LOGIC PROGRAMMING WITH GENERAL CLAUSES AND DEFAULTS BASED ON MODEL ELIMINATION

Marco A. Casanova, Ramiro Guerreiro, Andrea Silva

Rio Scientific Center - IBM Brazil
P.O. Box 4624
20.001, Rio de Janeiro, RJ - Brazil

Abstract

The foundations of a class of logic programming systems with the expressive power of full first-order logic and a non-monotonic component is addressed. The underlying refutation method is an extended version of weak model elimination. The first question addressed is how to compute answers with weak model elimination when queries and programs are sets of arbitrary clauses, which is completely settled by a soundness and completeness result. The question of computing only definite answers is also settled. Then, the problem of computing answers is rediscussed when the logic programs also include a finite set of defaults.

1. INTRODUCTION

This paper addresses the foundations of a class of logic programming systems with the expressive power of full first-order logic and a non-monotonic component. Systems in this class provide a direct generalization of pure Prolog and they can be implemented using the same technology as Prolog processors. The discussion centers on the underlying refutation method, which is an extended version of weak model elimination (loveland [1978]) enhanced with defaults (Reiter [1980]).

The first contribution of this paper consists in defining an adaptation of weak model elimination (*WME)* that is sound and complete with respect to computing answers when the logic programs and queries are expressed by sets of generic clauses. The proofs of these two results are far more complex than the corresponding results for SLD-resolution (see Lloyd [1984]), the basis of Prolog systems. The question of computing just definite answers is also settled, using a new result about refutations in *WME.*

Negation within the scope of *WME* has the classical meaning. The second contribution of this paper then refers to extending *WME* with defaults to capture non-monotonic reasoning. Within this broader scope, the notion of an answer to a query posed to a logic program raises interesting questions that are briefly discussed. Defaults provide a much more flexible mechanism than negation by finite failure (Clark [1978]), the treatment of negation commonly adopted to extend the expressive power of logic programs and queries in Prolog.

A detailed account of the results reported here can be found in Casanova et alii [1988] and Guerreiro et alii [1989]. A logic programming systems based on model elimination is also described in Silva et alii [1989].

The organization of this paper is as follows. Section 2 introduces the notions of program, query and answer. Section 3 reviews the weak model elimination method and extends it to compute answers, describing a variation for definite answers. Section 4 introduces an adaptation of the method that deals with a special class of defaults. Finally, section 5 contains the conclusions.

2. PROGRAMS, QUERIES and ANSWERS

A *program* P is a finite set of clauses and a *query* Q is a disjunction of conjunctions of literals, that is, a quantifier-free formula in disjunctive normal form. A query is *definite* iff it is a single conjunction of literals, otherwise it is *indefinite.*

An *answer* A to a query Q over a program P is either False or a dissjunction of instances of conjunctions in Q over the alphabet of P and Q, that is, a disjunction of conjunctions obtained from those in Q by substituting variables by terms of the alphabet used to write P and Q. An answer is *definite* iff it consists of a single conjunction, otherwise it is *indefinite* (Reiter [1978]).

An answer A to Q over P is *correct* iff P logically implies VA, the universal closure of A. Finally, an answer A to Q over P is more general than an answer B to Q over P iff A logically implies B. We let False be an answer simply because it will be the most general answer to any query over an inconsistent program.

For example, the following set of clauses is a program, that we call DIC:

1. program(a,fortran)
2. program(b,pascal)
3. program(c,fortran) program(c,pascal)
4. calls(a,b)
5. calls(b,c)
6. ¬calls (xy) depends(x,y)
7. ¬calls(xz) ¬depends(zj[j]) depends(x,y)

Thus, clause (3) indicates that c is an ordinary' program written in fortran or pascal and clauses (6) and (7) indicate that x depends on y if x calls y direct or transitively. The formula below is a query, that we call DEP[a]:

(depends(a,x) A program(x,pascal)) v
(depends(a,x) A program(x,fortran))

It asks for a program written in fortran or pascal that the program a depends on. An answer A to DEP[a] over DIC would be:

depends(a.b) A program(b,pascal)

Indeed, the conjunction in A is an instance of the first conjunction in DEP[a]. It is in fact a correct answer since DIC logically implies VA. A second correct answer to DEP[a] over DIC would be:

(depends(a,c) A program(cfortran)) v
(depends(a,c) A program(c,pascal))

3. COMPUTING ANSWERS WITH WME

3.1 Weak Model Elimination

To achieve completeness, the inference rules of WME sometimes maintain the resolved literals within the derived clauses and keep the literals (resolved or not) ordered within a clause. To distinguish these extended clauses from the ordinary clauses, they will be called chains. Moreover, resolved literals in a chain will be enclosed within brackets.

