

A META-LEVEL APPROACH TO MODAL LOGIC PROGRAMMING

SEIKI AKAMA

*Computational Logic Laboratory, Department of Information Systems,
Teikyo Heisei University, 2289 Uruido, Ichihara-shi,
Chiba, 290-01, Japan.*

Tel: +81-436-74-5511, Fax: +81-436-74-3659
e-mail: SJK15022@mgw.shijokyo.or.jp

ABSTRACT

A meta-level approach to modal logic programming is outlined. The proposed approach simulates a model (or proof) theory for modal logic in Horn clause logic programming using meta-programming techniques. For this purpose, we propose to use a translation of modal logic into a (two-sorted) classical logic (with explicit world parameter). Some methods to improve efficiency which are related to Bowen and Kowalski's amalgamated logic programming are also discussed.

1. Introduction

Modal logic programming is now familiar to researchers in the logic programming community since the development of the first modal logic programming language MOLOG; see Fariñas del Cerro (1986). In fact, MOLOG was designed as a modal extension of Prolog enriched with modal operators, whose inference engine is based on the so-called *modal resolution* originally proposed by Fariñas del Cerro (1985). MOLOG can then provide a powerful programming tool to cope with various problems related to modal notions both in artificial intelligence (AI) and computer science.

From a theoretical point of view, it is interesting to investigate foundations of modal logic programming. Although MOLOG was implemented by means of modal resolution, this approach is not the only one in this context. In fact, there seem to be viable alternatives to realize modal logic programming. One of these alternatives is a meta-level approach whose main objective is to simulate modal logic in a first-order meta-language. More formally, it simulates a model (or proof) theory for modal logic in Horn clause logic programming using meta-programming techniques. The advantage of this approach is a natural link with ordinary logic programming, which can be established in the so-called *amalgamated logic programming* of Bowen and Kowalski (1982). Consequently, the proposed foundation makes it possible to extend to other types of modal logics such as temporal and epistemic logics.

The idea of meta-level representation of modal logic is not really new. In fact, Morgan (1976) first suggested some possible methods for developing a theorem-prover for non-classical logics. He discussed only an outline of his idea and did not show how to implement it in detail. Our proposal, of course, does not fulfil Morgan's idea, but it is still worth exploring the method in the context of modal logic programming. The first attempt at utilizing this sort of system is found in Moore's (1979) meta-level formalization of logic of reasoning about knowledge and action. Moore's system raises a number of issues to improve efficiency and descriptive power, some of which will be mentioned in the subsequent sections.

The plan of this paper is as follows: In section 2, we discuss existing approaches to modal logic programming. Section 3 surveys a meta-logical foundation of modal logic. We introduce a meta-logical foundation of modal logic programming within the framework of meta-logic programming in section 4. Section 5 concludes the paper.

2. Approaches to Modal Logic Programming

The first proposal for modal logic programming is Fariñas del Cerro's (1986) MOLOG based on modal resolution. Around the same time, the present author also gave a similar idea in Akama (1986). Other modal resolution approaches are found in Abadi and Manna's (1987) TEMPLOG and Jiang's (1988) epistemic logic programming. This kind of approach extends the classical resolution method by adding new inference rules for modality. The resulting system constitutes an interpreter for modal logic programming. The merit of such an approach lies in its efficiency as Prolog is very fast in comparison with programming in full first-order logic. But we face a defect of the modal resolution approach: it lacks a flexibility in that different modal systems need different resolution rules. This implies that we may need to rewrite the interpreter in accordance with the modal system in question. This is, in some sense, an overhead for a programming language. An extra task is required in implementing a new modal system. In addition, the resolution method is not almighty. Not every modal system receives a modal resolution method. For example, we do not know any effective resolution system for the modal system G.

