New Generation Computing, 4 (1986) 35-50
OHMSHA, LTD. and Springer-Verlag

NEW GENERATION COMPUTING

MOLOG: A System That Extends PROLOG with Modal Logic

Luis FARIÑAS DEL CERRO
Langages et Systèmes Informatiques,
Université Paul Sabatier,
118 route de Narbonne-31062 Toulouse Cedex, France.

Received 16 March 1985
Revised manuscript received 6 January 1986

Abstract In this paper we present an extension of PROLOG using modal logic. A new deduction method is also given based on a rule closer to the classical inference rule of PROLOG.

Keywords: Theorem Proving, Logic Programming, Modal Logic, Modal Resolution, Problem Solving.

§1 Introduction

Using logic as a programming language is now becoming a current idea. The best example of this is perhaps the PROLOG language. Many extensions of this language have been done in order to consider parallel programming (PARLOG[5]), metatheory[3,7] or reasoning with assumption (N-PROLOG[10]). In this framework we try to marry modal logics and PROLOG and we call it MOLOG. MOLOG can be considered as a PROLOG type language where literals and Horn clauses can be qualified by modal expressions. We think that it will be a unified way to analyse many extensions of PROLOG. Another work connected with our approach is given by Warren[15] although he considers a restricted set of modal clauses and modalities. MOLOG and Warren's approach use resolution rules as an inference rule. Another type of inference rule is given by Malachi, Manna and Waldinger,[13] who describe a language and who use a tableaux method as a deduction system.

To define MOLOG, a parallel approach to PROLOG has been considered. To define what that means, consider the following inference rule of propositional PROLOG[4,12] in terms of the logical inference:

$$\frac{\begin{array}{l} E \leftarrow B \\ \leftarrow E' \ \& \ B' \end{array}}{\leftarrow B \ \& \ B' \ ; \text{ if } \vdash_{\overline{CL}} E \rightarrow E'} \quad (1)$$

The inference rule can be applied if $E \rightarrow E'$ is a theorem of classical logic, and it is equivalent to testing that $E = E'$.

The question is: how can we generalize this rule to modal logics? A natural way is to consider the following rule:

$$\frac{\begin{array}{l} E \leftarrow B \\ \leftarrow E' \ \& \ B' \end{array}}{\leftarrow B \ \& \ B' \ ; \text{ if } \vdash_{\overline{ML}} E \rightarrow E'} \quad (2)$$

The inference rule will be applied if $E \rightarrow E'$ is a theorem of the corresponding modal logic. Now the problem is: how can we find a simple operation $T(E, E')$ testing $\vdash_{\overline{ML}} E \rightarrow E'$? For the standard modal systems (M, S4, S5, ...) this test T is not so complex; however, we have not found a general test T for an arbitrary modal logic. Consequently, for every modal system a particular operation T must be defined, depending on the corresponding modal proof theory. Another approach that can be considered without restriction is to translate modal logics to classical logic and use a classical proof theory. For this approach for example FOL[1] can be a useful tool. But some modal logics such as the system G of Gödel[2] possess a second order formula as translation. So the general translation is not to first order logic but to higher order theories. Unfortunately, it is well known that it is difficult to deduce in higher order logic.

In this paper we will define modal Horn clauses and give a deduction method for a particular modal logic. Since some of the work realized by the deduction method rules can be done during the syntactical analysis of the clauses, a new deduction method called COMPILATION is introduced. This new method is very close to the deduction method of classical logic. At the end some examples about knowledge representation and about the description of some features of PROLOG will be given.

§2 Modal Horn Clauses

The main connective in the Horn clauses is the implication symbol. However there are modal expressions that aren't equivalent to a modal clause with only one implication symbol, as for example: $\Box(p \rightarrow \Box(q \rightarrow t))$, where $\Box$ is the modal operator necessary. This brings up the question of the choice of the definition of modal Horn clauses with or without nested implication symbols. In the Gabbay and Reyle approach[10] the definition of clauses allows nested implication symbols. In our approach we consider the expression with only one implication symbol. This definition is taken because it is closer to the classical

definition of Horn clauses and consequently closer to the procedural interpretation of Horn clauses, although the language is less expressive than the language with nested implication symbols. Some of the properties obtained in the nested approach can be simulated with our definition of modal Horn clauses as it will be shown later.

