

COMPUTING ANSWERS IN DEFAULT LOGIC

Ramiro A. de T. Guerreiro, Andrea Silva, Marco A. Casanova

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

ABSTRACT

A class of logic programming systems based on Reiter's default logic is discussed. The first question addressed is how to compute answers when a query is a set of clauses and a program is a set of clauses and defaults. Here, the clauses are not restricted to be Horn clauses and the defaults are a special case of Reiter's open normal defaults. Then, the problem of computing answers is rediscussed in the context of the concept of typical elements. This new concept of typicality is carefully defined and given a clean semantics.

1. INTRODUCTION

This paper addresses the logic foundations of a class of nonmonotonic logic programming systems based on Reiter's default logic [12]. It establishes soundness and completeness results with respect to computing answers to queries posed to logic programs with defaults. The paper also introduces the concept of typical element as a way of avoiding certain problems raised by the notion of an answer in the context of such programs.

Logic programming systems with the expressive power of default logic are interesting because defaults have a simple intuitive appeal. Using defaults, one can explicitly express that a conclusion β can be inferred if a condition α is satisfied and there is no evidence against β . Also, a proof theory for default logic is available, although not completely investigated.

Formalizations of the concept of typicality appear in many different contexts. Probability theory, fuzzy reasoning or likelihood reasoning treat statements like "Typically birds can fly" as "Most birds fly", i.e., they identify prototypical properties with statistical properties. However, Reiter [13] argues that such identification is not appropriate and that nonmonotonic logics are a better alternative. Indeed, Levesque [7] introduces the concept of typical P , written ∇P , as a new predicate. Using this representation, Levesque shows how to structure typicality to deal with problems raised by the interaction between defaults. McCarthy [6] presents a more complex structure, the unary predicates abs , for representing typicality as the negation of abnormality in the

application of circumscription to formalizing commonsense knowledge.

Our formalization of typicality differs from these in the following aspects. We first observe that typicality seems to be essentially context-dependent. For instance, a typical bird which flies may not be a typical bird which swims. Or a typical bird in Alaska may not be a typical bird in Brazil. In this case, to talk about typical birds without specifying the context does not make sense. Thus it is difficult to capture typicality attaching it to just one predicate. We then base our approach on default logic and formalize typicality as a special quantifier, Λ , governing implications, that is, we consider expressions of the form

$\Lambda \bar{x}(C \Rightarrow \alpha)$, where C is a relation on the Herbrand universe and \bar{x} is a list of the variables in α whose length is equal to the arity of C . The semantics of $\Lambda \bar{x}(C \Rightarrow \alpha)$ is "If $\bar{e} \in C$ then $\alpha\{\bar{x}/\bar{e}\}$ ". Then, C gives the context of the typical elements.

A detailed account of the results reported here can be found in Guerreiro et alii [4].

The organization of this paper is as follows. Section 2 introduces a special class of open normal defaults called generic clausal defaults and discusses the notions of program and query. It also presents a new version of the top down proof theory [12], adapted to these concepts. Section 3 discusses the question of computed answers in the scope of programs with defaults. Section 4 rediscusses computed answers introducing the concept of typical elements. Finally, section 5 contains the conclusions.

2. GENERIC CLAUSAL DEFAULTS, PROGRAM WITH DEFAULTS AND QUERIES

In this section, we introduce the notion of logic programs as a special type of default theories and we interpret queries as special type of formulas. We also informally introduce a new notion of proof in default logic, which simplifies the original definition. The appendix contains the formal definition.

A *generic clausal default* is any expression of the form $A:C$, where the *prerequisite* A of the default is a disjunc-

tion of conjunctions of literals, and the *consequent* C of the default is a conjunction of disjunctions of literals. We also accept $:C$ as a generic clausal default. If \mathbf{D} is a set of generic clausal defaults, we will denote by *consequent*(\mathbf{D}) the set of the consequents of all defaults in \mathbf{D} .

The generic clausal default $A:C$ should be understood as just a convenient notation for the open normal default $\frac{A:MC}{C}$, in the notation of Reiter [1980]. Therefore, since we limit ourselves to generic clausal defaults, we will be concerned with a particular case of open normal defaults.

A *clause* C is a disjunction of literals. A *program with defaults* is a pair $\Delta=(\mathbf{D},\mathbf{P})$, where \mathbf{P} is a finite set of clauses and \mathbf{D} is a finite set of generic clausal defaults. A *query* Q is a disjunction of conjunctions of literals. A query is *definite* iff it is a single conjunction of literals, otherwise it is *indefinite* [11].