An alternative approach to modal logic programming is a meta-logical foundation, which is one subject of this paper. The main claim in this view consists of a meta-logical representation of a proof (or model) theory for modal logic. For example, we can show a meta-theory for a *Kripke semantics* for modal logic in Horn clause logic programming if the *accessibility relation* in Kripke models receives a first-order translation. As mentioned above, however, the Kripke semantics for G is more complicated. This is because the semantics for G is second-order; see Hughes and Cresswell (1984). This makes it more difficult to develop a resolution for G. The same problem also arises in a meta-level approach, but we can avoid it by incorporating an appropriate meta-level control. A meta-level approach is therefore very natural since it can easily be constructed in ordinary logic programming systems.

In fact, the method is viewed as one of the applications of Bowen and Kowalski's *amalgamated logic programming*. The construction is not restricted to the meta-theoretic interpretation for traditional foundations of modal logic. For example, Akama (1990) simulates a *rationalist view* of modal logic within the meta-logical system as a basis for modal logic programming. The present method can be used for *any* theoretical description for modal logic. It is thus very promising. One can point out some difficulties in this foundation, however. The most striking shortcoming is its efficiency. Clearly, this approach is rather indirect. It is thus necessary to indicate a way of improving efficiency in meta-programming techniques. As will be seen from the discussion below, we will be able to debunk a fallacy that the meta-level approach is not efficient. A general discussion on modal logic programming can be found in Gabbay (1987) and Orgun and Ma (1994).

3. A Meta-Level Representation of Modal Logic

Modal logic is an extension of classical logic to deal with intensional concepts with the necessity operator L and the possibility operator M. Intuitively, the formula LA (MA) denotes that A is true in all (some) possible worlds. Since the interpretation of modality is non-truth-functional, its semantical description is more complicated than that of classical logic. It requires a new model to capture the semantics different from the standard classical model. A new kind of semantics was invented by Kripke, who used the notion of *possible world* to accommodate modal logic. The Kripke model is a tuple $\langle W, R, V, D \rangle$, where W is a set of possible worlds, R is a binary relation on W called the accessibility relation, V is an evaluation of formulas, and D is a domain. The pair $\langle W, R \rangle$ is called a *frame*; see Hughes and Cresswell (1984). Modal systems can be characterized by their accessibility relation, i.e. technically the properties of accessibility relation correspond to the set of axioms in the modal system in question.

It is natural to consider a meta-theoretic formulation of Kripke models. This is done by translating a Kripke frame into the classical meta-language. More formally, some modal formula receives first-order counterparts by the desired translation when interpreted on frames. Since these translations get a universal

second-order prefix, the resulting formula may or may not be equivalent to a first-order condition on the frame. This is a meta-level representation of modal logic from the standpoint of Kripke semantics. A similar representation may also be possible if we are concerned with a proof-theoretic perspective. Now, we show the translation T of modal formulas. The basic idea is to regard a modal formula as the classical formula with an additional world parameter. For quantificational formulas, we need the two sorted meta-language. The following are some examples of the desired translation T :

$$(3.1) \quad T(LA \rightarrow A) = \forall y(Rxy \rightarrow Py) \rightarrow Px,$$

$$(3.2) \quad T(LA \rightarrow LLA) = \forall y(Rxy \rightarrow Py) \rightarrow \forall y(Rxy \rightarrow \forall z(Ryz \rightarrow Pz)),$$

where the parameters x, y, z denote the current worlds to be evaluated, P the sets of worlds in which the proposition A holds; see van Benthem (1984). A similar translation was also proposed by Fine (1978). As noted above, this kind of translation is not always first-order. For instance,

$$(3.3) \quad L(A \vee B) \rightarrow (LAVLB)$$

is translated into the next second-order formula:

$$(3.4) \quad \forall P \forall Q (\forall y(Rxy \rightarrow (Py \vee Qy)) \rightarrow (\forall x(Rxy \rightarrow Py) \vee \forall y(Rxy \rightarrow Qy))).$$

Van Benthem's translation is, in fact, simplified by interpreting the above predicate P for representing sets of worlds in which a formula A holds as a two-sorted predicate $A(w)$. Then, an n -place predicate is represented as an $n + 1$ -place predicate augmented with a world parameter. There is no crucial difference between van Benthem's translation and ours, however. In our interpretation, the necessity and possibility operators can be translated into the following formulas:

$$(3.5) \quad T(LA) = \forall y(R(x, y) \rightarrow A(y))$$

$$(3.6) \quad T(MA) = \exists y(R(x, y) \& A(y))$$

The next theorem then states that the translation T preserves validity (provability) in the original modal language.