We will now give the general definition of modal Horn clauses.

Definition 1 Modal operator
A **universal** modal operator is either

- the name of a modal operator, or
- an expression consisting of the name of a modal operation followed by a list $(t_1, ..., t_n)$ of terms $(1 \leq i \leq n)$

where a **term** is either

- a constant, or
- a variable, or
- an expression $f(t_1, ..., t_n)$, where f is an n-ary function and each t_i $(1 \leq i \leq n)$ is a term.

An **existential** modal operator O' is introduced by

$$O'A \stackrel{\text{def}}{=} \sim O \sim A$$

where $\sim$ is the negation symbol.
Example of modal operators:

- After(p)
- knows(Pierre)
- □
- resolve

Definition 2 Modality
A **modality** is a (eventually empty) sequence of modal operators.
Example of modalities

- □ knows(Pierre)
- knows(Pierre) After(p)
- assert(p)

Definition 3 Atomic formula
An **atomic formula** is either

- the name of a proposition, or
- an expression consisting of the name of a predicate followed by the list of arguments $(t_1, ..., t_n)$ where each t_i, $1 \leq i \leq n$, is a term

We will suppose that names of atomic formulas are taken from disjoint alpha-

bets in order to distinguish the modal operator from the atomic formulas.

Definition 4 Modal atomic formula
A **modal atomic formula** is an atomic formula governed by a modality.

It must be noted that this definition of modal atomic formulas corresponds to **unary modal operators** i. e. only one atomic formula is qualified by the modal operators. The modal atomic formulas as A UNTIL B can't be defined explicitly in this framework. We could extend the definition to n-ary modal operators but we prefer here simplicity of reasoning.

Examples of modal atomic formulas

- □ knows(Pierre) Père(x)
- knows(Pierre) After(p) $P(x)$
- assume(Jean) Père(Pierre)
- resolve p

Definition 5 Conjunctive modal formula
A **conjunctive modal formula** is either:

- a conjunction (eventually with only one conjunct) of modal atomic formulas, or
- a conjucntive modal formula governed by a modality, or
- a conjunction of conjunctive modal formulas

Examples of conjunctive modal formulas

- Père(x), Frère(y)
- knows(Pierre) (Père(x), Frère(y)), Père(z)
- assume(Pierre) p, assume(Jean) q

In the example ',' represents the conjunction operator.

Definition 6 Modal regular clauses
A **modal regular clause** is an expression of the form:

Modality (Modal atomic formula ← conjucntive modal formula)

Examples of Modal Horn clauses.

- knows(Pierre) (knows(Jean) $p(x)$ ← credible(Jean) ($p(x)$, q))
- After(p) q ← knows(Jean) p, credible(Jean) p
- p ← assume(q) t, resolve $t(x)$

Definition 7
MOLOG fact is either a modal regular clause or an atomic modal formula.

We note by MH the set of MOLOG facts.

Definition 8

MOLOG goal is an expression of the form:

$\leftarrow$ conjunctive modal formula

We note by GH the set of goal modal clauses.
So Modal Horn clauses are either MOLOG facts or goals.

§3 The Semantics of MOLOG

We will give the semantics of MOLOG in term of possible worlds.

Definition 9
A **valuation** for a modal Horn clause is a 4-tuples $V = \langle W, R_o, D, m \rangle$, where

- W is a non empty set of worlds or states
- R_o is a relation in $W \times W$, associated to each modal operator o
- D is a set of objects.
- m is a function defined in the language of modal Horn clauses as follows
  - $m(c)$ is an element belonging to D, if c is a constant symbol
  - $m(f)$ is a function of $D^n \rightarrow D$, if f is a n-ary function symbol
  - $m(p)$ is a subset of $W \times D^n$, if p is a n-ary predicate symbol.

Definition 10
A **model** M for a language of modal Horn clauses is a pair $\langle V, w_a \rangle$ where V is a valuation and w_a is a distinguished world, called **actual world**.