More precisely, a resolved literal (or a R-literal) is an expression of the form [L], where L is a literal. An element is a literal or a R-literal. An elementary chain is any sequence of literals and a chain is any sequence of elements. The symbol □ will denote the empty chain, which is elementary by definition. Each chain C represents, by convention, the universal closure of the disjunction of its literals, in the sense that any structure for the first-order alphabet in question satisfies C if and only if it satisfies the formula that C represents. Hence, the R-literals of a chain do not influence its semantics. From these definitions, it should be clear that elementary chains and clauses are one and the same concept and will be used interchangeably in this text.

The next definitions are basic for the inference rules of the method and assume familiarity with the notion of unification. In what follows, B′B″ denotes the concatenation of two chains B′ and B″ and |L| indicates the atom of a literal L. Two literals L′ and L″ can be *cancelled by a substitution* θ if and only if θ is a most general unifier (m.g.u.) of {|L′|,|L″|} and L′θ and L″θ are complementary.

Let A′ be a chain and let A″ be an elementary chain. Let β be a renaming of the variables of A″ such that A″β has variables distinct from those of A′. Let L′ be the leftmost element of A′ and suppose that L′ is a literal. A chain A is an *extension* of A′ *by* A″ if and only if there exists a literal L″ of A″ and a substitution θ such that L′ and L″β can be cancelled by θ and A = B″B′, where B″ is the chain A″βθ with the literal L″βθ removed and B′ is a chain A′θ with the literal L′θ replaced by [L′θ].

Let A′ be a chain. Let L′ be the leftmost element of A′ and suppose that L′ is a literal. A chain A is a *reduction* of A′ if and only if there exists a R-literal [M′] of A′ and a substitution θ such that L′ and M′ can be cancelled by θ and A is A′θ with the literal L′θ removed.

A chain A is a *contraction* of a chain A′ if and only if A is the chain obtained by removing from A′ all R-literals that are to the left of the leftmost literal. If A has only R-literals, the result becomes the empty chain.

A chain A is a *full extension* of A′ by A″ if and only if A is the contraction of an extension of A′ by A″. A chain A is a *full reduction* of a chain A′ if and only if A is the contraction of a reduction of A′.

The *weak model elimination method*, ***WME***, works with the class of all sets of first-order chains, it has no axioms and two inference rules, *full extension* and *full contraction*, defined as follows:

Full Extension:

if A′ and A″ are chains and
A is a full extension of A′ by A″
then derive A from A′ and A″.

Full Reduction:

if A′ is chain and
A is a full reduction of A′
then derive A from A′.

A *WME-deduction* of a chain C from a set **S** of elementary chains is any finite sequence of chains $E = (E_1,...,E_n)$ such that C is the last chain of E, there is $i \leq n$ such that $E_1,...,E_i$, the *prefix* of E, consists of chains in **S** and, for each $j \in [i+1,n]$, E_j is derived from E_{j-1}, the *parent chain* of E_j, by full reduction or full extension, in the latter case using an *auxiliary chain* from the prefix of E. The chain E_i is called the *initial chain* of E. A *WME-refutation* from a set of elementary chains **S** is a WME-deduction of the empty chain from **S**.

The ***WME*** method defined above is slightly different from the original version of Loveland [1978] but the results therein can be easily adapted to establish that ***WME*** is refutationally sound and complete.

3.2 Computing Answers with WME

Given a WME-refutation R from the elementary chains in a program **P** and in the clausal representation of the negation of the existential closure of a query **Q**, denoted by $CL(\neg\exists\mathbf{Q})$, it is possible to show that the substitutions applied to the free variables of chains in $CL(\neg\exists\mathbf{Q})$ during the construction of R induce a correct answer to **Q** over **P**. However, to recover such substitutions is not exactly simple since $CL(\neg\exists\mathbf{Q})$ may possibly contain more than one chain, that may also be reused in R. This section then redefines the notion of chain and the inference rules of ***WME*** to register such substitutions, using answer literals (Green [1969]).