Theorem 3.7

For any formula A in the modal language L

$$\vdash_L A \text{ iff } \vdash_C TA$$

where the subscript C means two-sorted first-order (or second-order) classical logic.

The proof of this theorem is by a straightforward induction on A . It presents no difficulty. By theorem 3.7, we can show the soundness of our meta-level approach to modal logic programming. There are a number of topics in this research area called *correspondence theory*; see van Benthem (1984).

4. Meta-Logical Foundation of Modal Logic Programming

From a meta-theoretical view of modal logic, modal logic programming can be developed as an extension of Horn clause logic programming. This means that the interpretation of the meta-theory of modal logic in amalgamated logic programming provides a meta-level approach to modal logic programming. In this paper, we attempt to simulate a Kripke semantics for modal logic.

Before detailing the idea, we review Bowen and Kowalski's amalgamated logic programming, which formalizes a meta-theoretic representation of provability in Horn clause logic programming. Bowen and Kowalski introduced a meta-predicate "*demo*" to amalgamate the object-language and the meta-language in logic programming with the so-called *reflection principle* (4.1) linking both levels:

$$P \vdash_L G \text{ iff } Pr \vdash_M \text{demo}(P, G) \quad (4.1)$$

where P denotes a program and G a goal. \vdash_L (\vdash_M) represents object-level (meta-level) provability. Pr is a finite set of sentences for axiomatizing the object-level provability. By the reflection principle, the interchangeability between object- and meta-levels can be established. The Bowen and Kowalski system increases the expressive power of a standard logic programming language. For example, we can describe a meta-interpreter for pure Prolog:

$$\begin{aligned} \text{demo}(P, \text{true}) &\leftarrow \\ \text{demo}(P, G_1 \&G_2) &\leftarrow \text{demo}(P, G_1) \& \text{demo}(P, G_2) \\ \text{demo}(P, G_2) &\leftarrow \text{clause}(G_1, G_2) \& \text{demo}(P, G_1) \end{aligned} \quad (4.2)$$

This example suggests that modal logic can also be simulated in a meta-language. For this purpose, we have to introduce a new parameter for possible worlds. Then, a *world term* should be defined in the suitable way.

The Bowen and Kowalski “demo” predicate is modified so that two-sorted first-order logic can be formalized. A meta-predicate “ $demo(P, A(w))$ ” means that the formula A is provable from a program P at a world w . Since our language is two-sorted, the new predicate R must also be introduced to express the accessibility relation in Kripke models. In this regard, modal operators can be described as follows:

$$\begin{aligned} demo(P, LA(w)) &\leftarrow \neg R(w, v) \vee demo(P, A(v)) \\ demo(P, MA(w)) &\leftarrow R(w, q) \& demo(P, A(q)) \end{aligned}$$

where the variables w, v are assumed to be universally quantified, and q denotes a skolem constant for world parameters. Observe that disjunction \vee can be defined appropriately by using negation as failure \neg in the above definition. If the underlying interpreter supports classical negation, we can completely simulate modal logic in meta-logic programming. In the proposed formulation, there is a need to write down a set of axioms for R in a program. For simulating various modal systems, we simply rewrite a program for the accessibility relation instead of rewriting the whole interpreter.

Now, consider the following example for the modal system Q, which is the system extending the normal system K with the axiom $LA \rightarrow MA$ (where R is a *total* relation).

Example 1

$P = \{ LA \vee B, L\neg A \}$, $G = MB$, $AX = \forall x \exists y R(x, y)$,

where P is a program, R is a goal, and AX is a program for axiomatizing R in the modal system Q.