The relation of **satisfiability** between a model $M = \langle V, w \rangle$ and the modal Horn formulas is defined as usual by induction:

- $\langle V, w \rangle$ sat $p(t_1, ..., t_n)$ iff $(w, t_1, ..., t_n) \in m(p)$
- $\langle V, w \rangle$ sat $A \leftarrow B$ iff $\langle V, w \rangle$ sat A or not $\langle V, w \rangle$ sat B
- $\langle V, w \rangle$ sat A, B iff $\langle V, w \rangle$ sat A and $\langle V, w \rangle$ sat B
- $\langle V, w \rangle$ sat $A(x)$ iff $\langle V, w \rangle$ sat $A(t)$, for every $t \in D$
- $\langle V, w \rangle$ sat OA iff $\langle V, w' \rangle$ sat A for every w' such that wR_ow', and O is a universal modal operator
- $\langle V, w \rangle$ sat $O'A$ iff there is a w' such that wR_ow' and $\langle V, w' \rangle$ sat A, and O' is an existential modal operator

A set S of formulas is satisfied in a model M by a world w iff M sat A for all $A \in S$. S is unsatisfiable iff S is not satisfied in any model.

§4 Deduction

4.1 A Particular Modal System

In recent years deduction methods connected to the idea of resolution have been defined in modal logic.[8,9,12,14] This method can be used easily to obtain rules as follows:

$$\frac{M_1(E_1 \leftarrow B) \quad \leftarrow M_2(E_2)}{\leftarrow M_3(E_3, B);\ \text{if } T(M_1E_1, M_2E_2) \text{ is verified}} \tag{3}$$

where T possesses an associated procedure reflecting the corresponding modal proof theory and M_1, M_2, M_3 are modalities.

In order to simplify our approach we will consider, in what follows, the modal system S5, only with universal operators, and we will develop a proof theory for it. For every modal system a similar approach must be given.

Considering the system S5 as the base of our exposition means that every modal operator O, if it is universal, possesses a scheme of axioms:

- $O(A \rightarrow B) \rightarrow (OA \rightarrow OB)$
- $OA \rightarrow A$
- $O'A \rightarrow OO'A$

where $O'A \stackrel{\text{def}}{=} \sim O \sim A$, in other words O' is the existential modal operator of O.

From the semantic point of view the corresponding accesibility relation R_O is an equivalence relation.

The main property, known as the reduction theorem, of this kind of systems is that every sequence of modal operators O and O' is equivalent to the last operator of the sequence.[4)]

Since, for example $OO'OA \leftrightarrow OA$. However two different modal operators are not reductible. For example, if O_1 and O_2 are two different universal modal operators the sequence O_1O_2 is irreductible.

There are other theorems that will be useful when we consider simple Horn clauses. For example

$$O(A \leftarrow OB) \leftrightarrow OA \leftarrow OB \qquad O(A, B) \leftrightarrow OA, OB$$
$$O'(A, O'B) \leftrightarrow O'A, O'B \qquad O'(A \leftarrow OB) \leftrightarrow O'A \leftarrow OB$$
$$O'(A, OB) \leftrightarrow O'A, OB \qquad O(A \leftarrow O'B) \leftrightarrow OA \leftarrow O'B$$

The above theorems can be called **simplification** theorems.

In what follows, to simplify our reasoning, we consider a S5 modal logic in an attempt to formalize the idea of knowledge. In our case we consider the following unversal modal operators:

knows(a) A that means a knows A, where a is an arbitrary constant.

In this logic we have for example the following modal Horn clauses:

- knows(Pierre) $(p \leftarrow q)$
- knows(Pierre) (knows(Jean) $p \leftarrow$ (Jean))q
- knows(Pierre) $p \leftarrow q$, knows(e) t

4.2 Modal Resolution

Now we will give a proof theory for modal Horn clauses in the system S5 with the universal modal operators: knows(a). In other words we will define the test procedure T required at rule (3).

Definition 11
We will define a binary operation Σ on $MH \times GH$ and resolvability recursively as follows. We omit the implication symbol "$\leftarrow$" in the goal.

For classical Horn clauses:

- $\Sigma(A; A') \Rightarrow \phi$ if there is a most general unifier σ such that $A\sigma = A'\sigma$. We also say that $(A; A')$ is **resolvable** via σ.
- $\Sigma(A; B, C) \Rightarrow \Sigma(A; B), C$
  And if $(A; B)$ is resolvable, then $(A; B, C)$ is also resolvable
- $\Sigma(A \leftarrow B; C) \Rightarrow \Sigma(A; C), B$
  And if $\Sigma(A; C)$ is resolvable then $A \leftarrow B; C$ is also.

