# Well Founded Semantics for Logic Programs with Explicit Negation

**Luís Moniz Pereira and José Júlio Alferes**
*CRIA, Uninova and DCS, U. Nova de Lisboa*
*2825 Monte da Caparica, Portugal*

**Abstract.** The aim of this paper is to provide a semantics for general logic programs (with negation by default) extended with explicit negation, subsuming well founded semantics [22].

The Well Founded semantics for extended logic programs (WFSX) is expressible by a default theory semantics we have devised [11]. This relationship improves the cross–fertilization between logic programs and default theories, since we generalize previous results concerning their relationship [3, 4, 7, 1, 2], and there is an increasing use of logic programming with explicit negation for nonmonotonic reasoning [7, 15, 16, 13, 23]. It also clarifies the meaning of logic programs combining both explicit negation and negation by default. In particular, it shows that explicit negation corresponds exactly to classical negation in the default theory, and elucidates the use of rules in logic programs. Like defaults, rules are unidirectional, so their contrapositives are not implicit; the rule connective, $\leftarrow$, is not material implication, but has rather the flavour of an inference rule, like defaults.

It is worth noting that existing top–down procedures for well-founded semantics without explicit negation [24, 14] can be easily adapted to the semantics we are proposing for extended programs. This issue is only briefly considered here and will be the subject of a forthcoming report.

## 1 Introduction

Our WFSX semantics follows from one basic "coherence" requirement: $\neg L$ implies $\sim L$ (if $L$ is explicitly false, $L$ must be false) for any literal $L$. Once that is built into the very definition of interpretation, the rest ensues by adapting to that requirement the formal techniques used for defining WFS (cf. [17]).

**Example 1** Consider $P = \{a \leftarrow \sim b;\ b \leftarrow \sim a;\ \neg a \leftarrow\}$.

If $\neg a$ were to be simply considered as a new atom symbol, say, $a'$, and WFS used to define the semantics of $P$ (as suggested in [18]), the result would be $\{\neg a, \sim\neg b\}$, so that $\neg a$ is true and $a$ is undefined. We insist that $\sim a$ should hold because $\neg a$ does. Accordingly, the WFSX of $P$ is $\{\neg a, b, \sim a, \sim\neg b\}$, since $b$ follows from $\sim a$.

**Example 2** The program $\{\neg a \leftarrow \sim b;\ a\}$ has no semantics since any model is contradictory.

**Example 3** We sustain the semantics of the program $\{q \leftarrow b,\ \ q \leftarrow \neg b\}$ is $\{\sim q, \sim b, \sim \neg q, \sim \neg b\}$, so that $q$ is false. We argue that $b \vee \neg b$ is not even part of the language of rules, let alone a rule, so $q$ should not be concluded; also, that $'\leftarrow'$ is not material implication, and that the corresponding default theory $(\{\frac{b\ :}{q}, \frac{\neg b\ :}{q}\}, \{\})$ does not conclude $q$ as well.

Our stance is that if the law of excluded middle is desired of some atom then it should be explicitly stated for it, not necessarily for all. We have yet to enlarge our language for rules to accomodate such expressiveness, and subsequently to perhaps extended default rules to allow $\frac{:}{b|\neg b}$ (as in [8]).

Another semantics extending well founded semantics for extended logic programs is presented in [5]. It differs from ours in that it does not comply with the coherence principle. This is easily seen with an example.

Consider program $P$ of example 1. $P \cup \{\}$ is a complete admissible scenario in their sense, and so the semantics of $P$ is $\{\neg a\}$, violating the principle.

Another important difference is in what concerns contradiction. Their semantics exercises some contradiction removal, giving thus meaning to more programs. To the program of example 2 it assigns the semantics is $\{a\}$, leaving $b$ undefined. We consider the issue of contradiction removal a separate one and we treat it, differently, elsewhere in the spirit of [9, 10].

## 2 Language Used

Given a first order language $Lang$ [17], an extended logic program is a set of rules of the form $H \leftarrow B_1, \ldots, B_n, \sim C_1, \ldots, \sim C_m$, where

$H, B_1, \ldots, B_n, C_1, \ldots, C_m$ are classical literals. A (syntactically) classical literal (or explicit literal) is either an atom $A$ or its explicit negation $\neg A$. We also use the symbol $\neg$ to denote complementary literals in the sense of explicit negation. Thus $\neg\neg A = A$. The symbol $\sim$ stands for negation by default[1]. $\sim L$ is called a default literal. Literals are either classical or default literals. A set of rules stands for all its ground instances wrt *Lang*.