These programs can be translated into a first-order meta-logic as follows:

$$\begin{aligned} P' &= \{ \forall v (R(w, v) \rightarrow A(v)) \vee B(w), \forall v (R(w, v) \rightarrow \neg A(v)) \}, \\ G' &= \exists v (R(w, v) \& B(v)), \quad AX' = \forall x \exists y R(x, y) \end{aligned}$$

The translated goal G' succeeds from $P' \cup AX'$ in the following way:

- (1) $A(v) \vee B(w) \leftarrow R(w, v)$
- (2) $\neg A(v) \leftarrow R(w, v)$
- (3) $\neg B(v) \leftarrow R(w, v)$ goal ($\neg G$)
- (4) $R(x, f(x)) \leftarrow$ AX
- (5) $B(w) \leftarrow R(w, v)$ (1) and (2)
- (6) $\leftarrow R(w, w)$ (3) and (5)
- (7) *false* (4) and (6)

here, f is a skolem function for world terms. In the above derivation, non-Horn clause could be *split* properly. But, this technique in fact induces the problem of selecting from non-Horn clauses the right atoms to be chosen at every step.

The unification procedure in our system requires a typed version used in many-sorted unification for quantified modal logic. This can be implemented in modal logic programming by revising the standard unification method with a suitable control mechanism. A meta-interpreter for the revised unification is as follows:

$$demo(P, A\theta) \leftarrow demo(P, A) \& demo(P, B) \& unify(A, B, \theta, RES) \quad (4.3)$$

where P is a program and θ is a most general unifier (mgu) of A and B . “ $unify(A, B, \theta, RES)$ ” states that the term A and B are unifiable with mgu θ under the restriction RES . Therefore, we may specify the required condition on the typed unification:

$$\begin{aligned} unify(A, B, \theta, RES) &\leftarrow RES \& sub(\theta, v, s, A, A') \\ &\quad \& sub(\theta, w, s, B, B') \& same(A', B') \end{aligned} \quad (4.4)$$

where “ $sub(\theta, v, s, A, A')$ ” denotes the result A' by replacing all free occurrences of variable v with sort s in the formula A mgu θ . “ $same(A, B)$ ” says that two terms A and B are equivalent in the ordinary sense of unification. For two-sorted unification, the unification procedure is carried out if two terms coincide in sort, thus a variable x of sort $S1$ can only be unified with a term t of sort $S2$ if $S2 = S1$ or $S2$ is a subsort of $S1$. A similar idea also appears in Wallen (1989).

Even if we can formulate a two-sorted unification in our modal logic programming, it might not be satisfactorily efficient. The reason for inefficiency in the above naive method is that this kind of proof

system produces a number of new axioms which are not always necessary. One way of avoiding the trouble is to revise the unification algorithm according to the accessibility relation. This was in fact done by Ohlbach (1988) in his resolution calculus.

Ohlbach also used a two-sorted first-order meta-language by translating a Kripke frame into what he calls P-logic. The proposed translation is essentially the same as ours. The difference is that he revised the unification procedure for particular modal systems, where the *world-path* which is a list of possible worlds plays an important role to encode the information about the accessibility relation. Ohlbach proposed modal unification for various modal systems. For instance, the required unification for T has two transformation rules, i.e. separation and identity:

separation: $\langle s \ u \rangle = \langle t \ v \rangle \implies s = t \ \& \ u = v.$

identity: $\langle s \ w \ u \rangle = t \implies w = \langle \rangle \ \& \ \langle s \ u \rangle = t$

where $w = \langle \rangle$ denotes the removal of a variable completely from a world-path.

We can easily incorporate Ohlbach's idea in our approach to improve efficiency. We must, however, change the syntax of two-sorted first-order meta-logic by allowing world terms to be included in the world-path, which is a list of world terms, in the sense of Ohlbach. A meta-interpreter for the modal system T is written as

$$\begin{aligned} \text{world_path_unify}(p_1, p_2, \lambda) &\leftarrow \text{separate}(p_1, p_2) & (4.5) \\ \text{world_path_unify}(p_1, p_2, \lambda) &\leftarrow \text{identity}(p_1, p_2) \\ \text{separation}(p_1, p_2) &\leftarrow \text{world_path}(p_1, \langle s, u \rangle) \ \& \ \text{world_path}(p_2, \langle t, v \rangle) \\ &\quad \& \ s = t \ \& \ u = v \\ \text{identity}(p_1, p_2) &\leftarrow \text{world_path}(p_1, \langle s, w, u \rangle) \ \& \ \text{world_path}(p_2, t) \ \& \\ &\quad w = \langle \rangle \ \& \ \langle s, u \rangle = t \end{aligned}$$

where λ is an mgu for world terms. To develop meta-interpreters for other modal systems is not difficult. The merit of the representation is that it dispenses with the R predicate for the accessibility relation since the world-path has relevant information. Compare the previous proof for example 1 with the following:

Example 2 (A new proof for example 1 above)

- (1) $A(\langle w, v \rangle) \vee B(w) \leftarrow$
- (2) $\leftarrow A(\langle w, v \rangle)$
- (3) $\leftarrow B(\langle w, f(w) \rangle)$ goal
- (4) $B(w) \leftarrow$ (1) and (2)
- (5) false (3) and (4)

The unification in the final step is trivial since there is at least one world. Notice that the present proof is more compact than the previous naive one in the sense that it needs no additional clauses for the accessibility relation.

The topic is a treatment of quantification. As is well known, the quantification in modal logic is one of the challenging topics in philosophical logic. To interpret quantifiers in the Kripke model defined above, we need the following clauses:

$$\begin{aligned} F, w \models \forall x A(x) &\text{ iff } F, w \models A(t) \text{ for all } t \in D, \\ F, w \models \exists x A(x) &\text{ iff } F, w \models A(t) \text{ for some } t \in D. \end{aligned}$$

The interpretation is standard (cf. Hughes and Cresswell (1968)), in which the domain is constant over all the worlds. There are, however, other interesting interpretations; see Garson (1984). For example, the so-called *variable domain* has a nested domain relative to the accessibility relation. For modeling the domain, we require a domain function P in a Kripke model, which assigns a domain at each world. Then, the evaluation is:

$$\begin{aligned} F, w \models \forall x A(x) &\text{ iff } F, w \models A(t) \text{ for all } t \in P(w), \\ F, w \models \exists x A(x) &\text{ iff } F, w \models A(t) \text{ for some } t \in P(w), \end{aligned}$$

where $D = \bigcup_{w \in W} P(w)$ satisfying the monotonicity condition: $P(w) \subseteq P(v)$ for all worlds v such that wRv .

Now, we show a general method for quantification in modal logic programming so that various interpretations can be realized. The motivation is to expand the classical resolution with a modal version of Herbrand's theorem. For this purpose, we define a *modal prenex normal form*. If a formula contains no

modal operators, then the modal prenex normal form is equivalent to the classical one. If modal operators appear in the formula, then its modal prenex formula has quantifiers outside the scope of the modal operators and its matrix is in conjunctive normal form. The quantifiers in modal prenex normal form can be deleted in the same way as in the classical one. But, the transformation needs to index variables according to the position of modal operators and skolemize existential quantifiers.

Second, we id by a quantifier Q by counting the number of the nested modal operators which have Qx in their scope. An index is also attached to each constant after deleting quantifiers. If we adopt the *rigid terms* (the *denotation* of a term is equivalent in all worlds), indices are necessary only for variables. We here discuss a general case, however. For indexed formulas, the standard unification procedure can be given without violating the restrictions related to variable and constant domains due to the information implicit in indexing.

Example 3

$$\forall x L \forall y (A(x) \ \& \ \exists z B(x, y, z) \rightarrow LC(x, y)) \quad (1)$$

which can be transformed into

$$\forall x \forall y_2 \exists z_1 L(A(x) \ \& \ B(x, y_2, z_1) \rightarrow LC(x, y_2)) \quad (2)$$

Note that the variables not in the scope of modal operators have no index. (2) can be converted into the skolem normal form, i.e.

$$L(A(x) \ \& \ B(x, y_2, f_1(x, y_2)) \rightarrow LC(x, y_2)) \quad (3)$$

Here, f_1 is a skolem function with the index corresponding to the deleted existential quantifier.