For the knows(a) modal operator:

- $\Sigma(\text{knows}(a)\ A; \Delta B) \Rightarrow (\Delta\Sigma(A; B))$ where Δ is either knows(a) or is missing.

Definition 12
If C_1 is a modal Horn clause, C_2 is a goal clause and C_1 and C_2 are resolvable, then a clause is called **resolvent** of C_1 and C_2 if it is the result of substituting:

E for every occurrence of (ϕ, E)
ϕ for every occurrence of knows(a) ϕ

in $\Sigma(C_1, C_2)$, as many times as possible.

We denote by $R(C_1, C_2)$ a resolvent of C_1 and C_2 after the simplification.
Remark: C_1 and C_2 can have several resolvents.

Definition 13
Let C_1 and C_2 be two clauses. The rule

$$\frac{C_1,\ C_2}{R(C_1,\ C_2) \text{ if } (C_1;\ C_2) \text{ is resolvable}}$$

is called the **resolution rule**.

Consequently the test procedure T in the rule (3) in this method means testing whether $(ME; ME')$ is resolvable.

Definition 14
Let S be a set of clauses, and C_0 a goal clause in S. A **deduction** of C_n from S is a finite sequence of goal clauses $C_0, C_1, \ldots, C_n$, such that each C_i $(0 \leq 1 \leq n)$

is obtained with resolution from C_{i-1} against a clause C in S.

Definition 15
A deduction of the empty clause is called a **refutation**.

We can obtain the following completeness theorem:

Theorem 1
A set of clauses S is unsatisfiable iff S is refutable.

The proof can be obtained using the following lemmas:

Let $S = \{\text{knows}(a)\ A_1, ..., \text{knows}(a)\ A_n, A_{n+1}, ..., A_m\}$ be a set of modal Horn clauses.

Lemma 1
S and the goal clause knows(a) A_{m+1} is unsatisfiable iff the set $S' = \{A_1, ..., A_n, A_{m+1}\}$ is unsatisfiable.

Lemma 2
S and the goal clause A_{m+1} is unsatisfiable iff the set $S' = \{A_1, ..., A_{m+1}\}$ is unsatisfiable.

To prove Lemmas 1 and 2 it is necessary to check that any model of S' can be extended to a model of S.

Lemma 3
Let S be a set of modal Horn clauses, A a goal clause and S' the set of clauses obtained using Lemmas 1 or 2. If R is a refutation of S' then R can be extend to a refutation of S.

The proof is raised by putting the modal operator knows(a) in its own place and using the operator Σ for the knows(a) operator.

Now to prove Theorem 1 we use an induction on the number of nested modal operators. The case without modal operator is the classical one and by Lemmas 1 and 2, the induction hypothesis and Lemma 3 the proof is given.

4.3 Example

To clarify this method we give a simple example. Let S be the following set of clauses:

(1) knows(Pierre) knows(Jean) q
(2) knows(Pierre) ($p \leftarrow$ knows(Jean) q)

And the goal clause:

(3) $\leftarrow$ knows(Pierre) p

Using the operation Σ and the resolvability relation, we obtain successively:

- Σ(knows(Pierre) ($p \leftarrow$ knows(Jean) q); knows(Pierre) p) $\Rightarrow$ knows(Pierre) $\Sigma(p \leftarrow$ knows(Jean) $q; p)$

and if ($p \leftarrow$ knows(Jean) q; p) is resolvable, then (knows(Pierre) ($p \leftarrow$ knows(Jean) q); knows(Pierre) p) is resolvable

- $\Sigma(p \leftarrow$ knows(Jean) q; $p) \Rightarrow \Sigma(p, p)$, knows(Jean) q

Since $\Sigma(p, p) = \phi$, (p, p) is resolvable then (knows(Pierre) ($p \leftarrow$ knows(Jean) q); knows(Pierre) p) is resolvable, and the inference rule can be applied, to obtain the following new goal clause:

(4) $\leftarrow$ knows(Pierre) knows(Jean) q

Using this goal clause against the clause (1) we obtain the empty goal clause. Consequently the clauses (1) and (2) can give an answer to the goal clause (3).