## 3 A Semantics for Programs with Explicit Negation

In this section we define the Well Founded Semantics for programs with explicit negation, in line with the approach of the introduction. We present properties of this semantics, and illustrate with examples. For all proofs the reader is refered to the extended version of this paper [12]

We begin by providing a definition of interpretation for programs with explicit negation.

**Definition 3.1 (Interpretation)** *By an interpretation $I$ of a language Lang we mean any set $T \cup \sim F$* [2], *where $T$ and $F$ are disjoint subsets of classical literals over the Herbrand base, and if $\neg L \in T$ then $L \in F$ (coherence)*[3]. *The set $T$ contains all ground classical literals* true *in $I$, the set $F$ contains all ground classical literals* false *in $I$. The truth value of the remaining classical literals is* undefined *(The truth value of a default literal $\sim L$ is the 3-valued complement of $L$).*

**Proposition 3.1 (Noncontradiction condition)** *If $I = T \cup \sim F$ is an interpretation of a program $P$ then there is no pair of classical literals $A$, $\neg A$ of $P$ such that $A \in T$ and $\neg A \in T$.*

As before, an interpretation can be equivalently viewed as a function $I : Lit \rightarrow V$ where Lit is the set of all classical literals in the language and $V = \{0, \frac{1}{2}, 1\}$

Based on this function we can define a truth valuation of formulae.

**Definition 3.2 (Truth valuation)** *If $I$ is an interpretation, the truth valuation $\hat{I}$ corresponding to $I$ is a function $\hat{I} : C \rightarrow V$ where $C$ is the set of all formulae of the language, recursively defined as follows:*

- *if $L$ is a classical literal then $\hat{I}(L) = I(L)$.*
- $\hat{I}(\sim L) = 1 - I(L)$.
- $\hat{I}((S, V)) = min(\hat{I}(S), \hat{I}(V))$.
- $\hat{I}(L \leftarrow S) = 1$ *if* $\hat{I}(S) \leq \hat{I}(L)$ *or* $\hat{I}(\neg L) = 1$ *and* $\hat{I}(S) \neq 1$*; and* 0 *otherwise.*

The additional condition with respect to WFS, $\hat{I}(\neg L) = 1$ and $\hat{I}(S) \neq 1$, does not affect the valuation of formulae without $\neg$. Its purpose is to allow a conclusion $c$ to be false when the premises are undefined for some rule, on condition that $\neg c$ holds.

**Definition 3.3 (Model)** *An interpretation $I$ is called a model of a program $P$ iff for every ground instance of a program rule $H \leftarrow B$, $\hat{I}(H \leftarrow B) = 1$.*

As in [17] we expand our language by adding to it the proposition $\mathbf{u}$ such that every interpretation $I$ satisfies $I(\mathbf{u}) = \frac{1}{2}$. By a non-negative program we also mean a program whose premises are either classical literals or $\mathbf{u}$.

**Definition 3.4 (Least-operator)**
*We define $least(P)$, where $P$ is a non-negative program, as the set of literals $T \cup \sim F$ obtained as follows:*

- *Let $P'$ be the non-negative program obtained by replacing in $P$ every negative classical literal $\neg L$ by a new atomic symbol, say '$\neg\_L$'.*
- *Let $T' \cup \sim F'$ be the least 3-valued model of $P'$.*
- *$T \cup \sim F$ is obtained from $T' \cup \sim F'$ by reversing the replacements above.*

The generalization of Kowalski-Van Emden theorem made in [17] is also valid for extended logic of programs.

**Theorem 3.1** *$least(P)$ uniquely exists for every non-negative program $P$.*

We next extend with an additional rule the $P$ modulo $I$ transformation of [17], itself an extension of the Gelfond-Lifschitz modulo transformation.