Thirdly, we have to revise the classical unification according to particular interpretations of quantification in addition to the above modal unification. The basic assumption of the revised unification is as follows. If one of the unified terms is a constant, then it can be substituted for variables with any index if the constant is assumed to be rigid. Depending on the interpretation of quantification, some terms with different indices could be unified. Motivated by many-sorted unification, we can easily accommodate the proposed index schema. Let the ordering on a set of natural numbers \leq be a sub-sort relation. If $t_1 \leq x_2$ and $t_1 \leq x_3$, then x_2 and x_3 are unifiable since t_1 is their common sub-sort. For instance, we can state the condition on the unification for variable domain systems as: If the term t has an index less than or equal to that of the variable x , then they are unifiable. In contrast, the assumption for constant domain systems is: if the term t has a greater index than that of the variable x , then they are unifiable.

Example 4

$$P = \{ \}, \ G = L \forall x A(x) \rightarrow \forall x L A(x),$$

This is a proof of the converse Barcan formula, which is provable in variable domain systems having no restrictions on R . First, we index the formula in the manner described above:

$$G = \forall x_1 \forall y L(A(x_1) \rightarrow A(y))$$

Next, we transform it into the modal prenex form as:

$$G = L(A(x_1) \rightarrow A(y))$$

In a clausal form:

- (1) $LA(x_1) \leftarrow$
- (2) $\leftarrow L(A(s))$
- (3) $false$

where s is a skolem constant. From the above mentioned restriction, unification is possible.

Example 5

$$P = \{ \}, \ G = \forall x L A(x) \rightarrow L \forall x A(x),$$

This is a proof of the Barcan formula BF, which requires another restrict is:

$$G = \forall x_1 \forall y L(A(y) \rightarrow A(x_1))$$

After the transformation into the modal prenex form, we obtain the following clauses:

- (1) $LA(y) \leftarrow$
- (2) $\leftarrow L(A(s_1))$
- (3) $false$

Since the constant has a greater index than that of the variable, unification is possible. Without the condition, we cannot arrive at the empty clause from (1) and (2). The treatment of quantifiers can be recast in the meta-level system inspired by Ohlbach's resolution calculus. For example, the proof of BF is modified as follows:

Example 6

$P = \{ \}, G = \forall x L A(x) \rightarrow L \forall x A(x),$
 (1) $\leftarrow A(< u >, < x >)$
 (2) $A(< a >, f(< a >)) \leftarrow$
 (3) *false*

If x has no world-path, then the unification succeeds with mgu $\{u \leftarrow a, x \leftarrow f(< a >)\}$. On the other hand, for the variable domain interpretation the unification fails. This is because two world-paths are not unifiable. The role of world-path in Ohlbach's approach is essentially the same as the role of index in our method. Our indexing schema needs no change in the syntax of the first-order meta-language. We believe that the proposed technique can deal also with other interpretations of quantification in modal logic. The meta-interpreter for the indexing schema is immediate. The proposed modal unification can be regarded as the typed unification in many-sorted resolution; see Walther (1987).

5. Conclusion

We have presented a meta-level approach to modal logic programming within the framework of first-order meta-language. Thus, the proposed approach can be simulated in amalgamated logic programming. The meta-level formulation of modal logic can serve as a theoretical foundation of modal logic programming. We have also suggested some techniques for improving the efficiency of translated deduction.

We should observe that our work is related to the work of Balbiani et al. (1991) and Alliot et al. (1992) on *Toulouse Inference Machine* (TIM). TIM is an extension of MOLOG to provide a general framework for non-classical logic programming. TIM can generate a modal logic program from a user's specification of inference rules for modal operators and control strategies. In this sense, their approach can also be classified as a meta-level approach. However, it is not easy to establish the completeness result in their framework because the basis is proof-theoretic without any link with a Kripke semantics. Debart et al. (1992) gives a theoretical foundation for multi-modal logic programming based on algebraic and equational techniques. They also use Ohlbach's method based on world-path arguments with in order-sorted logic. They translate multi-modal logic into equational theories called *path theories* and employ Σ -E-resolution as an inference procedure for multi-modal logic programming. Although their framework translates a Kripke semantics, it needs a non-standard resolution system.