§5 Compilation

5.1 Rules of Compilation

In this paragraph we will give a new deduction method for the modal logic defined above which is closer to the classical resolution principle.

To obtain a simple procedure T in the rule (3) it is necessary to use simple modal Horn clauses. Then we will define a transformation that we call **compilation**, in order to obtain simple modal Horn clauses from general modal Horn clauses, and define a PROLOG like inference rule.

Definition 16

The rules of **compilation** are transformation rules defined recursively. The rules for the modal Horn clauses of the logic with knows(a) as the only modal operator, are as follows:

(1) $\mathrm{C}[M \text{ knows}(a)\,(B \leftarrow C)] \Rightarrow \begin{cases} 1. & \mathrm{C}[M(\text{knows}(a)\ B \leftarrow \text{knows}(a)\ C)] \\ 2. & \mathrm{C}[M(B \leftarrow C)] \end{cases}$

where M is a modality or is missing.

(2) $\mathrm{C}[\text{knows}(a)\ A \leftarrow B] \Rightarrow \begin{cases} 1. & \mathrm{C}[\text{knows}(a)\ A] \leftarrow \mathrm{C}[B] \\ 2. & \mathrm{C}[A] \leftarrow \mathrm{C}[B] \end{cases}$

where A is a modal atomic formula and B is a modal atomic formula or is missing.

(3) $\mathrm{C}[\leftarrow B] \Rightarrow \leftarrow \mathrm{C}[B]$

(4) $\mathrm{C}[\text{knows}(a)\ (A, B)] \Rightarrow \mathrm{C}[\text{knows}(a)\ A], \mathrm{C}[\text{knows}(a)\ B]$

(5) $\mathrm{C}[\text{knows}(a)\ \text{knows}(a)\ A] \Rightarrow \mathrm{C}[\text{knows}(a)\ A]$

(6) $\mathrm{C}[\Delta p] \Rightarrow \Delta p$ iff p is a atomic formula, where Δ is knows(a) or is missing.

We must note that for one clause, the compilation can generate many clauses.

The difficulty in the expression of the test T in the rule (3) is that the modality governs the implication, and consequently the test will be more

complicated than at the rule (2). Analogously to the elimination of the quantifiers in classical Horn clauses we have tried to eliminate this problem, by using **compilation**. After this transformation the corresponding proof theory is very close to the classical proof theory. For the standard modal systems such as Q, S4, S5 it is exactly the classical proof theory.

In the case of the considered logic for a given modal Horn clause by compilation we generate a set of clauses in the form

$$B_0 \leftarrow B_1, \ldots, B_n$$

where each B_i $(0 \leq i \leq n)$ is a modal atomic formula.

This transformation is based on the modal resolution given before, and it has allowed us to define a simple procedure for the test T, on the compiled clauses.

5.2 Example

Consider the following three clauses:

(1) knows(a) q
(2) knows(a) $(p \leftarrow q)$
(3) $\leftarrow$ knows(a) p

The compiled clauses will be:

(1) knows(a) q
    q
(2) knows(a) $p \leftarrow$ knows(a) q
    $p \leftarrow q$
(3) $\leftarrow$ knows(a) q

It is easily seen that the compiled program is a classical PROLOG program in which knows(a) p, knows(a) q are considered as atomic formulas.

5.3 Example

We consider again the example in Section 4.3. By compilation we obtain successively:

C[knows(Pierre) knows(Jean) q] $\Rightarrow$
(1) knows(Pierre) knows(Jean) q
(2) knows(Jean) q
(3) knows(Pierre) q
(4) q
C[knows(Pierre) $(p \leftarrow$ knows(Jean) $q)$] $\Rightarrow$
(5) knows(Pierre) $p \leftarrow$ knows(Pierre) knows(Jean) q
(6) $p \leftarrow$ knows(Jean) q

This new set of clauses gives an answer to the goal clause:

(7) $\leftarrow$ knows(Pierre) p

the deduction will be

(8) $\leftarrow$ knows(Pierre) knows(Jean) q using (5)
(9) empty clause using (1)

§6 Deduction Again

Now we will define deduction in the case of compiled modal Horn clauses. For this purpose we must extend the notion of unification to modal atomic formulas.