The semantics for programs with defaults follows from the concept of extensions for open default theories. Then, given a program with defaults $\Delta=(\mathbf{D},\mathbf{P})$, a generic clausal default $A:C$ in \mathbf{D} should be interpreted as a generator for the set of defaults $A\theta:C\theta$, for all substitutions θ such that $A\theta$ and $C\theta$ are ground. The generic clausal default $A\theta:C\theta$ therefore reads: $C\theta$ can be assumed "by default" if the prerequisite $A\theta$ is known and $C\theta$ is consistent with all known facts.

We will use linear resolution with factoring [10] as the underlying refutation method to default proofs.

Let $\Delta=(\mathbf{D},\mathbf{P})$ be a program with defaults and let Q be a query. Let $CL(\neg\exists Q)$ denote a clausal representation of the negation of the existential closure of Q . Intuitively, a *refutation sequence with defaults* from Δ and Q proceeds in stages. The first stage tries to refute the query from the clauses in \mathbf{P} , perhaps augmented with the consequents of some defaults in \mathbf{D} . Each subsequent stage tries to sanction the use of the defaults in the previous stage. The last stage tests if the consequents of the defaults fired and the program are simultaneously consistent.

To sanction the use of a default, we must find an instantiation of its prerequisite that is a logical consequence of \mathbf{P} , possibly augmented with the consequents of other defaults in \mathbf{D} . The new uses of defaults need also be sanctioned. So we proceed from stage to stage till we reach a stage that uses no defaults.

However, to construct refutation sequences with defaults, we have to work with *marked clauses*, which are triples of the form $(C,\mathbf{M},\mathbf{N})$, where C is a clause, \mathbf{M} is

a list of *default literals* and \mathbf{N} is a list containing at most one single prerequisite literal.

Intuitively, for each use of a default, there is an occurrence of a default literal in \mathbf{M} recording the substitutions affecting the variables of that default. The default literals are introduced by marking the programs as follows.

Given a program $\Delta=(\mathbf{D},\mathbf{P})$, the *marking* of $\Delta=(\mathbf{D},\mathbf{P})$ is the set of triples of the form (C_i,λ,λ) , where $C_i\in\mathbf{P}$, or the form $(C_i,[\delta_i(\bar{x}_i)],\lambda)$, for each default $A_i:C_i$ in \mathbf{D} , where \bar{x}_i is a list of the variables occurring in A_i and C_i and δ_i is a new predicate symbol whose arity is the length of \bar{x}_i . A query does not contribute with default or prerequisite literals. Hence, the *marking* of Q consists of the set of pairs (C_i,λ,λ) , where $C_i\in CL(\neg\exists Q)$. The default literal $\delta_i(\bar{x}_i)$ will record the substitutions applied to the variables of the default $A_i:C_i$.

The inference rules are modified to create the list of default literals of the resolvents and factors appropriately (see the appendix for a formal definition of the modified inference rules).

A prerequisite literal will simultaneously trace in one stage all the substitutions performed on all variables of the prerequisites of all defaults used in the previous stage. Formally, the prerequisite literal in the $(i+1)^{th}$ stage is a literal of the form $\pi_i(\bar{y}_i)$ where \bar{y}_i is a list of all variables occurring in the prerequisites of all defaults used in the i^{th} stage and π_i is a new predicative symbol whose arity is the length of \bar{y}_i .

The inference rules are also modified to use appropriately the prerequisite literals (see appendix).

We conclude this section with a brief example that illustrates the concept of refutation sequence with defaults.

Example

Let $\Delta_1=(\mathbf{P}_1,\mathbf{D}_1)$ be a program with defaults where $\mathbf{P}_1=\{p(a)\vee q(b)\}$ and $\mathbf{D}_1=\{p(x)\vee q(x):r(x)\}$. Let Q be $r(y)$.

First observe that the following set of triples is the marking of Δ_1 :

1. $(p(a)\vee q(b), \lambda, \lambda)$ (originating from \mathbf{P}_1)
2. $(r(x), [\delta(x)], \lambda)$ (originating from \mathbf{D}_1)

A refutation sequence with defaults R for Δ_1 and Q would be constructed as follows. The first stage of R is a marked refutation from the marking of Δ_1 and Q :

- 0.1 $(r(x), [\delta(x)], \lambda)$ (originating from D_1)
 0.2 $(\neg r(y), \lambda, \lambda)$ (originating from the negation of Q)
 0.3 $(\square, [\delta(y)], \lambda)$ (marked resolution of 0.2 and 0.1)

Note that $\delta(y)$ in 0.3 indicates that this refutation uses the instance $p(y)\vee q(y):r(y)$ of the original default in D . Hence, the next stage of R must try to sanction this use by trying to construct a refutation from the clauses in the marking of Δ_1 and the marked clauses from a clausal representation of $\neg(p(y)\vee q(y))$ (shown in 1.2 and 1.3 below). Let the prerequisite literal for this stage be $\pi(y)$, which is a valid assumption since y is the only variable in $p(y)\vee q(y)$. Then, the second stage will begin as follows:

- 1.1 $(p(a)\vee q(b), \lambda, \lambda)$ (originating from P_1)
 1.2 $(\neg p(y), [\delta(y)], [\pi(y)])$ (from the prerequisite)
 1.3 $(\neg q(y), [\delta(y)], [\pi(y)])$ (from the prerequisite)
 1.4 $(p(a), [\delta(b)], [\pi(b)])$
 (marked resolution of 1.3 and 1.1)

Observe that the deduction cannot proceed. A marked resolution of 1.4 and 1.2, where the m.g.u. employed in the resolution would be $\theta = \{y/a\}$ is not allowed, since the prerequisite literal in 1.4 is $\pi(b)$ and the prerequisite literal in 1.2, after applying θ , would be $\pi(a)$, which does not unify with $\pi(b)$.

Without this extra unification of prerequisite literals, we would indeed complete the above deduction to be a refutation, but using incompatible instantiations of 1.2 and 1.3. That is, there would be no instantiation of the default $p(y)\vee q(y):r(y)$ such that, after instantiated, the negation of the prerequisite would generate the clauses $\neg p(a)$ and $\neg q(b)$, as required to complete the above deduction. If we followed the notions introduced in [12], we would obtain the refutation and then verify that it is not a top down proof because the instantiations are incompatible. Using the prerequisite literal we are able to block the refutation as soon as an incompatible instantiation occurs.

3. COMPUTING ANSWERS

In this section we will use $F\{\bar{y}/\bar{t}\}$ to denote the formula obtained by applying the substitution $\{\bar{y}/\bar{t}\}$ to a formula F and we use $\forall G$ to denote the universal closure of a formula G .

In the previous sections, given a program with defaults Δ and a query Q , we were concerned only with verifying if there is an extension E of Δ such that $\exists \bar{y} Q \in E$. In this section we will show how to find a tuple of terms \bar{t} such that $\forall(Q\{\bar{y}/\bar{t}\}) \in E$.

An *answer* A to a query Q over a program Δ is either **False** or a disjunction of instances of conjunctions in Q over the alphabet of Δ and Q , that is, a disjunction of conjunctions obtained from those in Q by substituting variables by terms over the alphabet in question. To register such substitutions, we will use answer literals in the same way we used default literals in the section 2. An answer is *definite* iff it consists of a single conjunction, otherwise it is *indefinite* [11].

Considering birds, the question "Who flies?" may be answered as "Every typical bird flies". In this sentence, "typical" means a subset of the universe of birds, which in general may not be defined by a first-order formula. Thus, we propose to work directly with subsets of the universe of birds. More generally, we will work with relations in the Herbrand universe. Hence, we introduce a new concept, called relativized answer. An *answer* of A , *relativized* by a relation R on the Herbrand universe of the alphabet in question, to a query Q over a program Δ is either **False** or an expression of the form $\Lambda \bar{y}(R \Rightarrow K)$, where K is an answer to the query Q over the program Δ and \bar{y} is a list of some of the variables of Q with the same length as the arity of R . The variables in \bar{y} are called the *connection variables*. The intended meaning of such expression is "If $\bar{e} \in R$ then $K\{\bar{y}/\bar{e}\}$ ".

An answer A to Q over $\Delta = (D, P)$ is *correct* iff there is an extension E of Δ such that E logically implies $\forall A$. This definition extends to relativized answers. A relativized answer $\Lambda \bar{y}(R \Rightarrow K)$ to Q over $\Delta = (D, P)$ is *correct* iff, for every element \bar{e} of R , there is an extension E of Δ such that E logically implies $\forall(K\{\bar{y}/\bar{e}\})$.

We say that a Herbrand structure H is a *model* for an open formula F iff H is a model for $\forall F$. A Herbrand structure H is a *model* for a relativized answer $\Lambda \bar{y}(R \Rightarrow K)$ iff H is a model for $K\{\bar{y}/\bar{e}\}$, for each $\bar{e} \in R$.

Given two answers (resp. relativized answers) A_1 and A_2 to Q over Δ , we say that A_1 is *more general than* A_2 iff every Herbrand model of A_1 is also a model of A_2 .

The inference rules will now work with quadruples of the form (C, L, M, N) , where C is a clause, L is a list of answer literals for computing answers of a query, M is a list of default literals for monitoring the defaults used in a refutation and N is a list containing the prerequisite literal for tracing the proof of a prerequisite or is the empty list λ .

