A Metalogic Programming Approach to Multi-Agent Knowledge and Belief

Robert Kowalski and *Jin-Sang Kim*

Department of Computing
Imperial College of Science, Technology and Medicine
180 Queen's Gate
London SW7 2BZ

Abstract

We investigate, within a logic programming framework, the use of the provability predicate $demo(A, P)$ to represent that an agent named A believes a sentence named P. This use of the provability predicate extends its more usual metaprogramming applications and motivates its use within an amalgamation of object language and metalanguage. We study the application of the amalgamated logic programming language to the wise man problem in detail.

1 Introduction

Metalogic is used in logic programming to implement metaprograms, which manipulate other programs, databases, knowledge bases, or axiomatic theories as data. Typical applications include program transformers, expert system shells, and knowledge assimilators. For many of these applications it is useful to employ a metapredicate $demo(T, P)$, which holds when the sentence named P can be proved (or demonstrated) from the theory named T. For such applications it is usually adequate to implement systems in which the object level theory is separated from the metalevel theory.

The metalevel provability predicate can also be used to represent such modalities as necessity and belief [Kon82], [Per88], [Smu88]. Thus $demo(A, P)$ might represent that the agent named A believes the sentence named P, in

ISBN 0-12-450010-2

the sense that the theory (also named A) constituting the agent's explicitly held beliefs can be used to demonstrate the conclusion named P. For these purposes, an amalgamation [Kow79, BK82] which combines object language and metalanguage in a single language is necessary to obtain the naturalness of expression which is possible in modal logic.

In this paper we investigate how the simple model of logic programming can be extended by the amalgamation of object language and metalanguage to make it more useful for representing knowledge and belief in multi-agent domains. We shall investigate this extension in the context of a representation and solution of the wise man problem, which has been advanced by John McCarthy [MSHI78] as a benchmark for testing the expressive power and naturalness of knowledge representation formalisms. A simple version of this problem is this:

> *A king, wishing to determine which of his three wise men is the wisest, puts a white spot on each of their foreheads, and tells them that at least one of the spots is white. The king arranges the wise men in a circle so that they can see and hear each other (but cannot see their own spot) and asks each wise man in turn what is the colour of his spot. The first two say that they don't know, and the third says that his spot is white.*

The problem is to explain how the third wise man reaches his conclusion.

We will present a solution which uses a logic programming amalgamation of object language and metalanguage. In this respect our solution resembles that of Coscia *et al.* [CFL88], but differs from it in our explicit use of "*demo*" to represent belief. Our solution is similar to modal logic solutions using common knowledge. Following the approach taken by McCarthy *et al.* [MSHI78], we treat common knowledge as the knowledge possessed by any "*fool*" and therefore by any agent. The solution presented in this paper is a simplified and improved version of the one presented in an earlier paper [KK90].

2 A logic programming implementation of the provability predicate for propositional Horn clause logic

The simplest metainterpreter is one for an object language consisting of propositional Horn clause logic:

[MP]	$demo(T, P)$	*if*	$demo(T, (P \text{ if } Q))$
		and	$demo(T, Q)$
[Conj]	$demo(T, (P \text{ and } Q))$	*if*	$demo(T, P)$
		and	$demo(T, Q)$

These two clauses of a metalogic program express that every theory T is closed under *modus ponens* and the rule of conjunction.

In fact the conjunction rule is not strictly necessary because object level rules of the form

$$P \text{ if } Q \text{ and } R$$

can be written instead as

$$(P \text{ if } Q) \text{ if } R.$$

In this case the rule MP can perform the same function as the rule $Conj$. The rule MP is similar to the axiom schema of distribution in modal logic

$$(\Box P \text{ if } \Box(P \text{ if } Q)) \text{ if } \Box Q$$

where "$\Box$" is the necessity operator.