**Definition 3.5 ($\frac{P}{I}$ transformation)** *Let $P$ be an extended logic program and let $I$ be an interpretation. By $\frac{P}{I}$ we mean a program obtained from $P$ by performing the following four operations:*

- *Remove from $P$ all rules containing a negative premise $L = \sim A$ such that $A \in I$.*

---

[1] This designation has been used in the literature instead of the more operational *"negation as failure (to prove)"*. Its precise meaning, and relation to default theories corresponding to logic programs, is the object of this paper; at this stage, however, *"default"* can be envisaged as just a conventional name. Another appropriate designation is *"implicit negation"*, in contradistinction to explicit negation.

[2] By $\sim\{a_1, \ldots, a_n\}$ we mean $\{\sim a_1, \ldots, \sim a_n\}$.

[3] For any literal $L$, if $L$ is explicitly false $L$ must be false. Note that the complementary condition "if $L \in T$ then $\neg L \in F$" is implicit.

- *Remove from $P$ all rules containing a premise $L$ such that $\neg L \in I$.*
- *Remove from all remaining rules of $P$ their negative premises $L = \sim A$ such that $\sim A \in I$.*
- *Replace all the remaining negative premises by proposition* $\mathbf{u}$.

Note that the new operation (the second one) is not applicable to nonextended programs. (The need for this operation arises from the coherence principle, and is illustrated below in this section.)

The resulting program $\frac{P}{I}$ is by definition non-negative. Thus by theorem 3.1 it always has a unique $least(\frac{P}{I})$.

$least(\frac{P}{I})$ isn't always an interpretation. Conditions about noncontradiction and coherence may be violated. To avoid incoherence, when contradiction is not present, we define the partial operator:

**Definition 3.6 (*Coh* operator)** *Let $I = T \cup \sim F$ be a set of literals such that $T$ does not contain any pair $A$, $\neg A$. We define $Coh(I) = I \cup \sim\{\neg L \mid L \in T\}$. $Coh$ is not defined for other sets of literals.*

The result of $Coh$ applied to $least(\frac{P}{I})$ is always an interpretation. The noncontradiction and coherence conditions are guaranteed by definition. $T$ and the false part $(F \cup \{\neg L \mid L \in T\})$ are disjoint because $T$ and $F$ are disjoint and none of the classical literals added to the false part are in $T$ since $T$ is noncontradictory.

Now we generalize the $\Gamma^*$ operator of [17].

**Definition 3.7 (The $\Phi$ operator)** *Let $P$ be a logic program and $I$ an interpretation, and let $J = least(\frac{P}{I})$. If $Coh(J)$ exists we define $\Phi_P(I) = Coh(J)$. Otherwise $\Phi_P(I)$ is not defined.*

**Definition 3.8 (WFS with explicit negation)** *An interpretation $I$ of an extended logic program $P$ is called an Extended Stable Model (XSM) of $P$ iff $\Phi_P(I) = I$. The F-least Extended Stable Model is called the Well Founded Model. The semantics of $P$ is determined by the set of all XSMs of $P$.*

It is easy to see that some programs may have no semantics (cf. example 2).

**Definition 3.9 (Contradictory program)** *An extended logic program $P$ is contradictory iff it has no semantics, i.e. there exists no interpretation $I$ such that $\Phi_P(I) = I$.*

Theorem 3.5 below expresses an alternative, more illustrative definition of contradictory program.

**Example 4** Consider again the program of example 1. Now $\{\neg a, \sim\neg b\}$ is no longer a XSM as in [18], because it is not an interpretation, and thus $\Phi$ does not apply to it. Its only XSM, and consequently its WFM, is $I = \{\neg a, b, \sim a, \sim\neg b\}$. $\frac{P}{I} = \{b \leftarrow, \quad \neg a \leftarrow\}$, its least model is $I$, $Coh(I) = I$, and thus $\Phi_P(I) = I$.

**Remark 3.1** *According to [18] the above program has the two XSMs, $\{\neg a, \sim\neg b\}$ and $\{\neg a, b, \sim a, \sim\neg b\}$. It is not enough to throw out those not complying with coherence. Although that's true for this example, example 6 shows that is not in general the case.*

We now come back to the question of the need for the extra operation introduced in the modulo operator.