An *activated chain* is a pair of the form (C,**L**), where C is a chain and **L** is a set of literals. The *activation* of **P** is the set *activate*(**P**) consisting of the activated chains $(C,\emptyset)$, where $C \in \mathbf{P}$. The *activation* of a query **Q** of the form $Q_1 \vee ... \vee Q_n$ is the set *activate*(**Q**) of activated chains $(\sim Q_i, \{r_i(\bar{x}_i)\})$, $i = 1,...,n$, where $\sim Q_i$, by convention, is the chain consisting of the complement of the literals of Q_i, $\bar{x}_i$ is a list of the variables of Q_i and r_i is a predicate symbol, not in the original alphabet, whose arity is equal to the length of $\bar{x}_i$. The literal $r_i(\bar{x}_i)$ is the *answer literal* for Q_i in the activation of **Q**.

The activation of a query **Q** therefore produces a clausal representation of the negation of the existential closure of **Q**, with each elementary chain annotated with an answer literal whose function will be to record the substitutions applied to the variables of the chain. But, to effect this recording, the inference rules of ***WME*** had to be modified as follows.

An activated chain (A,**L**) is a *full activated reduction* of an activated chain (A′,**L**′) if and only if A is a full reduction of A′ with m.g.u. θ and $\mathbf{L} = \mathbf{L}'\theta$. An activated chain (A,**L**) is a *full activated extension* of (A′,**L**′) *by* an elementary activated chain (A″,**L**″) if and only if A is a full extension of A′ by A″, with m.g.u. θ and renaming β of A″, and $\mathbf{L} = \mathbf{L}'\theta \cup \mathbf{L}''\beta\theta$.

An answer **A** to **Q** over **P** is *WME-computed* if and only if there is an activated WME-refutation R from *activate*(**P**) ∪ *activate*(**Q**) such that either R terminates in $(\Box,\emptyset)$, in which case **A** must be equal to **False**, or R terminates in $(\Box,\mathbf{L})$, with $\mathbf{L} \neq \emptyset$, and **A** is a disjunction of all conjuncts B such that there is $(\sim Q_i, \{r_i(\bar{x}_i)\}) \in activate(\mathbf{Q})$ and $r_i(\bar{t}) \in \mathbf{L}$ such that B is equal to $Q_i\theta$, where $\theta = \{\bar{x}_i / \bar{t}\}$.

The following example illustrates how the method computes a definite answer to an indefinite query. Consider again the program **DIC** and the query **DEP[a]** introduced in section 2. An activated WME-refutation from the set of chains in the activation of **DIC** (chains 1 to 7) and the activation of **DEP[a]** (chains 8 and 9) is:

1. (program(a,fortran), $\emptyset$)
2. (program(b,pascal), $\emptyset$)
3. (program(c,fortran) program(c,pascal), $\emptyset$)
4. (calls(a,b), $\emptyset$)
5. (calls(b,c), $\emptyset$)
6. ($\neg$calls(x,y) depends(x,y), $\emptyset$)
7. ($\neg$calls(x,z) $\neg$depends(z,y) depends(x,y), $\emptyset$)
8. ($\neg$depends(a,u) $\neg$program(u,fortran), {$r_1(u)$})
9. ($\neg$depends(a,v) $\neg$program(v,pascal), {$r_2(v)$})
10. ($\neg$calls(a,y) [$\neg$depends(a,y)] $\neg$program(y,pascal), {$r_2(y)$}) (from 9 and 6)
11. ($\neg$program(b,pascal), {r_2(b)}) (from 10 and 4)
12. ($\Box$, {r_2(b)}) (from 11 and 2)

Hence, **A** = depends(a,b) ∧ program(b,pascal) is a WME-computed answer to **DEP[a]** over **DIC** since

r_2(b) in (12) indicates that the variable v of the chain in (9) was substituted by b.

The *WME* method, modified as described above, is sound and complete for computing answers in the following sense:

Theorem 1: (Soundness and Completeness Theorem)

Let **P** be a program and **Q** be a query.

(a) Every WME-computed answer to **Q** over **P** is correct.

(b) Given any correct answer to **Q** over **P**, there is a WME-computed answer which is more general.

We conclude this section with another variation of weak model elimination that computes only definite answers.

Let **S** be a set of activated elementary chains and **T** be a subset of **S**. We say that an activated WME-refutation R from **S** has *initial support* from **T** iff the initial activated chain of R is in **T** and no activated chain in **T** is ever used as an auxiliary chain in derivations in R.

Let **Q** be a query to a program **P**. An answer **A** to **Q** over **P** is *WME-computed with initial support from* **Q** iff there is an activated WME-refutation R from *activate*(**P**) ∪ *activate*(**Q**), with initial support from *activate*(**Q**), that computes **A**. Note that, since just one chain from *activate*(**Q**) is used in R, **A** is a definite answer. In fact, we can prove that:

Theorem 2: (Soundness and Completeness Theorem for Definite Answers)

Let **P** be a program and **Q** be a query.