It is usual in metalogic to distinguish between object level formulae, such as P *if* Q, and terms which name object level formulae and which can appear therefore as arguments of metapredicates. However, this distinction has generally been ignored in most practical metaprogramming applications. It seems that this practice of metaprogramming can be justified by the semantics developed by Barry Richards [Ric74], in which sentences name themselves. The context in which a formula syntactically appears distinguishes its use as a term from its use as a formula. We shall use this simplification throughout this paper.

To be used in practice, the rules MP and $Conj$ would need to be augmented by rules describing what axioms belong to what theories. This can be done most simply by treating theories as conjunctions, and conjunctions as lists, writing

$$A_1 \text{ and } A_2 \text{ and } \dots \text{ and } A_n$$

as shorthand for

$$A_1 \text{ and } (A_2 \text{ and } (\dots \text{ and } (A_n \text{ and } true)\dots)).$$

The rules are similar to those defining membership for lists:

$$[Memb1] \quad demo((P \ and \ Q), P)$$
$$[Memb2] \quad demo((P \ and \ Q), R) \quad if \quad demo(Q, R)$$

Using these two clauses together with MP and $Conj$, the metalevel can simulate object level reasoning. For example, the metalevel execution sequence

$$\begin{array}{lll} ? & demo(((p \ if \ q) \ and \ q), p) & \\ ? & demo(((p \ if \ q) \ and \ q), (p \ if \ Q)) \ and & \\ & demo(((p \ if \ q) \ and \ q), Q) & by \ MP \\ ? & demo(((p \ if \ q) \ and \ q), q) & by \ Memb1 \\ ? & demo((q), q) & by \ Memb2 \\ & yes & by \ Memb1 \end{array}$$

using the metalevel clauses pr which define the $demo$ predicate, simulates the object level execution

$$\begin{array}{lll} ? & p & \\ ? & q & by \ p \ if \ q \\ & yes & by \ q \end{array}$$

using the object level theory.

Notice that metalevel reasoning is more powerful than object level reasoning. For example, it can generate answers for such queries as

$$\begin{array}{lll} ? & demo(((p \ if \ q) \ and \ q), X) & \\ & yes & X = p \ if \ q \\ & & X = q \\ & & X = p \end{array}$$

$$\begin{array}{lll} ? & demo(((p \ if \ q) \ and \ X), p) & \\ & yes & X = p \\ & & X = q \\ & & \vdots \end{array}$$

3 Reflection rules

In the case of a propositional Horn clause object language, the metalevel program pr can completely simulate object level provability. This *completeness property* of pr justifies the reflection principle

$$[RP1] \qquad \frac{T \vdash P}{pr \vdash demo(T, P)}$$

Used as a rule of inference, $RP1$ is redundant if *pr* is complete. Nonetheless, its use can improve efficiency by replacing indirect metalevel reasoning by more direct object level reasoning. In cases where *pr* might not be complete, $RP1$ can be used as an inexpensive way to "complete" the procedures for computing *demo*. $RP1$ is the analogue of the rule of necessitation in modal logic:

$$\frac{P}{\Box P}$$

The definition *pr* correctly defines object level provability. This *correctness property* justifies the second reflection principle

$$[RP2] \qquad \frac{pr \vdash demo(T, P)}{T \vdash P}$$

This rule is redundant if *pr* is correct. Nonetheless, it too can be useful in practice, for example by facilitating the use of lemmas L stored by means of metalevel statements of the form

$$demo(T, L).$$

Such metalevel recording of lemmas, both for the sake of efficiency and to facilitate theory revision, is a major concern of truth maintenance systems. $RP2$ is analogous to, but weaker than, the modal necessitation axiom schema

$$P \ if \ \Box P.$$

The two reflection principles were first introduced by Weyhrauch [Wey80] and were a feature of the amalgamation logic developed by Bowen and Kowalski [BK82]. They constitute the sole mechanisms for linking object level and metalevel reasoning in the Reflective Prolog language of Costatini and Lanzarone [CL89]. The first reflection principle, also called attachment, plays an essential role in Konolige's modal deduction model of belief [Kon85], [Kon86].