Definition 17
Given two modal atomic formulas A and B we say that A is **unifiable** to B if there is a substitution σ such that $A\sigma = B\sigma$. We must note that A and B possess a list structure and variables can appear in the modal operators. After this extending classical unification to modal atomic formulas is obvious.

The definition of the **most general unifier** is done in the same way.

Example
Let us give A = knows(x) $p(y)$ and B = knows(Pierre) $p(a)$ then the unification will be x/Pierre and y/a.

Definition 18 Inference rule
Given two compiled clauses $B_0 \leftarrow B_1, ..., B_n$ and $\leftarrow B'_0, B'_1, ..., B'_n$ the inference rule

$$\frac{\begin{array}{l} B_0 \leftarrow B_1, ..., B_n \\ \leftarrow B'_0, B'_1, ..., B'_m \end{array}}{\leftarrow (B_1, ..., B_n, B'_1, ..., B'_m)\sigma} \qquad (4)$$

will be applied, if σ is the most general unifier of B_0 and B'_0.

We will define, as in classical logic, the notion of deduction on the set of compiled modal Horn clauses.

Definition 19
Let S be a set of compiled clauses. A **deduction** is a sequence of clauses C_1, ..., C_n, where C_1 is a compiled goal clause and each C_i is a compiled goal clause obtained from C_{i-1} against a clause in S using the inference rule (4).

This notion of deduction is a simplification of the general resolution method for modal logic. Compilation is a transformation that accomplishes some of the operations given in Section 4.2.

The compilation eliminates the operation Σ corresponding to the modal

operators by transforming the clauses in such a way that only a simple unification operation will be necessary for deduction. The consequence of this approach is that the number of clauses depends on the nested modal operators. The completeness of this new method is obtained easily by transforming any proof of compiled clauses to a proof of the original clauses.

We can represent this argument by the diagram as shown in Fig. 1, where C_1 and C_2 are two given clauses.

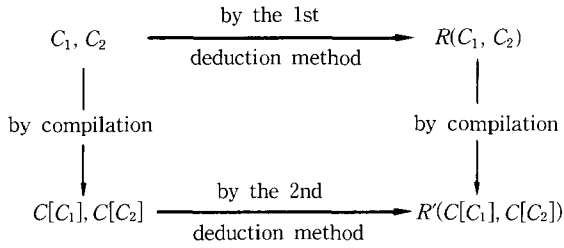

Fig. 1

In Fig. 1, $R(C_1, C_2)$ and $R'(C[C_1], C[C_2])$ represent the resolvent of C_1 and C_2 by the 1st method and the resolvent of $C[C_1]$ and $C[C_2]$ by the 2nd method respectively.

In our proof the main case that must be proved is when we use the inference rule

$$\frac{\begin{array}{l}\text{knows}(a)\ (p \leftarrow q) \\ \leftarrow \Delta p\end{array}}{\leftarrow \Delta q}$$

where Δ is knows(a) or is missing, it can be eliminated by using the compilation transformations:

$$\mathrm{C}[\text{knows}(a)\ (p \leftarrow q)] \Rightarrow \begin{cases} 1. & \text{knows}(a)\ p \leftarrow \text{knows}(a)\ q \\ 2. & p \leftarrow q \end{cases}$$

that corresponds to the two values of Δ, in the scheme

$$\Sigma(\text{knows}(a)\ A;\ \Delta B) = \Delta\Sigma(A; B)$$

where Δ is knows(a) or is missing.

Consequently, by induction on the number of these inferences in the proof, a refutation by resolution can be transformed into a refutation after compilation.

§7 Examples

We will give some examples related to problem solving and manipulation

of control in PROLOG.

We will present a simplified version of a classical example in epistemic logic, the wise man puzzle. Then we will give a formalisation of it, and resolved this version in our system.