(a) Let **A** be an answer to **Q** over **P** that is WME-computed with initial support from **Q**. Then, **A** is definite and correct.

(b) Given any definite correct answer **A** to **Q** over **P**, there is a definite answer **B** to **Q** over **P** such that **B** is WME-computed with initial support from **Q** and **B** is more general than **A**.

4. EXTENDING WME WITH DEFAULTS

4.1 Clausal Defaults

A *clausal default* is any expression of the form "A:C", where the *prerequisite* A of the default is a conjunction of literals and the *consequent* C of the default is an elementary chain. A *clausal default theory* is a pair $\Delta = (\mathbf{P},\mathbf{D})$, where **P** is a finite set of elementary chains and **D** is a finite set of clausal defaults. We will denote by *consequent*(**D**) the set of the consequents of all defaults in **D**. We also accept ":C" as a clausal default.

The clausal default "A:C" should be understood as a convenient way of expressing the open normal default $\frac{A:MF}{F}$, in the notation of Reiter [1980], where F is a disjunction of the literals in the clause C. Therefore, since we will limit ourselves to clausal defaults, we will be concerned with a particular case of open normal defaults.

The semantics for clausal default theories follows from the concept of extensions for open default theories. Then, given a clausal default theory $\Delta = (\mathbf{P},\mathbf{D})$, a clausal default "A:C" in **D** should be interpreted as a generator of the set of defaults "$A\theta$:$C\theta$", for all substitutions θ of the variables in A and C by terms of the Herbrand universe for the current alphabet. The clausal default "$A\theta$:$C\theta$" therefore reads: $C\theta$ can be assumed "by default" if the prerequisite $A\theta$ is believed and $C\theta$ is consistent with the beliefs.

Let $\Delta = (\mathbf{P},\mathbf{D})$ be a clausal default theory and let **Q** be a disjunction of conjunctions of literals. Let $CL(\neg\exists\mathbf{Q})$ again denote a clausal representation of the negation of the existential closure of **Q**. Intuitively, a refutation with defaults from Δ and **Q** is a sequence of WME-refutations such that the first is a WME-refutation from the chains in **P** ∪ $CL(\neg\exists\mathbf{Q})$ ∪ *consequent*(**D**) with initial chain in $CL(\neg\exists\mathbf{Q})$ and, after the first, each WME-refutation intends to refute the negation of the conjunction of the prerequisites of the defaults used on the preceding WME-refutation, with the appropriate substitutions, from the chains in **P** ∪ *consequent*(**D**) (but not in $CL(\neg\exists\mathbf{Q})$). The sequence should also satisfy a global consistency test, verifying if the use of the defaults is acceptable.

To record the substitutions affecting each use of each default in each WME-refutation in the sequence, we will use default literals in the same way we used answer literals for computing answers. Hence, the *indexing* of $\Delta = (\mathbf{P},\mathbf{D})$ is the set of pairs of the form $(C_i,\emptyset)$, where $C_i \in \mathbf{P}$, or the form $(C_i,\{\delta_i(\bar{x}_i)\})$, for each default "$A_i$:$C_i$" in **D**, where $\bar{x}_i$ is a list of the variables occurring in A_i and C_i and δ_i is a new predicative symbol whose arity is the length of $\bar{x}_i$. The *indexing* of **Q** consists of the set of pairs $(C_i,\emptyset)$, where $C_i \in CL(\neg\exists\mathbf{Q})$. Similarly to the answer literals, $\delta_i(\bar{x}_i)$ will record the substitutions applied to the variables of the default "A_i:C_i". For that purpose, the inference rules of ***WME*** and the notions of WME-deduction and WME-refutation are also modified to account for default literals similarly to what was described in section 3 for answer literals.

Let R be an indexed WME-refutation from the chains in the indexing of Δ and **Q**. Suppose that R terminates in ($\Box$,**S**). A default ψ is *returned* by R iff there is a pair $(C_i,\{\delta_i(\bar{x}_i)\})$ in the indexing of Δ, corresponding to some default "A_i:C_i" in **D**, and there exists a literal of the form $\delta_i(\bar{t})$ in **S** such that ψ can be written as "$A_i\theta$:$C_i\theta$", where $\theta = \{\bar{x}_i/\bar{t}\}$.