More precisely, the *activation and marking* of a program Δ is defined exactly as the marking of Δ , except that the list of answer literals is the empty list, for every activated and marked clause originating from Δ . The *activation and marking* of a query Q of the form $Q_1 \vee \dots \vee Q_n$ is the

set of quadruples of the form $(\sim Q_i, [r_i(\bar{x}_i)], \lambda, \lambda)$, for $i = 1, \dots, n$, where $\sim Q_i$, by convention, is the clause consisting of the complement of the literals of Q_i , \bar{x}_i is a list of the variables of Q_i and the *answer literal* r_i for Q_i is a predicate symbol, not in the original alphabet, whose arity is equal to the length of \bar{x}_i . The answer literals record all the substitutions applied to variables of clauses from the query during the construction of each refutation in a refutation sequence with defaults and in the consistency test.

For that purpose, the inference rules and the notion of refutation with defaults are modified to account for answer literals similarly to what was described in section 2 for default literals.

The definition of the refutation with defaults immediately induces the following notion of computed answer. Let $\Delta = (\mathbf{D}, \mathbf{P})$ be a program with defaults and \mathbf{Q} be a query to Δ . An answer \mathbf{A} to \mathbf{Q} over Δ is *computed by defaults* iff there exists a refutation with defaults $\mathbf{R} = (R_0, \dots, R_k)$ from the activation and marking of Δ and \mathbf{Q} such that the last refutation in R_k ends in $(\square, \mathbf{S}, \mathbf{M}, \mathbf{N})$, the consistency test for R uses the substitution θ and either $\mathbf{S} = \lambda$, in which case \mathbf{A} must be equal to **False**, or $\mathbf{S} \neq \lambda$ and \mathbf{A} is a disjunction of all conjunctions \mathbf{B} such that there is $(\sim Q_i, [r_i(\bar{x}_i)], \lambda, \lambda)$ in the activation and marking of \mathbf{Q} and there is $r_i(\bar{t}) \in \mathbf{S}$ such that \mathbf{B} is equal to $Q_i \gamma \theta$, where $\gamma = \{\bar{x}_i/\bar{t}\}$.

For example, the following set of quadruples is the activation and marking of the program with defaults $\Delta = (\mathbf{D}, \mathbf{P})$ where $\mathbf{P} = \{\text{bird}(x)\}$ and $\mathbf{D} = \{\text{bird}(y):\text{fly}(y)\}$.

1. $(\text{bird}(x), \lambda, \lambda, \lambda)$ (originating from \mathbf{P})
2. $(\text{fly}(y), \lambda, [\delta(y)], \lambda)$ (originating from \mathbf{D})

Suppose that the query is $\mathbf{Q} = \text{fly}(\text{canary})$. Consider the refutation sequence with defaults $\mathbf{R} = (R_0, R_1)$, from Δ and \mathbf{Q} , constructed as follows. First, R_0 consists of just one activated and marked refutation from the activation and marking of Δ and \mathbf{Q} :

- 0.1 $(\text{fly}(y), \lambda, [\delta(y)], \lambda)$ (originating from \mathbf{D})
- 0.2 $(\neg \text{fly}(\text{canary}), [r_1], \lambda, \lambda)$ (originating from \mathbf{Q})
- 0.3 $(\square, [r_1], [\delta(\text{canary})], \lambda)$
(activated and marked resolution of 0.2 and 0.1)

Note that R_0 returns the only default fired, which is $\text{bird}(\text{canary}):\text{fly}(\text{canary})$. Hence, R_1 must be a sequence containing only one activated and marked refutations from the clauses in the activation and marking of Δ and the clause representing the negation of the prerequisite of the default fired:

- 1.1 $(\text{bird}(x), \lambda, \lambda, \lambda)$ (originating from \mathbf{P})
- 1.2 $(\neg \text{bird}(\text{canary}), [r_1], [\delta(\text{canary})], [\pi(\text{canary})])$
(from the prerequisite)
- 1.3 $(\square, [r_1], [\delta(\text{canary})], [\pi(\text{canary})])$
(activated and marked resolution of 1.2 and 1.1)

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

The consistency test of the set $\mathbf{E} = \{\text{bird}(z), \text{fly}(\text{canary})\}$ succeeds. Then, $\mathbf{A} = \text{fly}(\text{canary})$ is a correct answer of \mathbf{Q} over Δ . Note that the substitution θ used in the consistency test was the empty substitution.

4. TYPICAL ELEMENTS

The definition of computed answer given in section 3 is hard to work with, though, because it admits answers with arbitrary instantiations, 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 refutation sequences with defaults. This problem also occurs if we use the original proof theory for default logic. The need of instantiations coming from the consistency test can be overcome by defining a relativized computed answer.

