negation of the disjunction of the constrains of the frontier is the satisfiable constraint $\{\forall VW.\ X \neq f(V) \wedge Y \neq g(W)\}$, we can already build a failed SLDFA-tree and add a successful son to $\mathit{true}\,\Box\,\neg p(X,Y)$:

$$\begin{tikzcd}
\mathit{true}\,\Box\,\neg p(X,Y) \arrow[d] \\
\forall VW.\ X \neq f(V) \wedge Y \neq g(W)
\end{tikzcd}$$

Notice that the first frontier of the tree for the query $\mathit{true}\,\Box\,p(X,Y)$ corresponds to $(\Psi_P{\uparrow}^1)_{|p(\widehat{X,Y})}$, while the first answer to $\mathit{true}\,\Box\,\neg p(X,Y)$ is in $(\Psi_P{\uparrow}^1)_{|\neg p(X,Y)}$. Notice also that according to our semantics $X \neq f(a) \wedge (Y = g(c) \vee \forall W. Y \neq g(W))$ is an answer for the query $\mathit{true}\,\Box\,\neg p(X,Y)$.

To find it, we must expand the subsidiary pre-failed tree to another level. The first branch finds the success constraint $X = f(a)$. In the second branch the negative atom $\neg r(W)$ is selected and hence we begin the tree for $r(W)$ which is exploited till the first frontier:

$$\begin{tikzcd}
 & \mathit{true} \,\square\, r(W). \arrow[dl] \arrow[d] \arrow[dr] & \\
W = c. & W = c \,\square\, r(W). & W = b \,\square\, r(W).
\end{tikzcd}$$

Since $W = c$ is an answer to the query $\mathit{true} \,\square\, r(W)$ and $CET \models \mathit{true} \rightarrow W = c \vee W \neq c$ then the following is a pre-failed tree for the query $\mathit{true} \,\square\, p(X,Y)$:

$$\begin{tikzcd}
 & \mathit{true} \,\square\, p(X,Y) \arrow[dl] \arrow[dr] & \\
X = f(V) \,\square\, q(V) \arrow[d] & & Y = g(W) \,\square\, \neg r(W) \arrow[d] \\
X = f(a) & & Y = g(W) \wedge W \neq c
\end{tikzcd}$$

The negation of the disjunction of the constrains of the frontier of this pre-failed tree is the satisfiable constraint $X \neq f(a) \wedge \forall W.(Y \neq g(W) \vee W = c)$. Since it produces a failed SFDFA-tree, it is added to the tree $\mathit{true} \,\square\, \neg p(X,Y)$ as a new successful son:

$$\begin{tikzcd}
 & \mathit{true} \,\square\, \neg p(X,Y) \arrow[dl] \arrow[dr] & \\
\forall VW.\ X \neq f(V) \wedge Y \neq g(W) & & X \neq f(a) \wedge \forall W.(Y \neq g(W) \vee W = c)
\end{tikzcd}$$

Now we continue to expand the tree $\mathit{true} \,\square\, r(W)$ so to find new (pre-)failed tree for $\mathit{true} \,\square\, p(X,Y)$ and hence new answers to the query $\mathit{true} \,\square\, \neg p(X,Y)$, and we enter an infinite loop. Note that this loop is captured by $(\Psi_P\!\uparrow^2)_{|?p(X,Y)} = \{X \neq f(a) \wedge Y = g(b)\}$. In fact, $\Psi_P\!\uparrow^2$ is the least fixpoint of Ψ_P, hence all elements in $[X \neq f(a) \wedge Y = g(b) \,\square\, p(X,\, Y)]$ are undefined in the completion of the program. Then any sound computation rule should produce an infinite derivation on all queries $\sigma \,\square\, \neg p(X,Y)$ such that $\sigma \wedge X \neq f(a) \wedge Y = g(b)$ is satisfiable.

We end up this section with a conjecture that is also our future work. This conjecture expresses our belief about the existence of a strong relationship between this bottom-up semantics and the SLDFA-derivation.

Conjecture 5.1 *Let P be a normal program, G the query $c_0 \,\square\, \tilde{L}$, and $c \,\square\, p(\tilde{X})$ an atomic query. For any representative $\psi_P\!\uparrow^n$ of $\Psi_P\!\uparrow^n$, $n \geq 0$,*

1. $\{c \mid$ *there exists an SLDFA-refutation for $P \cup \{G\}$, with answer c*$\}$
$\approx \bigcup_{n \geq 0} \mathcal{S}_G(\psi_P\!\uparrow^n)$.

2. $\{c \mid c \preceq c_0$ *and there exists a finitely failed SLDFA-tree for $P \cup$* $\{c \,\square\, \tilde{L}\}\ \}$
$\approx \bigcup_{n \geq 0} \mathcal{F}_G(\psi_P\!\uparrow^n)$.