4 Extensions of the definition of the demo predicate

The provability predicate for any logic defined by means of a finite set of axioms and inference rules can be defined by means of a Horn clause metaprogram. Suppose for example that a logic L has axioms

$$ax_1, ax_2, \ldots, ax_n$$

and inference rules of the form

$$\frac{C_1, C_2, \ldots, C_m}{C}$$

then the proof predicate for L can be defined by means of a metaprogram of the form

$$\begin{array}{lll} demo_L(T, ax_1) & & \\ demo_L(T, ax_2) & & \\ \vdots & & \\ demo_L(T, ax_n) & & \\ demo_L(T, C) & if & demo_L(T, C_1) \\ & and & demo_L(T, C_2) \\ & \ldots & \\ & and & demo_L(T, C_m) \end{array}$$

together with Memb1 and Memb2. Thus provability in any logic can be reduced to provability in Horn clause logic.

For the formalisation and solution of the wise man problem, it is necessary to extend the object level logic to include disjunction and negation. The necessary rules include

$$\begin{array}{llll} [Disj1] & demo(T, (P \ or \ Q)) & if & demo(T, P) \\ [Disj2] & demo(T, (P \ or \ Q)) & if & demo(T, Q) \\ [Disj3] & demo(T, Q) & if & demo(T, (P \ or \ Q)) \\ & & and & demo(T, (not \ P)) \end{array}$$

Although in theory it is not necessary to go outside Horn clause metalogic, in practice it is often useful, as it is for example in the case of defining object level negation as negation by failure:

$$demo(T, not\ P) \ if \ \ not\ demo(T, P).$$

Unfortunately, although this definition is useful for many applications, it is not appropriate when *demo* is interpreted as belief. Given, for example, an assumption that a person is agnostic, it would be possible to conclude using this definition, that that person is an atheist.

What is acceptable, however, both in the case of metainterpreters and in the case of belief, is to hold that an agent believes *not P* if it does not believe P, but has complete information about P, i.e.

$$[Comp1] \quad demo(T, not\ P) \quad if \quad not\ demo(T, P) \quad and \quad complete(T, P)$$

In fact, this rule and its contrapositive

$$[Comp2] \quad demo(T, P) \quad if \quad not\ demo(T, not\ P) \quad and \quad complete(T, P)$$

are both trivial consequences of the more general specification, which, however, goes beyond logic programming form:

$$[Comp] \quad complete(T, P) \quad iff \quad [demo(T, P)\ or\ demo(T, not\ P)]$$

In the solution of the wise man problem, given in section 8, it is necessary to reason that because the first wise man has complete knowledge of the colour of the second wise man's spot (i.e. *white*2) and complete knowledge of the colour of the third wise man's spot (i.e. *white*3), he also has complete knowledge of the disjunction

$$white2\ or\ white3.$$

The necessary reasoning can be carried out by means of the rule

$$[Comp3] \quad complete(T, (P\ or\ Q)) \quad if \quad complete(T, P) \quad and \quad complete(T, Q)$$

This property can be derived from the specification *Comp* together with the rules *Disj*1 and *Disj*2.

The solution of the wise man problem requires the use of *Disj*3 in its contrapositive form

$$not\ demo(T, not\ P) \quad if \quad demo(T, (P\ or\ Q)) \quad and \quad not\ demo(T, Q)$$

Such use of contrapositives, however, is not possible within currently available logic programming systems. Fortunately, all such uses of the contrapositive are used together with the clause *Comp*2, to eliminate the literal

$$not\ demo(T, not\ P).$$

Consequently the effect of using the two clauses can be achieved by using their resolvent instead:

$$[Reason] \quad demo(T, P) \quad if \quad demo(T, P\ or\ Q) \quad and \quad not\ demo(T, Q) \quad and \quad complete(T, P)$$

The term "*Reason*" is borrowed from Aiello *et al.* [ANS88, ANS89].