Let $S_i(1 \leq i \leq n)$ denote the three wise men and let $p(i)$ be the sentence asserting that S_i has a white spot on the forehead. The simplified version of the wise man puzzle is formalized as follows:

(1) They all know that here is at least one spot which is white.

$$\text{knows}(S_3)\ \text{knows}(S_2)\ \text{knows}(S_1)\ (p(3) \leftarrow \text{not } p(1), \text{not } p(2))$$
$$\text{knows}(S_1)\ \text{knows}(S_3)\ \text{knows}(S_2)\ (p(3) \leftarrow \text{not } p(1), \text{not } p(2))$$
$$\vdots$$

(2) They all know that everybody knows the color of each other's spot.

$$\text{knows}(S_3)\ \text{knows}(S_2)\ \text{knows}(S_1)\ (\text{knows}(S_2)\ p(3) \leftarrow p(3))$$
$$\text{knows}(S_2)\ \text{knows}(S_3)\ \text{knows}(S_1)\ (\text{knows}(S_2)\ p(3) \leftarrow p(3))$$
$$\vdots$$

(3) S_3 knows that S_2 knows that the spot of S_1 is black.

$$\text{knows}(S_3)\ \text{knows}(S_2)\ \text{not } p(1)$$

(4) S_3 knows that the spot of S_2 is black.

$$\text{knows}(S_3)\ \text{not } p(2)$$

Now we compile this set of clauses. Let us consider some of them:

(1) $\text{knows}(S_3)\ p(3) \leftarrow \text{knows}(S_3)\ \text{not } p(1), \text{knows}(S_3)\ \text{not } p(2)$
(2) $\text{knows}(S_1)\ \text{knows}(S_2)\ p(3) \leftarrow \text{knows}(S_1)\ p(3)$
(3) $\text{knows}(S_3)\ \text{not } p(1)$
(4) $\text{knows}(S_2)\ \text{not } p(1)$
(5) $\text{knows}(S_3)\ \text{not } p(2)$

Now we try to resolve the following question:

> Is it possible to deduce from the assumptions that S_3 knows $p(3)$?

In other words we try to prove the following goal clause:

$$\leftarrow \text{knows}(S_3)\ p(3)$$

The deduction will be:

$\leftarrow \text{knows}(S_3)\ \text{not } p(1), \text{knows}(S_3)\ \text{not } p(2)$ (using (1))
$\leftarrow \text{knows}(S_3)\ \text{not } p(2)$ (using (4))
$\leftarrow \text{knows}(S_3)\ \text{not } p(2)$ (using (3))

And by (5) the answer is assured.

We must observe that the proof of knows(S_3) $p(3)$ after compilation is a classical PROLOG program.

Another important problem is to connect the control facilities of PROLOG with modal logic. Broadly we can consider two different ways of doing this.

(1) describing the semantics of PROLOG using temporal logic. This has been done by Warren[15] for the "assume" evaluable predicate,
(2) defining a modal logic and simulating the control of PROLOG in this logic. This is a more difficult task.

Now we try to approach this second problem for the "assume" predicate. The "assume" predicate is like the "assert" predicate with the following restriction: the facts added by the "assume" predicate are removed from the database on backtracking. The "assert" predicate in PROLOG allows us to change the set of clauses by adding new facts. We will give an example and see how it is resolved using the modal operator "knows".

Let us give the following program:

$$t \leftarrow e$$
$$p$$
$$q \leftarrow p,\ \text{assume}(e)\ t$$

then the question: $\leftarrow q$ can be resolved in it.

The translation of the assume predicate in the frame of our modal logic would be:

$$\text{knows}(e)\ e$$

i. e. e is true in every world known by the assumption of e.
The whole program is translated to

$$\text{knows}(x)\ (t \leftarrow e)$$
$$\text{knows}(x)\ p$$
$$\text{knows}(x)\ (q \leftarrow p,\ \text{knows}(e)\ t)$$

The goal clause will be:

$$\leftarrow q$$

It is resolvable in the new set of clauses. This form of introducing the "assumed" operator can be considered as a way of simulating some of the characteristics of the Horn clauses with the nested implications.[10] The clause $C \leftarrow (B \leftarrow A)$ is translated either to $C \leftarrow \text{assume}(A)\ B$, or, in the frame of the logic defined before to $\text{knows}(x)\ (C \leftarrow \text{knows}(A)\ B)$. The first solution is a better solution, because assume(A) is considered as a modal operator, and in this case it will be necessary to define again an axiomatic, a semantic and a set of

resolution rules which define its behaviour.