3. *$c \,\Box\, p(\tilde{X})$ has an infinite SLDFA-derivation under any computation rule iff for every $n \geq 0$ there exists $c' \in (\psi_P \!\uparrow^n)_{|\widetilde{p(\tilde{X})}}$ such that $c \wedge c'$ is satisfiable.*

4. *$c \,\Box\, \neg p(\tilde{X})$ has an infinite SLDFA-derivation under any computation rule iff for every $n \geq 0$ there exists $c' \in (\psi_P \!\uparrow^n)_{|?p(\tilde{X})}$ such that $c \wedge c'$ is satisfiable.*

References

[1] A. Bossi, M. Bugliesi, and M. Fabris. A New Fixpoint Semantics for Prolog. In D. Warren, editor, *Proceeding of the Tenth Int. Conf. on Logic Programming, ICLP'93*, pages 374–389. The MIT Press, 1993.

[2] A. Bossi, M. Fabris, and M. C. Meo. A Bottom-up Semantics for Constructive Negation. In P. Van Hentenryck, editor, *Proceeding of the Eleventh Int. Conf. on Logic Programming, ICLP'94*, pages 520–534. The MIT Press, 1994.

[3] A. Bottoni, G. Levi. Computing in the Completion. In D. Saccá, editor, *Proceeding of the Eight Italian Conf. on Logic Programming*, pages 375–389. Mediterranean Press, 1993.

[4] A. Bottoni, G. Levi. The Inverse of Fitting's Functional. In G. Gottlob, A. Leitsch and D. Mundici, editors, *Proceeding Third Kurt Gödel Colloquium, Computational and Proof Theory, KGC'93*, pages 132–143. Springer-Verlag, 1993.

[5] D. Chan. Constructive Negation Based on the Completed Database. In R. A. Kowalski and K. A. Bowen, editors, *Proc. Fifth Int'l Conf. on Logic Programming*, pages 111–125. The MIT Press, Cambridge, Mass., 1988.

[6] D. Chan. An Extension of Constructive Negation and its Application in Coroutining. In E. Lusk and R. Overbeek, editors, *Proc. North American Conf. on Logic Programming'89*, pages 477–493. The MIT Press, Cambridge, Mass., 1989.

[7] K. L. Clark. Negation as Failure. In H. Gallaire and J. Minker, editors, *Logic and Data Bases*, pages 293–322. Plenum Press, New York, 1978.

[8] G. Delzanno and M. Martelli. A bottom-up characterization of finite success. Technical Report, Università di Genova, DISI, 1992.

[9] G. Delzanno and M. Martelli. S-semantica per modellare insiemi di soluzioni. In S. Costantini, editor, *Proc. Seventh Italian Conference on Logic Programming*, pages 191–205, 1992.

[10] W. Drabent. SLS-resolution without floundering. In L. M. Pereira and A. Nerode, editors, *Proc. of the Workshop on Logic Programming and Non-monotonic reasoning*, pages 82–98, 1993.

[11] W. Drabent. What is Failure? An Approach to Constructive Negation. *Acta Informatica*, 1993. To appear.

[12] M. Fitting. A Kripke-Kleene semantics for logic programs. *Journal of Logic Programming*, 2:295–312, 1985.

[13] M. Falaschi, G. Levi, M. Martelli, and C. Palamidessi. A new Declarative Semantics for Logic Languages. In R. A. Kowalski and K. A. Bowen, editors, *Proc. Fifth Int'l Conf. on Logic Programming*, pages 993–1005. The MIT Press, Cambridge, Mass., 1988.

[14] M. Gabbrielli and G. Levi. Modeling Answer Constraints in Constraint Logic Programs. In K. Furukawa, editor, *Proc. Eighth Int'l Conf. on Logic Programming*, pages 238– 252. The MIT Press, Cambridge, Mass., 1991.

[15] 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.

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

[17] K. Kunen. Negation in logic programming. *Journal of Logic Programming*, 4:289–308, 1987.

[18] D. Lugiez. A Deduction Procedure for First Order Programs. In *Proc. of Sixth International Conf. on Logic Programming*, Lisbon, pages 585–599, MIT Press, 1989.

[19] P. Mancarella, S. Martini, and D. Pedreschi. Complete Logic Programs with Domain Closure Axiom. *Journal of Logic Programming*, 5(3):263–276, 1988.

[20] M.J. Maher. Logic Semantics for a Class of Committed-Choice Programs. In *Proc. of Fourth International Conf. on Logic Programming*, pages 858–876, MIT Press, 1987.

[21] J. A. Plaza. Fully Declarative Logic Programming. In *Programming Language Implementation and Logic Programming, Proceedings 1992*, pages 415–427, Springer-Verlag, 1992, LNCS 631.

[22] T. Przymusinsky. On Constructive Negation in Logic Programming. In *Proc. North American Conference on Logic Programming*, Addendum. MIT Press, 1989.