For example, let $\Delta = (\mathbf{D}, \mathbf{P})$ be a program, where $\mathbf{P} = \{\text{bird}(z), \neg \text{fly}(\text{penguin}), \neg \text{fly}(\text{ostrich}), \text{yellow}(\text{canary})\}$ and $\mathbf{D} = \{\text{bird}(y):\text{fly}(y)\}$. Consider the query $\mathbf{Q} = \text{fly}(x)$. Consider the refutation sequence with defaults $\mathbf{R} = (R_0, R_1)$, from Δ and \mathbf{Q} , constructed as follows. First, R_0 is a sequence containing only one activated and marked refutation from the activation and marking of Δ and \mathbf{Q} :

- 0.1 $(\text{fly}(y), \lambda, [\delta(y)], \lambda)$ (originating from Δ)
- 0.2 $(\neg \text{fly}(x), [r(x)], \lambda, \lambda)$ (originating from \mathbf{Q})
- 0.3 $(\square, [r(x)], [\delta(x)], \lambda)$
(activated and marked resolution of 0.2 by 0.1)

Note that R_0 returns only the default $\text{bird}(x):\text{fly}(x)$. Hence, R_1 must be a sequence containing only one marked and activated refutation from the clauses in the marking and activation of Δ and the clause representing the negation of the prerequisite of $\text{bird}(x) : \text{fly}(x)$ (the clause (1.2) below):

- 1.1 $(\text{bird}(z), \lambda, \lambda, \lambda)$ (from Δ)
- 1.2 $(\neg \text{bird}(x), [r(x)], [\delta(x)], [\pi(x)])$
(from the prerequisite)
- 1.3 $(\square, [r(x)], [\delta(x)], [\pi(x)])$
(activated and marked resolution of 1.2 by 1.1)

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

Note that the choice of θ is entirely arbitrary. On the other hand, it is not possible to ignore θ , since the set $\{\text{bird}(z), \neg\text{fly}(\text{penguin}), \neg\text{fly}(\text{ostrich}), \text{yellow}(\text{canary})\} \cup \{\text{fly}(x)\}$, is not satisfiable. Intuitively, the default in \mathbf{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 appearing in \mathbf{P} , \mathbf{D} and \mathbf{Q} , 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 refutation with defaults R , except that the substitution of the consistency test is now, by definition, $\theta = \{x/p_0\}$. Since, for this choice of θ , the set $\{\text{bird}(z), \neg\text{fly}(\text{penguin}), \neg\text{fly}(\text{ostrich}), \text{yellow}(\text{canary})\} \cup \{\text{fly}(x)\theta\}$, is consistent, we have that $\text{fly}(p_0)$ is the new computed answer by the 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 $\exists x(\text{fly}(x))$, except for the intuitive interpretation of the Skolem constant introduced (i.e., the typical element).

We now formally define relativized answers computed by defaults. Let $\Delta = (\mathbf{D}, \mathbf{P})$ be a program with defaults and \mathbf{Q} be a query to Δ . Let \mathcal{L} be the first-order language used. A *relativized answer* \mathbf{A} to \mathbf{Q} over Δ is computed by defaults iff there exists a candidate refutation with defaults (R_1, \dots, R_k) from the activation and marking of Δ and \mathbf{Q} such that the last refutation in R_k ends in $(\square, \mathbf{S}, \mathbf{M}, \mathbf{N})$, \mathbf{C} is the set of consequents of all defaults returned by R_k , and either $\mathbf{S} = \lambda$, in which case \mathbf{A} must be equal to **False**, or $\mathbf{S} \neq \lambda$ and \mathbf{A} is of the form $\Lambda \bar{y}(\text{CONSIST}[\mathbf{P}, \mathbf{C}] \Rightarrow \mathbf{K})$, where \mathbf{K} is a disjunction of all conjunctions \mathbf{B} such that there is $(\sim \mathbf{Q}_i, [r_i(\bar{x}_i)], \lambda, \lambda)$ in the activation and marking of \mathbf{Q} and there is $r_i(\bar{t}) \in \mathbf{S}$ such that \mathbf{B} is equal to $\mathbf{Q}_i \gamma$, where $\gamma = \{\bar{x}_i/\bar{t}\}$, and \bar{y} is the list of all variables in \mathbf{C} . We define $\text{CONSIST}[\mathbf{P}, \mathbf{C}]$ as the relation in the Herbrand universe of \mathcal{L} as follows:

$\bar{e} \in \text{CONSIST}[\mathbf{P}, \mathbf{C}]$ iff $\mathbf{P} \cup \mathbf{C}\{\bar{y}/\bar{e}\}$ is consistent

where \bar{y} be the list of all variables in \mathbf{C} .