**Example 5** Consider $P = \{c \leftarrow a;\ a \leftarrow b;\ b \leftarrow \sim b;\ \neg a\}$. Its only XSM is $I = \{\neg a, \sim a, \sim c, \sim\neg b, \sim\neg c\}$. In fact: $\frac{P}{I} = \{\neg a,\ a \leftarrow b,\ b \leftarrow \mathbf{u}\}$, $least(\frac{P}{I}) = \{\neg a, \sim c, \sim\neg b, \sim\neg c\}$ and consequently $\Phi(I) = I$. Note that if the new operation for the modulo transformation were absent $\frac{P}{I}$ would contain the rule $c \leftarrow a$, and $c$ would be undefined rather than false. This would be against the coherence principle, since $\neg a$ implies $\sim a$, and as the only rule for $c$ has $a$ in the body, it should also imply $\sim c$. The rôle of the new operation is to ensure the propagation of false as a result of any $\sim L$ implied by a $\neg L$ through coherence.

**Example 6** Consider $P = \{c \leftarrow \sim b;\ b \leftarrow \sim a;\ a \leftarrow \sim a;\ \neg b\}$. Its only XSM is $M = \{\neg b, c, \sim b, \sim\neg c, \sim\neg a\}$. Indeed: $\frac{P}{M} = \{c \leftarrow,\ b \leftarrow \mathbf{u},\ a \leftarrow \mathbf{u},\ \neg b \leftarrow\}$, its least model is $\{c, \neg b, \sim\neg c, \sim\neg a\}$, and consequently $\Phi_P(M) = M^4$. By simply considering $\neg b$ as a new atom this program would have a single XSM, $\{\neg b\}$, which is not a coherent interpretation.

It is also interesting to see in this example that $M$ is not a model in the usual sense because for the second rule of $P$ the value of the head $(M(b) = 0)$ is smaller than the value of the body $(M(\sim a) = \frac{1}{2})$.

The intuitive idea is that the truth of $\neg b$ overrides any rule for $b$ with undefined body, so that $\sim b$ becomes true (and $b$ false), rather than undefined.

**Theorem 3.2 (XSMs are models)** *Every XSM $I$ of a program $P$ is a model of $P$ (cf. definition 3.3).*

In the above definition of the semantics (definition 3.8) we define the WFM as the F-least XSM. This is possible because:

[4] Note how the truth of $\neg b$ compels the truth of $\sim b$ via the $Coh$ operator.

**Theorem 3.3 (Existence of the semantics)** *For noncontradictory programs there always exists a unique F-least XSM.*

**Theorem 3.4 (Monotonicity of $\Phi$)** *Let $P$ be a noncontradictory program. Then the operator $\Phi_P$ is monotonic wrt the F-ordering, i.e. $A \subseteq B \Rightarrow \Phi_P(A) \subseteq \Phi_P(B)$ for any interpretations $A$ and $B$.*

**Definition 3.10** *In order to obtain a constructive bottom–up*[5] *definition of the WFM of a given noncontradictory program $P$, we define the following transfinite sequence $\{I_\alpha\}$ of interpretations of $P$:*

$$\begin{array}{rcl} I_0 & = & \{\} \\ I_{\alpha+1} & = & \Phi_P(I_\alpha) \\ I_\delta & = & \bigcup \{I_\alpha \mid \alpha < \delta\} \quad \text{for a limit ordinal } \delta \end{array}$$

*By theorem 3.4, and according to the properties of monotonic operators, there must exist a smallest $\lambda$ such that $I_\lambda$ is a fixpoint of $\Phi_P$, and $WFM = I_\lambda$.*

This constructive definition requires one to know *a priori* if the given program is contradictory. This requirement is not needed if we consider the following theorem.

**Theorem 3.5** *A program $P$ is contradictory iff for the sequence of the $I_\alpha$ there exists a $\lambda$ such that $\Phi_P(I_\lambda)$ is not defined, i.e. $least(\frac{P}{I_\lambda})$ has a pair of classical literals $A$, $\neg A$.*

Thus, in order to compute the WFM of a program $P$ one starts by building the above sequence. If at some step $\Phi_P$ is not applicable then we end our iteration and say that $P$ is contradictory. Otherwise we iterate until the least fixpoint of $\Phi_P$, which is the WFM of $P$.

It is worth noting that this semantics is a generalization of the stationary semantics (or extended stable model semantics) [18, 19, 20], to programs with explicit negation.