[23] T. Sato, and F. Motoyoshi. A Complete Top-down Interpreter for First Order Logic Programs. In *Logic Programming, Proc. of the 1991 International Symposium*, pages 35–53, MIT Press, 1991.

[24] J. C. Shepherdson. Language and equality theory in logic programming. Technical Report PM-91-02, School of Mathematics, University of Bristol, 1991.

[25] P. J. Stuckey. Constructive Negation for Constraint Logic Programming. In *Proc. Sixth IEEE Symp. on Logic In Computer Science*, pages 328–339. IEEE Computer Society Press, 1991.

[26] M. Wallace. Negation by Constraints: a Sound and Efficient Implementation of Negation in Deductive Databases. In *Proc. 1987 Symposium on Logic Programming*, San Francisco, pages 253–263, August 1987.

A Appendix

Lemma 3.10 *Let $I, J \in \mathcal{R}$. If $I \leq J$ then for any predicate symbol $p \in \Pi_B$,*

1. $I_{|\neg p(\bar{X})} \sqsubseteq J_{|\neg p(\bar{X})}$
2. $\bigvee J_{|?p(\bar{X})} \preceq \bigvee I_{|?p(\bar{X})}$.
3. $\bigvee J_{|\neg p(\bar{X})} \preceq \bigvee(I_{|\neg p(\bar{X})} \cup I_{|?p(\bar{X})})$.

Proof.

1. By 2. and 3. of Definition 3.9 we obtain $\bigvee(J_{|p(\bar{X})} \cup J_{|\widehat{p(\bar{X})}}) \preceq \bigvee(I_{|p(\bar{X})} \cup I_{|\widehat{p(\bar{X})}})$, hence by definition of consistent set and by negating the constraints we derive the thesis.

2. First of all, observe that given $K \in \mathcal{R}$, by Definition 3.4 and by standard first order logic properties we have that
$$\bigvee K_{|?p(\bar{X})} \equiv \bigwedge\{\neg c \mid c \in K_{|p(\bar{X})}\} \wedge \bigvee K_{|\widehat{p(\bar{X})}}. \qquad (i)$$
Moreover, by reversing 1. of Definition 3.9,
$$\bigwedge\{\neg c \mid c \in J_{|p(\bar{X})}\} \preceq \bigwedge\{\neg c \mid c \in I_{|p(\bar{X})}\}. \qquad (ii)$$
and by 2. of Definition 3.9
$$\bigvee J_{|\widehat{p(\bar{X})}} \preceq \bigvee I_{|\widehat{p(\bar{X})}}. \qquad (iii)$$
Therefore
$$\begin{array}{ll}\bigvee J_{|?p(\bar{X})} & \equiv \text{ (by (i))} \\ \bigwedge\{\neg c \mid c \in J_{|p(\bar{X})}\} \wedge \bigvee J_{|\widehat{p(\bar{X})}} & \preceq \text{ (by (ii) and (iii))} \\ \bigwedge\{\neg c \mid c \in I_{|p(\bar{X})}\} \wedge \bigvee I_{|\widehat{p(\bar{X})}} & \equiv \text{ (by (i))} \\ \bigvee I_{|?p(\bar{X})}. & \end{array}$$

3. By Definition 3.4, $\bigvee J_{|\neg p(\bar{X})} \equiv \bigwedge\{\neg c \mid c \in J_{|p(\bar{X})} \cup J_{|\widehat{p(\bar{X})}}\}$. Moreover, by reversing 1. of Definition 3.9, $\bigwedge\{\neg c \mid c \in J_{|p(\bar{X})}\} \preceq \bigwedge\{\neg c \mid c \in I_{|p(\bar{X})}\}$ and hence we obtain $\bigvee J_{|\neg p(\bar{X})} \preceq \bigwedge\{\neg c \mid c \in I_{|p(\bar{X})}\}$.
Now, by Definition 3.4, we have that $\bigwedge\{\neg c \mid c \in I_{|p(\bar{X})}\} \equiv \bigvee(I_{|\neg p(\bar{X})} \cup I_{|?p(\bar{X})})$ and hence we derive the thesis. ■

In the following, given a valuation ϑ and a set of variables W we denote by $\vartheta_{|W}$ the restriction of ϑ to the variables in W, i.e. $\vartheta_{|W}$ is the mapping, whose domain is the set of variables W, such that $\vartheta_{|W}(X) = \vartheta(X)$ for any $X \in W$. Moreover we simplify the double negation.

Proposition 3.12 *Let I and J be uncertain interpretations such that $I \leq J$. Then $\Psi_P(I) \leq \Psi_P(J)$.*
Proof.
We have to prove that for any predicate symbol $p \in \Pi_B$ the following conditions hold

- $\Psi_P(I)_{|p(\bar{X})} \sqsubseteq \Psi_P(J)_{|p(\bar{X})}$.
  This statement follows by definition of Ψ_P, by 1. of Definition 3.9 and by 1. of Lemma 3.10.