Another domain of application for modal logic concerns expert systems. We give an elementary example in a multi-agent context, where each agent can possess his own knowledge.

Examples

Let a and b be two agents who both know the following facts and rules:

(1) knows(a) p
(2) knows(a) $(q \leftarrow p)$
(3) knows(b) $(s \leftarrow p)$

The communication between a and b is represented by the clause:

(4) knows(b) $p \leftarrow$ knows(a) p

expressing that the fact p can be exported from the data base knowledge of a to the data base of b. Consequently we can share the general data base to several partial data bases and structure it by a set of clauses that allows us to transfer information from one partial data base to another one. Thus the question knows(b) s can be answered in this example after using successively the clauses (3), (4) and (1).

§8 Conclusion

In this paper we have presented a method of extending PROLOG with modal operators. Two deduction methods have been defined, one that uses modal resolution restricted to modal Horn clauses. The other based on the first is called compilation and allows us to obtain modal Horn clauses closer to classical Horn clauses.

The methodology of our approach has been defined, considering modalities as terms that modify or qualify either facts or rules. Epistemic modal operators have been given as examples of modal operators. Then a semantic in terms of possible worlds has been associated to each modal operator, and a complete set of resolution rules and compilation transformations allowing the deduction in the set of modal operators has been given.

An interesting result, in this approach, would be not to give, as we have done, the resolution rules or the compilation transformations as data for MOLOG, but to obtain automatically the set of resolution rules and compilation transformations for the defined modal operators from their axiomatisation. The general result, i. e. finding resolution rules for every modal operator, is very complex, but for the practical modal operators, it looks to be very promising. It will be explored in future work.

Acknowledgements

We wish to thank R. Arthaud, S. Soulhi and M. Penttonen for their useful comments and suggestions.

References

1) Aiello, L. and Weyhrauch, R. W., "Using Meta-theoric Reasoning to do Algebra," *Proc. of the 5th Automated Deduction Conf.*, Les Arcs, 1980.
2) Boolos, G., *The improvability of inconsistance*, Cambridge Univ. Press, London, 1979.
3) Bowen, K. and Kowalski, R., "Amalgamating language and metalanguage in logic programming," in *Logic Programming* (K. Clark and S.-Å. Tärnlund, eds.), Academic Press, New York, pp. 153-172, 1982.
4) Carnap, R., "Modalities and quantification," *J. S. L.*, Vol. 1 No. 12, 1946.
5) Clark, K. and Greory, S., "Notes on systems programming in PARLOG," *Proc. of the Int. Conf. on Fifth Generation Computer Systems 1984* (ICOT, ed.), Tokyo, pp. 299-306, 1984.
6) Colmerauer, A., "Metamorphosis grammars," in *Natural language communication with computer* (L. Bolc, ed.), Springer-Verlag, 1978.
7) Dincbas, M., "The metalog problem solving system: an informal presentation," *Proc. of Logic Programming Workshop*, Bebrean, pp. 80-91, July, 1980.
8) Fariñas del Cerro, L., "A simple deduction method for modal logic," in *Information Processing Letters, Vol. 14, No. 2*, 1982.
9) Fariñas del Cerro, L., "Resolution Modal logics — Automated reasoning in non-classical logic," *Logique et Analyse* (L. Fariñas del Cerro and E. Orlowska, eds.), Sept., 1985.
10) Gabbay, D. and Reyle, U., "N-Prolog: An extension of Prolog with hypothetical implication," *Imperial College Report*, July, 1984.
11) Konolige, K., "A Deduction Model of Belief and its Logics," *Ph.D. Thesis*, Stanford University, 1984.
12) Kowalski, R., "Predicate logic as programming language," *Proc. of IFIP*, pp. 569-574, 1984.
13) Malachi, Y., Manna, Z. and Waldinger, R., "Tablog: The deductive tableau programming language," *Proc. ACM LISP and Functional Programming Conf.*, 1984.
14) Orlowska, E., "Resolution systems and their applications I & II," *Fundamenta Informaticae, Vol. III, No. 2*, pp. 235-267 & 333-362, 1980.
15) Warren, D. S., "Database update in pure Prolog," *Proc. of the Int. Conf. on Fifth Generation Computer Systems* (ICOT, ed.), Tokyo, p. 244-253, 1984.