In the solution of the problem the clause *Reason* is used twice. First it is used to reason that, because the first wise man knows (or believes) that at least one of the spots is white (i.e. *white*1 *or* *white*2 *or white*3), and because he does not know that his own spot is white, and because he has complete information about the colour of the other spots, therefore he knows that at least one of the other spots is white (i.e. *white*2 *or white*3).

It is also used a second time to reason that because the second wise man also knows *white*2 *or white*3, and because he does not know that his own spot is white, and because he has complete knowledge about the colour of the other spot, therefore he knows that the third wise man's spot is white.

5 Belief, knowledge, and confidence

In modal logic, the necessitation axiom schema

$$P \ \mathit{if} \ \Box P$$

is what distinguishes the interpretation of "$\Box$" as knowledge from its interpretation as belief. In metalogic, the analogous axiom schema

$$[Conf] \quad P \quad \mathit{if} \quad demo(T, P) \ \mathit{and} \ \ agent(T)$$

can be interpreted as an expression of confidence in the agent named T. Here the same term T is used both as the name of an agent and as a name of that agent's beliefs.

In the solution of the wise man problem, the confidence schema is used twice. First it is used to derive the conclusion

$$white2 \ \mathit{or} \ white3$$

from the previously derived conclusion (using *Reason*) that the first wise man believes *white*2 *or white*3. It is also used to derive the conclusion

$$white3$$

from the conclusion (also obtained by using *Reason*) that the second wise man believes *white*3. In both cases the confidence schema justifies interpreting the beliefs of the two wise men as knowledge rather than as mere belief.

6 Constants as names of theories

In cases where *demo* means belief, it is not generally possible to identify explicitly all the axioms belonging to the theory which constitutes an agent's beliefs. It is possible, however, to use constants as names of theories instead, and to assert by means of meta-axioms what object level axioms can be proved from the object level theory. Thus, for example, we could use the meta-axioms

[$t1$] $demo(t, p \text{ if } q)$
[$t2$] $demo(t, q)$

to define what axioms belong to the object level theory

$$p \text{ if } q$$
$$q$$

using the constant t as a name of the theory. This use of constants, to name theories and of meta-axioms like $t1$ and $t2$ to perform the same function as $Memb1$ and $Memb2$, is in the spirit of McCarthy's notion of abstract syntax [McC63]. In the same way that $Memb1$ and $Memb2$ belong to the definition *pr* of *demo*, so too should $t1$ and $t2$.

In the wise man puzzle, we will use the constants

$$wise1, wise2, wise3$$

both as names of theories and as names of the wise men themselves. For the sake of simplicity, we assume that the constant $wisei$ names the state of the i-th wise man just before he answers the king's question. In a more precise representation we could represent the state of a theory as a pair (wi, s) consisting of an invariant name wi for the theory and a name s for the state.

Because it is virtually impossible to have complete knowledge of all an agent's beliefs, it is not appropriate to use negation as failure to prove conditions of the form

$$not\ demo(A, P)$$

when A is a constant which names the agent's beliefs. However, conditions of that form occur in the *Reason* axiom. They can be proved by using negative assertions which are given as part of the input. In the wise man puzzle the statements that the first and the second wise men do not know the colours of their spots is just this kind of input:

$$\begin{array}{l} not\ demo(wise1, white1) \\ not\ demo(wise1, not\ white1) \\ not\ demo(wise2, white2) \\ not\ demo(wise2, not\ white2) \end{array}$$

The use of such negative assertions to prove negative conditions requires a trivial extension of the usual logic programming execution strategies. It is implemented in the Prolog solution by introducing a new predicate symbol *not_demo* and by replacing negative literals of the form $not\ demo(T, P)$ by positive literals of the form $not\_demo(T, P)$.

7 Common knowledge

The solution of the wise man puzzle, which is presented formally in section 8, is presented from the third wise man's point of view, i.e. using the theory *wise*3. Perhaps the trickiest part of the solution is to formalise how the third wise man concludes that the second wise man knows *white*2 *or white*3. Does *wise*3 simulate the reasoning of *wise*2? Or does *wise*3 first reason for himself that *white*2 *or white*3, and then conclude somehow or other that *wise*2 must also know *white*2 *or white*3?