- $\bigvee \Psi_P(J)_{|\widehat{p(\tilde{X})}} \preceq \bigvee \Psi_P(I)_{|\widehat{p(\tilde{X})}}$.
We have to prove that for any valuation ϑ, if ϑ is a solution of $\bigvee \Psi_P(J)_{|\widehat{p(\tilde{X})}}$, then ϑ is a solution of $\bigvee \Psi_P(I)_{|\widehat{p(\tilde{X})}}$. Let ϑ be a solution of $\bigvee \Psi_P(J)_{|\widehat{p(\tilde{X})}}$. Then there exists $c\,\Box\,\widehat{p(\tilde{X})} \in \Psi_P(J)$, such that ϑ is a solution of c. By definition of Ψ_P, there exists a clause, $C:\ p(\tilde{X}) \leftarrow c_0\Box q_1(\tilde{t}_1),\dots,q_n(\tilde{t}_n),\neg r_1(\tilde{s}_1),\dots,\neg r_m(\tilde{s}_m) \in P$, such that
for any $i \in [1,n]$, there exists $c_i\Box Q_i \in J^+ \cup \widehat{J^+}$,
for any $j \in [1,m]$, there exists $d_j\Box R_j \in J^- \cup J^?$,
renamed apart, as specified in Definition 4.1, such that
$c = \exists_{-\tilde{X}}(c_0 \wedge \bigwedge_{i=1}^n (c_i \wedge \tilde{X}_i = \tilde{t}_i) \wedge \bigwedge_{j=1}^m (d_j \wedge \tilde{Y}_j = \tilde{s}_j))$.
Since ϑ is a solution of c, there exists a valuation σ, such that $\sigma_{|\tilde{X}} = \vartheta_{|\tilde{X}}$, σ is a solution of c_0, for any $i \in [1,n]$, $j \in [1,m]$, σ is a solution of $c_i \wedge \tilde{X}_i = \tilde{t}_i$ and of $d_j \wedge \tilde{Y}_j = \tilde{s}_j$.
By 2. and 3. of Definition 3.9, for any $i \in [1,n]$, there exist $c_i'\Box Q_i' \in I^+ \cup \widehat{I^+}$ such that σ is a solution of c_i' and either $Q_i' = q_i(\tilde{X}_i)$ or $Q_i' = \widehat{q_i(\tilde{X}_i)}$. Moreover by 2. and 3. of Lemma 3.10, for any $j \in [1,m]$, there exist $d_j'\Box R_j' \in I^- \cup I^?$, such that σ is a solution of d_j' and either $R_j' = r_j(\tilde{Y}_j)$ or $R_j' = ?r_j(\tilde{Y}_j)$.
Note that if there exists $i \in [1,n]$, such that $c_i\Box Q_i \in \widehat{J^+}$, then, by Definition 3.9, we can choose $c_i'\Box Q_i' \in \widehat{I^+}$. Analogously if there exists $j \in [1,m]$, such that $d_j\Box R_j \in J^?$. Then σ is a solution of d where d is equal to
$\exists_{-\tilde{X}}(c_0 \wedge \bigwedge_{i=1}^n (c_i' \wedge \tilde{X}_i = \tilde{t}_i) \wedge \bigwedge_{j=1}^m (d_j' \wedge \tilde{Y}_j = \tilde{s}_j))$
and, by previous observation, $d \in \Psi_P(I)_{|\widehat{p(\tilde{X})}}$. Since $\sigma_{|\tilde{X}} = \vartheta_{|\tilde{X}}$ we have that ϑ is a solution of d and this completes the proof.

- $\bigvee \Psi_P(J)_{|p(\tilde{X})} \preceq \bigvee(\Psi_P(I)_{|p(\tilde{X})} \cup \Psi_P(I)_{|\widehat{p(\tilde{X})}})$.
Analogous to the previous one. ∎

The Propositions A.1, A.2 and A.3 are all immediate consequence of the definition of uncertain interpretation. Note that, given an uncertain interpretation $I_{\simeq}$, the notation $\langle I_{\simeq}\rangle$ denotes the partial interpretation associated to $I_{\simeq}$, namely $\langle I_{\simeq}\rangle = [I^+] \cup [I^-]$.

Proposition A.1 *Let $I_{\simeq}$ be an uncertain interpretation.*
$p(\tilde{t}) \in \langle I_{\simeq}\rangle$ implies $\neg p(\tilde{t}) \notin \langle I_{\simeq}\rangle$ and $\neg p(\tilde{t}) \in \langle I_{\simeq}\rangle$ implies $p(\tilde{t}) \notin \langle I_{\simeq}\rangle$.