Given a relativized answer $\Lambda \bar{y}(\text{CONSIST}[\mathbf{P}, \mathbf{C}] \Rightarrow \mathbf{K})$, computed by defaults, the tuples in $\text{CONSIST}[\mathbf{P}, \mathbf{C}]$ are

called *typical elements*. Assume that $\text{CONSIST}[\mathbf{P}, \mathbf{C}]$ has arity n , then the n -tuples over the Herbrand universe that are not typical elements are called *atypical elements*.

Consider again our previous example, where $\mathbf{P} = \{\text{bird}(z), \neg\text{fly}(\text{penguin}), \neg\text{fly}(\text{ostrich}), \text{yellow}(\text{canary})\}$ and $\mathbf{D} = \{\text{bird}(y):\text{fly}(y)\}$, and $\mathbf{Q} = \text{fly}(x)$. The relativized answer computed by defaults is $\Lambda x(\text{CONSIST}[\mathbf{P}, \{\text{fly}(x)\}] \Rightarrow \text{fly}(x))$. Note that this relativized answer is more general than, for example, $\text{fly}(\text{canary})$. Actually, for this relativized answer, canary is a typical element and penguin is an atypical element.

Theorem 1: (Soundness and Completeness Theorem for Relativized Answers computed by defaults)

Let \mathcal{L} be the first-order logic language, $\Delta = (\mathbf{D}, \mathbf{P})$ be a program with defaults and \mathbf{Q} be a query to Δ .

- (a) Every relativized answer computed by defaults to \mathbf{Q} over Δ is correct.
- (b) Given any correct answer to \mathbf{Q} over Δ , there is a relativized answer computed by defaults which is more general.

The proof of this theorem is presented in [4].

Note that a relativized answer computed by defaults may reduce to a kind of tautology, and so it becomes void of information, if $\text{CONSIST}[\mathbf{P}, \mathbf{C}]$ is the empty relation. Therefore it is quite useful to assure that $\text{CONSIST}[\mathbf{P}, \mathbf{C}]$ is not empty. Observe that if $\text{CONSIST}[\mathbf{P}, \mathbf{C}]$ is not empty then $\mathbf{P} \cup \exists \mathbf{C}$ is consistent. So we can apply the Skolem theorem to check the consistency of $\mathbf{P} \cup \mathbf{C}_s$, where \mathbf{C}_s is the Skolemization of \mathbf{C} .

Unfortunately, the consistency of $\mathbf{P} \cup \mathbf{C}_s$ does not entail $\text{CONSIST}[\mathbf{P}, \mathbf{C}] \neq \emptyset$. For instance, assume that the Herbrand universe has only the element \mathbf{a} , \mathbf{P} is $\{\neg\text{fly}(\mathbf{a})\}$ and \mathbf{C} is $\{\text{fly}(x)\}$. Then $\mathbf{P} \cup \exists \mathbf{C}$ is consistent, but $\text{CONSIST}[\mathbf{P}, \mathbf{C}] = \emptyset$, where $\exists \mathbf{C}$ means the existential closure of all clauses in \mathbf{C} . To overcome this problem, we may assume that \mathcal{L} contains an enumerable family of constants not appearing in \mathbf{P} , \mathbf{D} or \mathbf{Q} . So the above-mentioned Skolemization will use these constants, which are called *typical constants* and are generally denoted with subscript 0.

Consider now the following example. Let $\Delta = (\mathbf{D}, \mathbf{P})$ be a program, where $\mathbf{P} = \{\text{bird}(z), \neg\text{fly}(\text{penguin})\}$ and $\mathbf{D} = \{\text{bird}(x):\text{fly}(x)\}$, and $\mathbf{Q} = \text{fly}(y)$ be the query. If we assume the first-order language to be the smallest possible one, then the Herbrand universe will be $\{\text{penguin}\}$, $\text{CONSIST}[\mathbf{P}, \{\text{fly}(x)\}]$ will be the empty relation and the relativized answer computed by defaults will be trivially true. Actually, as an open default stands for the family of its ground instances over the Herbrand universe, it is

not possible to prove that $\exists y \text{fly}(y)$. Therefore, the poverty of \mathcal{L} allows the statement $\neg \text{fly}(\text{penguin})$ about one atypical bird to block completely the default $\text{bird}(y) : \text{fly}(y)$, which refers to birds in general. Therefore it is a good practice to assume that the first-order language contains the typical constants. With this assumption, in the above example, the Herbrand universe contains enumerably many constants different from penguin.

Finally, we observe that the consistency test sanctioning a refutation sequence with defaults can be implemented as a test for the finite failure to refute the consequents, together with the original set of clauses from the program.