**Theorem 3.6** *For programs without explicit negation our semantics coincide with stationary semantics.*

[5] Top–down procedures computing this semantics (for noncontradictory programs) can be easily obtained by adapting existing procedures for programs without explicit negation, such as [14], as follows: replace every literal of the form $\neg A$ by a new literal, say $A'$; include two new rules "$\sim A$ *rewrites to* $A'$ and "$\sim A'$ *rewrites to* $A$. If $A$ and $A'$ are both derivable then the program is contradictory.

# 4 Relation between the Semantics of Default Theories and Logic Programs with Explicit Negation

A relationship between logic programs and default theories were first proposed in [3] and [4]. The idea is to translate every program rule into a default rule and then compare extensions of the default theory with the semantics of the corresponding program.

In [4], stable model semantics [6] was shown equivalent to a special case of default theories in the sense of Reiter [21]. This result was generalized in [7] to programs with explicit negation and answer-set semantics, where they claim that explicit negation is in fact the classical negation used in default theories.

[1] extends Reiter's semantics of default theories, resolving some issues of the latter, namely that some theories have no extension and that some theories have no least extension. This is achieved by iterating twice Reiter's $\Gamma$ operator. Recently, in [2], the well founded semantics for programs without explicit negation was shown equivalent to a special case of the extension classes of default theories in the sense of [1]. It turns out that in attempting to extend this last result to extended logic programs we get some unintuitive results.

These problems are addressed in [11]. There a new semantics for defaults is defined. This semantics is similar to the one in [1], but uses semi-normal defaults in the inner step. A brief description of this semantics follows.

**Definition 4.1 (The $\Omega$ operator)** *Let $\Delta$ be a default theory and $E$ a context. We define $\Omega(E)$ as $\Gamma'_\Delta(\Gamma'_{\Delta^s}(E))$, where $\Delta^S$ is the default theory obtained from $\Delta$ by transforming every default rule into semi-normal, and $\Gamma'$ is obtained from $\Gamma$ by not making any deductive closure.*

**Definition 4.2 ($\Omega$-extension)** *A context $E$ is a $\Omega$-extension of a default theory $\Delta$ iff $E = \Omega(E)$, $E \subseteq \Gamma'_{\Delta^s}(E)$, and $E$ is consistent.*

Now we present the equivalence of $\Omega$–extensions and XSMs of extended logic programs.

**Definition 4.3** *Let $\Delta = (D, \{\})$ be a default theory. We say that an extended logic program $P$ corresponds to $\Delta$ iff for every default of the for $\frac{\{a_1,\ldots,a_n\} \,:\, \{b_1,\ldots,b_m\}}{C} \in \Delta$ there exists a rule $c \leftarrow a_1, \ldots, a_n, \sim\neg b_1, \ldots, \sim\neg b_m \in P$, where $b_j$ denotes the complement of $\neg b_j$.*

**Definition 4.4** *An interpretation $I$ of a program $P$ corresponds to a context $E$ of the corresponding default theory $T$ iff for every classical literal $L$ of $P$ (and literal $L$ of $T$):*

- $I(L) = 1$ *iff* $L \in E$ *and* $L \in \Gamma'_{\Delta^s}(E)$.
- $I(L) = \frac{1}{2}$ *iff* $L \notin E$ *and* $L \in \Gamma'_{\Delta^s}(E)$.
- $I(L) = 0$ *iff* $L \notin E$ *and* $L \notin \Gamma'_{\Delta^s}(E)$.

The main theorem relating both semantics is now presented as follows:

**Theorem 4.1 (Correspondence)** *Let $\Delta = (D, \{\})$ be a default theory corresponding to the program $P$. $E$ is a $\Omega$-extension of $\Delta$ iff the interpretation $I$ corresponding to $E$ is a XSM of $P$.*

According to this theorem we can say that explicit negation is nothing but classical negation in default theories. As $\Omega$ default semantics is a generalization of $\Gamma$ default semantics [12], and since answer-sets semantics correspond to $\Gamma$ default semantics [7], it turns out that answer-sets semantics (and hence the semantics defined in [23]) are special cases of our semantics.

## Acknowledgements

We thank COMPULOG, INIC, JNICT and Gabinete de Filosofia do Conhecimento for their support. Thanks to Gabriel David, Michael Gelfond, and Luís Monteiro for their comments.

## References

[1] C. Baral and V. S. Subrahmanian. Stable and extension class theory for logic programs and default logics. In *International Workshop on Nonmonotonic Reasoning*, 1990.