Proposition A.2 *Let $I_{\simeq}$ be an uncertain interpretation. If $p(\tilde{t}) \notin \langle I_{\simeq}\rangle$ and $\neg p(\tilde{t}) \notin \langle I_{\simeq}\rangle$, then*

i) *there exists $c\,\Box\,\widehat{p(\tilde{X})} \in I$ and there exists a solution ϑ of c such that $p(\tilde{X})\vartheta = p(\tilde{t})$.*

ii) *there exists $d\,\Box\,?p(\tilde{X}) \in I$ and there exists a solution σ of d such that $p(\tilde{X})\sigma = p(\tilde{t})$.*

Proposition A.3 *Let $I_{\simeq}$ be an uncertain interpretation. Then*

i) *If there exists $c\,\Box\,\widehat{p(\tilde{X})} \in I$ then for any solution ϑ of c, $\neg p(\tilde{X})\vartheta \notin \langle I_{\simeq}\rangle$.*

ii) *If there exists $d\,\Box\,?p(\tilde{X}) \in I$ then for any solution ϑ of c, $p(\tilde{X})\vartheta \notin \langle I_{\simeq}\rangle$.*

Lemma A.4 *Let P be a normal program and let I be an uncertain interpretation. $\langle\Psi_P(I)\rangle = \Phi_P(\langle I\rangle)$.*

Proof.

a) We prove that $p(\tilde{t}) \in \langle \Psi_P(I) \rangle$ iff $p(\tilde{t}) \in \Phi_P(\langle I \rangle)$.

We have the following equivalences
$p(\tilde{t}) \in \langle \Psi_P(I) \rangle$
iff (by definition of $\langle\ \rangle$)
there exists $c \,\square\, p(\tilde{X}) \in \Psi_P(I)$ and there exists a solution ϑ of c such that $p(\tilde{X})\vartheta = p(\tilde{t})$
iff (by definition of Ψ_P)
there exists a clause $p(\tilde{X}) \leftarrow c_0 \,\square\, q_1(\tilde{t}_1), \ldots, q_n(\tilde{t}_n), \neg r_1(\tilde{s}_1), \ldots, \neg r_m(\tilde{s}_m) \in P$ and for any $i \in [1,n]$, $j \in [1,m]$, there exists $c_i \,\square\, q_i(\tilde{X}_i) \in I$ and there exists $d_j \,\square\, \neg r_j(\tilde{Y}_j) \in I$, renamed apart, such that $c = \exists_{-\tilde{X}}(c_0 \wedge \bigwedge_{i=1}^{n}(c_i \wedge \tilde{X}_i = \tilde{t}_i) \wedge \bigwedge_{j=1}^{m}(d_j \wedge \tilde{Y}_j = \tilde{s}_j))$ and there exists a solution ϑ of c such that $p(\tilde{X})\vartheta = p(\tilde{t})$. Now note that

there exists a solution ϑ of c

iff (since for any $i \in [1,n]$, $j \in [1,m]$, $\tilde{X}_i$ and $\tilde{Y}_j$ are all new distinct variables)

there exists a valuation σ such that $\sigma_{|\tilde{X}} = \vartheta_{|\tilde{X}}$, σ is a solution of c_0 and for any $i \in [1,n]$, $j \in [1,m]$, σ is a solution of $c_i \wedge \tilde{X}_i = \tilde{t}_i$ and of $d_j \wedge \tilde{Y}_j = \tilde{s}_j$

iff (by definition of $\langle\ \rangle$)

there exists a valuation σ such that $\sigma_{|\tilde{X}} = \vartheta_{|\tilde{X}}$, σ is a solution of c_0 and for any $i \in [1,n]$, $j \in [1,m]$, $q_i(\tilde{t}_i)\sigma \in \langle I \rangle$ and $\neg r_j(\tilde{s}_j)\sigma \in \langle I \rangle$.

By the previous observations and by definition of Φ_P, we have that $p(\tilde{t}) \in \langle \Psi_P(I) \rangle$ *iff* $p(\tilde{X})\sigma = p(\tilde{X})\vartheta = p(\tilde{t}) \in \Phi_P(\langle I \rangle)$.

b) $\neg p(\tilde{t}) \in \langle \Psi_P(I) \rangle$ iff $\neg p(\tilde{t}) \in \Phi_P(\langle I \rangle)$.