5. CONCLUSIONS

Default logic offers an interesting alternative for the development of logic programming systems for nonmonotonic reasoning. Sections 3 and 4 established the theoretical foundations of such systems, stating a soundness and completeness theorem for computing answer to queries posed to logic programs with defaults. The notion of answer was appropriately revised to include the concepts of typical element and typical constant.

REFERENCES

- [1] Aida, H., H. Tanaka and T. Moto-Oka, "A Prolog Extension for Handling Negative Knowledge", *New Generation Computing*, vol. 1, no. 1, p. 87, 1983.
- [2] Bowen, K.A., "Programming with Full First-Order Logic", in *Machine Intelligence*, vol. 10, p. 421, 1982.
- [3] Clark, K.L., "Negation as Failure", in *Logic and Databases*, H. Gallaire e J. Minker (eds.), Plenum Press, 1978.
- [4] Guerreiro, R.A.T., A. Silva and M. A. Casanova, "Foundations of Logic Programming with Defaults", (technical report in preparation)
- [5] Green, C., "Applications of Theorem Proving to Problem Solving", *IJCAI-69*, Washington, D.C., p. 219, 1969.
- [6] Levesque, H.J., "Foundations of a functional approach to knowledge representation", *Artificial Intelligence*, vol. 23, p. 155, 1982.
- [7] McCarthy, J., "Applications of circumscription to formalizing commonsense knowledge", *Artificial Intelligence*, vol. 28, p. 89, 1986.
- [8] Lloyd, J.W., *Foundations of Logic Programming*, Springer-Verlag, 1984.
- [9] Loveland, D.W., "A simplified format for the model elimination theorem-proving procedure", *Journal of the ACM*, vol. 16, no. 3, p. 349, 1969.
- [10] Loveland, D.W., *Automated Theorem Proving: a Logical Basis*, North-Holland Publishing Company, Amsterdam, 1978.
- [11] Reiter, R., "On Closed World Databases", in *Logic and Databases*, H. Gallaire and J. Minker (eds.), Plenum Press, 1978.
- [12] Reiter, R., "A logic for default reasoning", *Artificial Intelligence*, vol. 13, p. 81, 1980.
- [13] Reiter, R., "Nonmonotonic Reasoning", in *Exploring Artificial Intelligence: Survey Talks from the National Conferences on Artificial Intelligence*, California, p. 439, San Mateo: Morgan Kaufmann Publishers, 1988.
- [14] Silva, A., R.A.T. Guerreiro and M.A. Casanova, "ME-D: A General Clause Logic Programming System with Defaults", (technical report in preparation), 1989.
- [15] Stickel, M.E., "A PROLOG technology theorem prover", *New Generation Computing*, vol. 2, p. 371, 1984.

APPENDIX

In this appendix we formally define a new notion of proof in default logic.

A marked clause (A, M, N) is a *marked factor* of a marked clause (A', M', N') if and only if A is a factor of A' with m.g.u. θ , $M = M'\theta$ and $N = N'\theta$. A marked clause (A, M, N) is a *marked resolvent* of $C' = (A', M', N')$ and $C'' = (A'', M'', N'')$ if and only if β is a renaming of some variables of C' such that C' and $C''\beta$ do not have variables in common, $A = \bar{A}\alpha$ where \bar{A} is a resolvent of A' and A'' , with m.g.u. θ , α is a m.g.u. of the prerequisite literals in $N'\theta$ and $N''\beta\theta$ (remember that each of these lists never has more than one element), $M = M'\theta \cup M''\beta\theta\alpha$ and $N = N'\theta\alpha$. If N' or N'' is the empty list, α is the identity substitution. A resolution of A' by A'' can be blocked if there is no m.g.u. for the prerequisite literals in $N'\theta$ and $N''\beta\theta$.

A *marked linear resolution refutation* R , or *marked refutation* in our context, with *initial marked clause* A , from a set P of marked clauses is a sequence of marked clauses of the form:

- 1) $R = (R_0, \dots, R_{p-1}, R_p, \dots, R_n)$ where $R_0, \dots, R_{p-1} \in P$ and $R_p = A$,
- 2) for $p < i \leq n$, R_i is a marked factor of R_{i-1} or R_i is a marked resolvent of R_{i-1} and R_j , for some $j < i$, where R_j is called *auxiliary clause*,
- 3) $R_n = (\square, M_n, N_n)$.