Our theory can be justified by proving the so-called *modal Herbrand property* due to Cialdea and Fariñas del Cerro (1986). This means that our theory can be seen as a meta-level formulation of modal theorem-proving. Although we can prove the modal Herbrand property, we should work out a *declarative* semantics, e.g. fixed-point semantics, for the proposed system as a programming language. We believe that our approach abounds with many applications, e.g. epistemic and temporal reasoning.

6. References

- Abadi, M. and Manna, Z. (1987): Temporal logic programming, *Proc. of IEEE Symposium on Logic Programming*, 4-16.
- Akama, S. (1986): A proposal of modal logic programming, *Proc. of the 6th Canadian Artificial Intelligence Conference*, 99-102, Montreal.
- Akama, S. (1990): The rationalist view of modal logic programming, *Proc. of Fuzzy Logic Programming*, 203-211, Ohio.

- Alliot, J.-M., Herzig, A. and Lima-Marques, M. (1992): Implementing Prolog extensions: a parallel inference machine, *Proc. of the 1992 International Conference on Fifth Generation Computer Systems*, 833-842, ICOT, Tokyo.
- Balbani, P., Herzig, A. and Lima-Marques, M. (1991): TIM: The Toulouse inference machine for non-classical logic programming, *PDK'91: International Workshop on Processing Declarative Knowledge*, 365-382, LNAI 567, Springer-Verlag, Berlin.
- Bowen, K. and Kowalski, R. (1982): Amalgamating language and metalanguage in logic programming, Clark, K. L. and Tärnlund, S.-A. (eds.), *Logic Programming*, 153-172, Academic Press, New York.
- Cialdea, M. and Fariñas del Cerro, L. (1986): A modal Herbrand's property, *Zeitschrift für mathematische Logik und Grundlagen der Mathematik* 32, 523-530.
- Debart, F., Enjalbert, P. and Lescot, M. (1992): Multimodal logic programming using equational and order-sorted logic, *Theoretical Computer Science* 105, 141-166.
- Fariñas del Cerro, L. (1985): Resolution modal logics, *Logique et Analyse* 110-111, 152-172.
- Fariñas del Cerro, L. (1986): MOLOG, a system that extends Prolog with modal logic, *New Generation Computing* 4, 35-50.
- Fine, K. (1978): Model theory for modal logic I, *Journal of Philosophical Logic* 7, 125-156.
- Gabbay, D. (1987): Modal and temporal logic programming, Galton, A. (ed.), *Temporal Logics and Their Applications*, 197-237, Academic Press, New York.
- Garson, J. (1984): Quantification in modal logic, D. Gabbay and F. Guenther (eds.), *Handbook of Philosophical Logic vol. II*, 249-307, Reidel, Dordrecht.
- Hughes, G. E. and Cresswell, M. J. (1968): *A Companion to Modal Logic*, Methuen, London.
- Jiang, Y. J. (1988): Epistemic logic programming, *Proc. of the 1992 International Conference on Fifth Generation Computer Systems*, 515-523, ICOT, Tokyo.
- Moore, R. (1979): *Reasoning about Knowledge and Action*, dissertation, MIT.
- Morgan, C. (1976): Methods for automated theorem proving in nonclassical logics, *IEEE Transactions on Computers* C-25, 852-862.
- Ohlbach, H. (1988): *A Resolution Calculus for Modal Logics*, dissertation, University of Kaiserslautern.
- Orgun, M. and Ma, W. (1994): An overview of temporal and modal logic programming, Macquarie University, Sydney.
- van Benthem, J. (1984): Correspondence theory, D. Gabbay and F. Guenther (eds.), *Handbook of Philosophical Logic vol. II*, 167-247, Reidel, Dordrecht.
- Wallen, L. (1989): *Automated Theorem-Proving in Non-Classical Logics*, MIT Press, Cambridge.
- Walther, C. (1987): *A Many-Sorted Calculus based on Resolution and Paramodulation*, Morgan Kaufmann Publishers, Los Altos.