- $\neg p(\tilde{t}) \in \langle \Psi_P(I) \rangle$ implies $\neg p(\tilde{t}) \in \Phi_P(\langle I \rangle)$.
  Assume that $\neg p(\tilde{t}) \notin \Phi_P(\langle I \rangle)$. Then by definition of Φ_P, there exists a clause $p(\tilde{X}) \leftarrow c_0 \,\square\, B_1, \ldots, B_m \in P$ and there exists a valuation α such that $p(\tilde{X})\alpha = p(\tilde{t})$, α is a solution of c_0 and for any $i \in [1,m]$, $\neg B_i\alpha \notin \langle I \rangle$.
  We have the following possibilities
  1) for any $i \in [1,m]$, $B_i\alpha \in \langle I \rangle$.
  In this case $p(\tilde{X})\alpha \in \Phi_P(\langle I \rangle)$. Then, by previous **a)**, $p(\tilde{X})\alpha \in \langle \Psi_P(I) \rangle$ and, by Proposition A.1, this contradicts the hypothesis.
  2) there exists $i \in [1,m]$, such that $B_i\alpha \notin \langle I \rangle$ and $\neg B_i\alpha \notin \langle I \rangle$.
  Without loss of generality we can suppose that $i = 1$, $B_1 = q_1(\tilde{t}_1)$ (or $B_1 = \neg q_1(\tilde{t}_1)$) and for any $j \in [2,m]$, $B_j\alpha \in \langle I \rangle$. By Proposition A.2 there exists $c_1 \,\square\, \widehat{q_1(\tilde{X}_1)} \in \langle I \rangle$ (or $c_1 \,\square\, ?q_1(\tilde{X}_1) \in \langle I \rangle$), where $\tilde{X}_1$ are new distinct variables, and there exists ϑ_1 solution of c_1, such that $q_1(\tilde{X}_1)\vartheta_1 = q_1(\tilde{t}_1)\alpha$. Now analogously to the previous **a)**, we can prove that there exists $c \,\square\, \widehat{p(\tilde{X})} \in \Psi_P(I)$ such that α is a solution of c and $p(\tilde{X})\alpha = p(\tilde{t})$. Therefore, by *i)* of Proposition A.3, $\neg p(\tilde{t}) \notin \langle \Psi_P(I) \rangle$.
- $\neg p(\tilde{t}) \in \Phi_P(\langle I \rangle)$ implies $\neg p(\tilde{t}) \in \langle \Psi_P(I) \rangle$.
  Assume that $\neg p(\tilde{t}) \notin \langle \Psi_P(I) \rangle$. Then for any $d \,\square\, \neg p(\tilde{X}) \in \Psi_P(I)$ and for any valuation ϑ such that $p(\tilde{X})\vartheta = p(\tilde{t})$, we have that ϑ is not a solution of d. By Definition 3.4, $d = \bigwedge\{\neg c \mid c \,\square\, p(\tilde{X}) \in \Psi_P(I)\} \wedge \bigwedge\{\neg c' \mid c' \,\square\, \widehat{p(\tilde{X})} \in \Psi_P(I)\}$.

Therefore for any valuation ϑ such that $p(\tilde{X})\vartheta = p(\tilde{t})$ there exists a constraint $c \in \Psi_P(I)_{|p(\tilde{X})} \cup \Psi_P(I)_{|\widehat{p(\tilde{X})}}$, such that ϑ is a solution of c.
By definition of Ψ_P, there exists a clause
$p(\tilde{X}) \leftarrow c_0 \,\Box\, q_1(\tilde{t}_1), \ldots, q_n(\tilde{t}_n), \neg r_1(\tilde{s}_1), \ldots, \neg r_m(\tilde{s}_m)$ in P and
for any $i \in [1,n]$, there exists $c_i \,\Box\, Q_i \in I^+ \cup \widehat{I^+}$,
for any $j \in [1,m]$, there exists $d_j \,\Box\, R_j \in I^- \cup I^?$,
renamed apart, as specified in Definition 4.1, such that
$c = \exists_{-\tilde{X}}(c_0 \wedge \bigwedge_{i=1}^{n}(c_i \wedge \tilde{X}_i = \tilde{t}_i) \wedge \bigwedge_{j=1}^{m}(d_j \wedge \tilde{Y}_j = \tilde{s}_j))$.
Then there exists a valuation σ such that $\vartheta_{|\tilde{X}} = \sigma_{|\tilde{X}}$, σ is a solution of c_0 and for any $i \in [1,n]$, $j \in [1,m]$, σ is a solution of $c_i \wedge \tilde{X}_i = \tilde{t}_i$ and of $d_j \wedge \tilde{Y}_j = \tilde{s}_j$.
Then we have the following possibilities
1) For any $i \in [1,n]$, $j \in [1,m]$, $c_i \,\Box\, Q_i \in I^+$ and $d_j \,\Box\, R_j \in I^-$. Then, by the previous **a**), $p(\tilde{t}) \in \Phi_P(\langle I \rangle)$ and this contradicts the hypothesis.
2) There exists $i \in [1,n]$ such that $c_i \,\Box\, Q_i \in \widehat{I^+}$.
Without loss of generality, we can assume that $i = 1$ and for any $h \in [2,n]$, $j \in [1,m]$, $c_h \,\Box\, Q_h \in I^+$ and $d_j \,\Box\, R_j \in I^-$. Then by definition of $\langle\rangle$, for any $h \in [2,n]$, $j \in [1,m]$, $q_h(\tilde{t}_h)\sigma \in \langle I \rangle$ and $\neg r_j(\tilde{s}_j)\sigma \in \langle I \rangle$. Moreover, since $c_1 \,\Box\, Q_1 \in \widehat{I^+}$ and σ is a solution of c_1, by *i)* of Proposition A.3, we have that $\neg q_1(\tilde{t}_1)\sigma \notin \langle I \rangle$.
Therefore there exists a clause
$p(\tilde{X}) \leftarrow c_0 \,\Box\, q_1(\tilde{t}_1), \ldots, q_n(\tilde{t}_n), \neg r_1(\tilde{s}_1), \ldots, \neg r_m(\tilde{s}_m)$ in P
and there exists a valuation σ such that $p(\tilde{X})\sigma = p(\tilde{t})$ and
$(c_0 \,\Box\, q_1(\tilde{t}_1), \ldots, q_n(\tilde{t}_n), \neg r_1(\tilde{s}_1), \ldots, \neg r_m(\tilde{s}_m))\sigma$
is not *false* in $\langle I \rangle$. Thus, by definition of Φ_P, $\neg p(\tilde{t}) \notin \Phi_P(\langle I \rangle)$ and this contradicts the hypothesis.
3) There exists $j \in [1,m]$ such that $c_j \,\Box\, Q_j \in I^?$.
The proof is analogous to the previous one, by using *ii)* of Proposition A.3. ■