Let R be a marked refutation from the clauses in the marking of Δ and Q . Suppose that R terminates in (\square, S, T) . A default ψ is *returned* by R iff there is a triple

$(C_i, [\delta_i(\bar{x}_i)], \lambda)$ in the marking of Δ , corresponding to some default $A_i; C_i$ in \mathbf{D} , and there exists a literal of the form $\delta_i(\bar{t})$ in \mathbf{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 \mathbf{D} and $(C_i, [\delta_i(\bar{x}_i)], \lambda)$ be the corresponding triple in the marking of Δ . This default is *fired* in R iff there is a marked clause in R derived by marked resolution with $(C_i, [\delta_i(\bar{x}_i)], \lambda)$ as auxiliary clause. Then, each default $A_i; C_i$ fired in R , as well as each default corresponding to a default literal in the initial clause of R , if any, generates a *descendent* default in the set of defaults returned by R .

A *candidate refutation sequence with defaults* from a program $\Delta = (\mathbf{D}, \mathbf{P})$ and a query \mathbf{Q} is a finite sequence $\mathbf{R} = (R_0, \dots, R_k)$ of sequences of marked refutations such that:

- 1) R_0 is composed by only one marked refutation from the marking of Δ and the marking of \mathbf{Q} , with initial clause in the marking of \mathbf{Q} ;
- 2) for $0 \leq i \leq k$:
 - let \mathbf{D}_i be the set of defaults returned by the last marked refutation in R_i ,
 - let \mathbf{M}_i be the set of default literals corresponding to the defaults in \mathbf{D}_i ,
 - let $\mathbf{D}^{(i)} = \{A_{i,j}; C_{i,j} \mid 1 \leq j \leq n_i\}$ be $\mathbf{D}^{(i)}$ be the set of defaults in \mathbf{D}_i which are the descendents of the defaults fired in all marked refutations in R_i ,
 - let $\mathbf{N}_i = \{\pi_i(\bar{X})\}$, where \bar{x} is the list of all variables in $\{A_{i,j} \mid 1 \leq j \leq n_i\}$, that is, let \mathbf{N}_i be an one-element list composed of the prerequisite literal for the prerequisites of all defaults in $\mathbf{D}^{(i)}$,
 - for $1 \leq j \leq n_i$, let $CL(\neg \exists A_{i,j}) = \{B_{i,j,q} \mid 1 \leq q \leq m_{i,j}\}$ be a clausal representation of $\neg \exists A_{i,j}$ and if θ is a substitution over some variables of \mathbf{N}_i , let $\mu(A_{i,j}, \theta) = \{B_{i,j,q}\theta, \mathbf{M}_i\theta, \mathbf{N}_i\theta \mid 1 \leq q \leq m_{i,j}\}$.

Then:

- for $0 \leq i \leq k-1$, R_{i+1} is the sequence $\{R_{i+1,j}\}_{1 \leq j \leq n_i}$ where,
 - $R_{i+1,1}$ is a marked refutation from the marking of Δ and $\mu(A_{i,1}, \epsilon)$, with initial clause in $\mu(A_{i,1}, \epsilon)$. The last clause in $R_{i+1,1}$ is $(\square, \mathbf{M}_{i+1,1}, \mathbf{N}_i\theta_1)$, where θ_1 is a substitution over some variables of \mathbf{N}_i .
 - for $2 \leq j \leq n_i$, $R_{i+1,j}$ is a marked refutation from the marking of Δ and $\mu(A_{i,j}, \theta_{j-1})$, with initial clause in $\mu(A_{i,j}, \theta_{j-1})$. The last clause in $R_{i+1,j}$ is $(\square, \mathbf{M}_{i+1,j}, \mathbf{N}_i\theta_j)$ where θ_j is a substitution over some variables of \mathbf{N}_i .
- $\mathbf{D}^{(k)} = \emptyset$.

A *refutation sequence with defaults* from $\Delta = (\mathbf{D}, \mathbf{P})$ and query \mathbf{Q} is a candidate refutation sequence with defaults that satisfies:

- (*Consistency Test*). Let \mathbf{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 \mathbf{C} by ground terms of Herbrand universe over the alphabet in question such that $\mathbf{P} \cup \mathbf{C}\theta$ is satisfiable.

Recall that the class of *top down default proofs* in [12] is defined as a restriction of the class of *admissible refutation sequences*. The restriction is actually a test that relates the descendents of the defaults fired in each refutation of an admissible refutation sequence. In our definition of refutation sequence with defaults, the prerequisite literal allow us to find out as early as possible that the admissible refutation sequence being constructed will not be a top down default proof.

More precisely, recall that the test originally defined in [12] becomes necessary in two cases:

- 1) when a resolvent is reused as an auxiliary clause in one of the refutations in the sequence;
- 2) when a refutation in the sequence uses more than once a clause originating from the negation of the prerequisite of a default fired in the previous refutation.

However, in our approach, these cases are treated through the use of prerequisite literals, which were not introduced in the original proof theory for default logic.