Let "A_i:C_i" be a default in **D** and $(C_i,\{\delta_i(\bar{x}_i)\})$ be the corresponding pair in the indexing of Δ. This default is *fired* in R iff exists an indexed chain in R derived by indexed full extension with $(C_i,\{\delta_i(\bar{x}_i)\})$ as auxiliary chain. Then, each default "A_i:C_i" fired in R, as well as each default corresponding to a default literal in the initial chain of R, if any, generates a *descendent* default in the set of defaults returned by R.

A *WME-refutation sequence with defaults* from $\Delta = ($**P**,**D**$)$ and **Q** is a finite sequence $\boldsymbol{R} = (R_0,\ldots,R_k)$ of indexed WME-refutations such that:

1) R_0 is an indexed WME-refutation from the indexing of Δ and the indexing of **Q**, with initial chain in indexing of **Q**;
2) for $0 \le i \le k$, let $\mathbf{D}_i$ be the set of defaults returned by R_i, $\mathbf{M}_i$ be the set of default literals corresponding to the defaults in $\mathbf{D}_i$, $\mathbf{D}^{(i)}$ be the set of defaults in $\mathbf{D}_i$ which are the descendents of the defaults fired in R_i and B_i be the chain representing the negation of the conjunction of the prerequisites of all defaults in $\mathbf{D}^{(i)}$. Then:
   a) For $1 \le i \le k$, R_i must be an indexed WME-refutation, with initial support $\{(B_{i-1},\mathbf{M}_{i-1})\}$, from the indexed chains in the indexing of $\Delta \cup \{(B_{i-1},\mathbf{M}_{i-1})\}$;
   b) $\mathbf{D}^{(k)} = \emptyset$;
   c) *(Consistency Test)*. Let **C** be the set of the consequents of all defaults occurring in $\mathbf{D}_k$. Then, there is a substitution θ of the variables occurring in **C** by ground terms of Herbrand universe over the current alphabet such that $\mathbf{P} \cup \mathbf{C}\theta$ is satisfiable.

It follows from the results in Reiter [1980] and from the soundeness and completeness theorem for computing definite answers that the WME method, adapted to account for clausal defaults as described above, is refutationally correct and complete.

4.2. Computing Answers with Defaults

A *program with defaults* is a pair $\Delta = ($**P**,**D**$)$, where **P** is a finite set of elementary chains and **D** is a finite set of clausal defaults. The notions of query and answer are not modified. The notion of correct answer is now based on the notion of extensions of default theories.

The computation of answers now combines the two notions described in sections 3.2 and 4.1: activation and indexing. Then, the inference rules now work with triples of the form (C,**L**,**M**), where C is a chain, **L** is a set of answer literals for computing answers of a query, following the description in the section 3.2, and **M** is a set of default literals for monitoring the defaults used in a refutation, as described in section 4.2.

The definition of the WME-refutation with defaults immediately induces the following notion of computed answer. Let $\Delta = ($**P**,**D**$)$ be a program with defaults and **Q** be a query to Δ. An answer **A** to **Q** over Δ is *WME-computed by defaults* iff there exists a WME-refutation with defaults R from the activation and indexing of Δ and **Q** such that the last refutation in R terminates with ($\Box$,**S**,**E**), the consistency test for R uses the substitution θ and either $\mathbf{S} = \emptyset$, in which case **A** must be equal to **False**, or $\mathbf{S} \neq \emptyset$ and **A** is a disjunction of all conjuncts B such that there is $(\sim Q_i,\{r_i(\bar{x}_i)\},\emptyset)$ in the activation and indexing of **Q** and there is $r_i(\bar{t}) \in \mathbf{S}$ such that B is equal to $Q_i\gamma\theta$, where $\gamma = \{\bar{x}_i/\bar{t}\}$.

This definition is not reasonable, though, because it admits answers with arbitrary instanciations, coming from the substitution θ generated for the consistency test. On the other hand, by the semantics itself of open defaults, it is not possible to abandon such substitution under the risk of invalidating the correctness of WME-refutation sequences with defaults. For example, let $\Delta = ($**P**,**D**$)$ be a program, where **P** = {bird(z), $\neg$fly(penguin), $\neg$fly(ostrich), yellow(canary)} and **D** = {bird(y):fly(y)}. Consider the query **Q** = fly(x). Consider the WME-refutation sequence with defaults $R = (R_0,R_1)$, from Δ and **Q**, constructed as follows. First, R_0 is an indexed and activated WME-refutation from the activation and indexing of Δ and **Q**:

0.1 (fly(y),∅,{δ(y)}) (from Δ)
0.2 (¬fly(x),{r(x)},∅) (from **Q**)
0.3 (□,{r(x)},{δ(x)}) (from 0.2 by 0.1)

Note that R_0 returns the default "bird(x):fly(x)". Hence, R_1 must be an indexed and activated WME-refutation from the chains in the indexing and activation of Δ and the chain representing the negation of the prerequisite of "bird(x) : fly(x)" (the chain (1.2) below):

1.1 (bird(z),∅,∅) (from Δ)
1.2 (¬bird(x),{r(x)},{δ(x)})
(from the negation of the prerequisite)
1.3 (□,{r(x)},{δ(x)}) (from 1.2 by 1.1)

Note that the chain in 1.2 carries on the set of answer literals and the set of default literals from the chain 0.3. This is necessary to correctly compute answers.

By the definition of WME-refutation sequence with defaults, we must also test the consistency of the set **E** = {bird(z), ¬fly(penguin), ¬fly(ostrich), yellow(canary)} U {fly(x)θ}, for some substitution θ of x by a term of the Herbrand universe of the alphabet in question. Indeed, by taking θ = {x/canary}, the set **E** becomes consistent. Hence, fly(canary) is the answer WME-computed by the refutation R, for this choice of θ.

Note that the choice of θ is entirely arbitrary. On the other hand, it is not possible to ignore θ, since the set {bird(z), ¬fly(penguin), ¬fly(ostrich), yellow(canary)} U {fly(x)}, is not satisfiable. Intuitively, the default in **D** cannot be fired for a substitution of x by, for example, penguin.

To solve the above dilemma, we propose to always base the consistency test on a class of substitutions that change each variable by a new constant not in the original language, whose semantics would be "the typical individual such that ...". In the current example, we introduce the new constant P_0, understood as "the typical bird". Consider again the WME-refutation with defaults *R,* except that the substitution of the consistency test is now, by definition, 0 = {X/P_0}. Since, for this choice of 0, the set (bird(z), ¬flyfpenguin), ¬flyfostrich), yellow(canary)} U {fly(x)0}, is consistent, we have that fly(Po) is the new computed answer by the WME-refutation *R.* Intuitively, this answer indicates that "the typical bird" flies. Note that the introduction of P_0 is similar to the Skolemization of the formula 3x(fly(x)), except for the intuitive interpretation of the Skolem constant introduced.

Using the idea of "typical individuals" we can extend our concepts to consider answers that, in addition to indicating the appropriate substitutions as before, possibly including "typical individuals", point out some "atypical individuals" or even all "atypical individuals".

5. CONCLUSIONS

Weak model elimination offers an interesting alternative for the development of logic programming systems since it works with classes of programs and queries which are more general than those commonly considered. Section 3 established soundness and completeness results for computing answers. Section 4 extended these results to a special class of defaults. In this case, the notion of answer must be appropriately revised to include the concept of "typical individual" thus avoiding arbitrary components.

REFERENCES

Casanova, M.A., R A T . Guerreiro and A. Silva [1988]. "Foundations of logic Programming based on Model Elimination", Technical Report CCR073, Rio Scientific Center, IBM Brazil (Dec. 1988).

Clark, K.L. [1978]. "Negation as Failure", in *Logic and Databases,* H. Gallaire e J. Minker (eds.), Plenum Press.

Guerreiro, R.A.T., Silva A. and Casanova, M.A. [1989]. "Foundations of Logic Programming with Defaults", (technical report in preparation).

Green, C. [1969]. "Applications of Theorem Proving to Problem Solving", UCAI-69, Washington, D C , 219-239.

Lloyd, J.W. [1984]. *Foundations of Logic Programmingy* Springer-Verlag.

Lloyd, J.W. and R.W. Topor [1985]. "Making Prolog more Expressive", J. Logic Programming 2:2, 93-109. *Journal of the ACM* 16:3, 349-363.

Loveland, D.W. [1978]. *Automated Theorem Proving: a Logical Basis,* North-Holland Publishing Company, Amsterdam.

Reiter, R. [1978]. "On Closed World Databases", in *Logic and Databases, H* Gallaire and J. Minker (eds.), Plenum Press.

Reiter, R. [1980]. "A logic for default reasoning", *Artificial Intelligence* 13,81-132.

Silva, A., Guerreiro, R.A.T. and Casanova, M.A. [1989], "ME-D: A General Clause Logic Programming System with Defaults", (in preparation).