Proposition 4.4 *Let P be a normal program. $\langle \Psi_P \uparrow^k \rangle = \Phi_P \uparrow^k$, for any finite k*
Proof.
The proof is by induction on k.
Clearly for the base case, $k = 0$, $\langle \Psi_P \uparrow^0 \rangle = \Phi_P \uparrow^0 = \emptyset$.
Inductive case: assume that the thesis holds for k and consider $k + 1$.

$$\begin{array}{lcl}
\langle \Psi_P \uparrow^{k+1} \rangle & = & \text{by definition of } \uparrow^{k+1} \\
\langle \Psi_P(\Psi_P \uparrow^k) \rangle & = & \text{by Lemma A.4} \\
\Phi_P(\langle \Psi_P \uparrow^k \rangle) & = & \text{by inductive hypothesis} \\
\Phi_P(\Phi_P \uparrow^k) & = & \text{by definition of } \uparrow^{k+1} \\
\Phi_P \uparrow^{k+1} & &
\end{array}$$

and this completes the proof. ■

Proposition 4.13 *Let P be a normal program, Φ_P be Fitting's operator and $G:\ c_0 \,\Box\, \tilde{L}$ be a query. Then*

- *if $c \in \mathcal{S}_G(\Psi_P \uparrow^k)$ (resp. $c \in \mathcal{F}_G(\Psi_P \uparrow^k)$) then for any $G' \in [c \,\Box\, \tilde{L}]$, $\Phi_P \uparrow^k(G') = true$ (resp. $\Phi_P \uparrow^k(G') = false$)*

- *if $G' \in [c_0 \Box \tilde{L}]$ and $\Phi_P \uparrow^k(G') = true$ (resp. $\Phi_P\uparrow^k(G') = false$) then there exists $c \in \mathcal{S}_G(\Psi_P\uparrow^k)$ (resp. $c \in \mathcal{F}_G(\Psi_P\uparrow^k)$) such that $G' \in [c \Box \tilde{L}]$*

where $[c \Box \tilde{L}]$ denotes the set of formulas $\{\tilde{L}\vartheta \mid \vartheta$ is a solution of $c\}$.
Proof.
In the following we assume that $\tilde{X} = FV(G)$ and $\tilde{L} = L_1, \ldots, L_n$.

- **a)** Let $c \in \mathcal{S}_G(\Psi_P\uparrow^k)$ and let $G' \in [c \Box \tilde{L}]$.
  By definition of $\mathcal{S}_G$ and of [ ], we have that $c \Box ans(\tilde{X}) \in \Psi_{ans(\tilde{X})\leftarrow G}(\Psi_P\uparrow^k)$ and there exists a solution ϑ of c such that $G' = \tilde{L}\vartheta$. Since ϑ is a solution of c and by definition of $\langle\rangle$, we have that $ans(\tilde{X})\vartheta \in \langle\Psi_{ans(\tilde{X})\leftarrow G}(\Psi_P\uparrow^k)\rangle$. Therefore, by Lemma A.4 and Proposition 4.4, $ans(\tilde{X})\vartheta \in \Phi_{ans(\tilde{X})\leftarrow G}(\Phi_P\uparrow^k)$. Then, by definition of Φ_P, $\Phi_P\uparrow^k(G') = true$.

  b) Let $c \in \mathcal{F}_G(\Psi_P\uparrow^k)$.
  By definition of $\mathcal{F}_G$, there exists $i \in [1,n]$ and there exists $c_i \in \mathcal{S}_{\neg L_i}(\Psi_P\uparrow^k)$, such that $c = c_0 \wedge c_i$. Then, by **a)**, for any solution ϑ of c we have that $\neg L_i\vartheta$ is *true* in $\Phi_P\uparrow^k$. Therefore $G' = \tilde{L}\vartheta$ is *false* in $\Phi_P\uparrow^k$ and this completes the proof.