[2] C. Baral and V. S. Subrahmanian. Dualities between alternative semantics for logic programming and nonmonotonic reasoning. In A. Nerode, W. Marek, and V. S. Subrahmanian, editors, *LP-NMR'91*. MIT Press, 1991.

[3] N. Bidoit and C. Froidevaux. Minimalism subsumes default logic and circumscription in stratified logic programming. In *Symposium on Principles of Database Systems*. ACM SIGACT-SIGMOD, 1987.

[4] N. Bidoit and C. Froidevaux. General logic databases and programs: default logic semantics and stratification. *Journal of Information and Computation*, 1988.

[5] P. M. Dung and P. Ruamviboonsuk. Well founded reasoning with classical negation. In A. Nerode, W. Marek, and V. S. Subrahmanian, editors, *LP-NMR'91*. MIT Press, 1991.

[6] M. Gelfond and V. Lifschitz. The stable model semantics for logic programming. In R. A. Kowalski and K. A. Bowen, editors, *5th ICLP*, pages 1070–1080. MIT Press, 1988.

[7] M. Gelfond and V. Lifschitz. Logic programs with classical negation. In *7th ICLP*, pages 579–597. MIT Press, 1990.

[8] M. Gelfond, H. Przymusinska, V. Lifschitz, and M. Truczsinski. Disjunctive defaults. In *KR'91*. Morgan Kaufmann, 1991.

[9] L. M. Pereira, J. J. Alferes, and J. N. Aparício. Contradiction Removal within Well Founded Semantics. In A. Nerode, W. Marek, and V. S. Subrahmanian, editors, *LPNMR'91*. MIT Press, 1991.

[10] L. M. Pereira, J. J. Alferes, and J. N. Aparício. The extended stable models of contradiction removal semantics. In P. Barahona, L. M. Pereira, and A. Porto, editors, *5th Portuguese AI Conf.'91*. Springer-Verlag, 1991.

[11] L. M. Pereira, J. J. Alferes, and J. N. Aparício. Default theory for logic programs with explicit negation. Technical report, AI Centre, Uninova, March 1992.

[12] L. M. Pereira, J. J. Alferes, and J. N. Aparício. Well founded semantics with explicit negation and default theory. Technical report, AI Centre, Uninova, March 1992.

[13] L. M. Pereira, J. N. Aparício, and J. J. Alferes. Counterfactual reasoning based on revising assumptions. In Ueda and Saraswat, editors, *ILPS'91*. MIT Press, 1991.

[14] L. M. Pereira, J. N. Aparício, and J. J. Alferes. A derivation procedure for extended stable models. In *IJCAI'91*. Morgan Kaufmann Publishers, 1991.

[15] L. M. Pereira, J. N. Aparício, and J. J. Alferes. Hypothetical reasoning with well founded semantics. In B. Mayoh, editor, *SCAI'91*. IOS Press, 1991.

[16] L. M. Pereira, J. N. Aparício, and J. J. Alferes. Nonmonotonic reasoning with well founded semantics. In *8th ICLP*. MIT Press, 1991.

[17] H. Przymusinska and T. Przymusinski. *Semantic Issues in Deductive Databases and Logic Programs.* Formal Techniques in Artificial Intelligence. North Holland, 1990.

[18] T. Przymusinski. Extended stable semantics for normal and disjunctive programs. In *7th ICLP*, pages 459–477. MIT Press, 1990.

[19] T. Przymusinski. Stationary semantics for disjunctive logic programs and deductive databases. In *NACLP'90*, pages 40–57. MIT Press, 1990.

[20] T. Przymusinski. A semantics for disjunctive logic programs. In *ILPS'91 Workshop in Disjunctive Logic Programs*, 1991.

[21] R. Reiter. A logic for default reasoning. *Artificial Intelligence*, 13:68–93, 1980.

[22] A. Van Gelder, K. A. Ross, and J. S. Schlipf. The well-founded semantics for general logic programs. *Journal of ACM*, pages 221–230, 1990.

[23] G. Wagner. A database needs two kinds of negation. In B. Thalheim, J. Demetrovics, and H-D. Gerhardt, editors, *MFDBS'91*, pages 357–371. Springer-Verlag, 1991.

[24] D.S. Warren. The XWAM: A machine that integrates prolog and deductive databases. Technical report, SUNY at Stony Brook, 1989.