- In the following we assume that ϑ is a solution of c_0 and $G' = \tilde{L}\vartheta$.

  c) Let $\Phi_P\uparrow^k(G') = true$.
  By definition of Φ, $ans(\tilde{X})\vartheta \in \Phi_{ans(\tilde{X})\leftarrow G}(\Phi_P\uparrow^k)$. Then, by Lemma A.4 and Proposition 4.4, $ans(\tilde{X})\vartheta \in \langle\Psi_{ans(\tilde{X})\leftarrow G}(\Psi_P\uparrow^k)\rangle$. Therefore, by definition of $\langle\rangle$, there exists $c \Box ans(\tilde{X}) \in \Psi_{ans(\tilde{X})\leftarrow G}(\Psi_P\uparrow^k)$ such that ϑ is a solution of c. By definition of $\mathcal{S}_G$ and by previous observation we have that $c \in \mathcal{S}_G(\Psi_P\uparrow^k)$. Moreover, since ϑ is a solution of c and by definition of [ ], $G' = \tilde{L}\vartheta \in [c \Box \tilde{L}]$.

  d) Let $\Phi_P\uparrow^k(G') = false$.
  By definition of Φ and since ϑ is a solution of c_0, there exists $i \in [1,n]$ such that $L_i\vartheta$ is *false* in $\Phi_P\uparrow^k$ and therefore $\neg L_i\vartheta$ is *true* in $\Phi_P\uparrow^k$. Then, by **c)**, there exists $c_i \in \mathcal{S}_{\neg L_i}(\Psi_P\uparrow^k)$ such that ϑ is a solution of c_i. By definition of $\mathcal{F}_G$ and by definition of [ ], we have that $c_0 \wedge c_i \in \mathcal{F}_G(\Psi_P\uparrow^k)$ and $G' \in [c_0 \wedge c_i \Box \tilde{L}]$. ∎

Theorem 4.14 *Let P be a normal program over the structure $\mathcal{D}$ and $G: c_0 \Box L_1, \ldots, L_m$ be a query. If there exists a finite k such that $c \in \mathcal{S}_G(\Psi_P\uparrow^k)$ then $\mathcal{T} \wedge P^* \models_3 \forall(c \to L_1 \wedge \ldots \wedge L_m)$.*
Proof.
Assume that there exists a finite k such that $c \in \mathcal{S}_G(\Psi_P\uparrow^k)$. By Proposition 4.13, for any solution ϑ of c, $\Phi_P\uparrow^k((L_1 \wedge \ldots \wedge L_m)\vartheta) = true$. Moreover, by Theorem 4.5, if there exists a finite k such that $\Phi_P\uparrow^k((L_1 \wedge \ldots \wedge L_m)\vartheta) = true$ then $\mathcal{T} \wedge P^* \models_3 (L_1 \wedge \ldots \wedge L_m)\vartheta$. Therefore, for any solution ϑ of c, $\mathcal{T} \wedge P^* \models_3 (L_1 \wedge \ldots \wedge L_m)\vartheta$ and then $\mathcal{T} \wedge P^* \models_3 \forall(c \to L_1 \wedge \ldots \wedge L_m)$.
∎

Theorem 4.15 *Let P be a normal program over the structure $\mathcal{D}$ and $G: c_0 \Box L_1, \ldots, L_m$ be a query. If $\mathcal{T} \wedge P^* \models_3 \forall(c_0 \to L_1 \wedge \ldots \wedge L_m)$ then there exists a finite k such that $\mathcal{T} \models \forall(c_0 \leftrightarrow \bigvee \mathcal{S}_G(\Psi_P\uparrow^k))$.*

Proof.
Assume that $\mathcal{T} \wedge P^* \models_3 \forall(c_0 \rightarrow L_1 \wedge \ldots \wedge L_m)$. Then, by Theorem 4.5, there exists a finite k such that $\Phi_P\uparrow^k(\forall(c_0 \rightarrow L_1 \wedge \ldots \wedge L_m))$ is *true*. Therefore, for any solution ϑ of c_0, $\Phi_P\uparrow^k((L_1 \wedge \ldots \wedge L_m)\vartheta) = true$. By Proposition 4.13, for any solution ϑ of c_0 there exists $c \in \mathcal{S}_G(\Psi_P\uparrow^k)$, such that ϑ is a solution of c. Then $\mathcal{T} \models \forall(c_0 \rightarrow \bigvee \mathcal{S}_G(\Psi_P\uparrow^k))$. Moreover, by definition of $\mathcal{S}_G$, for any $c \in \mathcal{S}_G(\Psi_P\uparrow^k)$ we have that $\mathcal{T} \models \forall(c \rightarrow c_0)$. Therefore $\mathcal{T} \models \forall(\bigvee \mathcal{S}_G(\Psi_P\uparrow^k) \rightarrow c_0)$ and this completes the proof. ■