Our solution to this problem, like several others ([Kon82], [Kon85]), is based on the concept of *common knowledge*. The third wise man reasons that it is common knowledge that *white*2 *or white*3. Therefore both he and *wise*2 know *white*2 *or white*3. This solution can be viewed as an elegant combination of the two alternative solutions mentioned above.

Following McCarthy *et al.* [MSHI78], we represent common knowledge by means of a common theory *wise*0, which is a subtheory of every agent, i.e.

$$[Common1] \quad demo(T, P) \quad if \quad demo(wise0, P) \quad and \quad agent(T)$$

In the wise man puzzle, the statements made by the king and by each of the wise men are all common knowledge. It is also common knowledge that each wise man has complete information about the colour of every other wise man's spot, and that each wise man is an intelligent agent. Therefore the following "facts" all belong to the theory of every agent:

$(F1)$	$demo(wise0, complete(wise1, white2))$
$(F2)$	$demo(wise0, complete(wise1, white3))$
$(F3)$	$demo(wise0, complete(wise2, white1))$
$(F4)$	$demo(wise0, complete(wise2, white3))$
$(F5)$	$demo(wise0, complete(wise3, white1))$
$(F6)$	$demo(wise0, complete(wise3, white2))$
$(F7)$	$demo(wise0, white1 \text{ or } white2 \text{ or } white3)$
$(F8)$	$demo(wise0, not\ demo(wise1, white1))$
$(F9)$	$demo(wise0, not\ demo(wise1, not\ white1))$
$(F10)$	$demo(wise0, not\ demo(wise2, white2))$
$(F11)$	$demo(wise0, not\ demo(wise2, not\ white2))$
$(F12)$	$demo(wise0, agent(wise1))$
$(F13)$	$demo(wise0, agent(wise2))$
$(F14)$	$demo(wise0, agent(wise3))$

Every common knowledge belief P is held by every agent *wisei* in the form of a metalevel statement $demo(wise0, P)$. Given the additional assumption

$$[Common2] \quad agent(wise0)$$

then by confidence *wisei* can also hold P as an object level belief.

It is a characteristic property of common knowledge, not only that all agents know it, but that all agents know that all agents know it, and that all agents know that all agents know that all agents know it, etc. This property can be captured quite simply by the axiom

$$[Common3] \quad demo(wise0, (demo(T, P) \text{ if } P \text{ and } agent(T)))$$

This axiom is more powerful, and also more useful, than the usual common knowledge axiom:

$$\begin{aligned} demo(wise0, demo(T, P)) \quad & \text{if} \quad demo(wise0, P) \\ & \text{and} \quad demo(wise0, agent(T)) \end{aligned}$$

which can be derived from *Common*3 by one application each of MP and *Conj*.

As we shall see in the proof below, *Common*3 is especially useful when using the attachment rule $RP1$ to construct and reason within an object level representation of the common theory *wise*0. *Common*3 can be used in this context simply to add the axiom schema

$$demo(T, P) \text{ if } P \text{ and } agent(T)$$

to the object level representation. In contrast the usual axiom would need to be applied an infinite number of times to add a clause of the form

Erratum "A metalogic programming approach to multi-agent knowledge and belief" by Kowalski and Kim

[Common 3] on page 241 is too strong. The weaker axiom

demo (wise0, demo (T, P))	if	demo (wise0, P)
	and	demo (wise0, agent (T))

needs to be used instead. This makes the proof on pages 242-243 more complicated. Either the first part of the proof needs to be carried out at the metalevel without using reflection, or else a more complicated form of reflection needs to be employed where the effect of the weaker axiom is obtained by an inference rule

$$\frac{\text{P, agent (T)}}{\text{demo (T, P)}}$$

which is restricted for use within the object level representation of wise0.

To see that the stronger axiom [common 3] is incorrect, consider what follows from the general assumptions

[Conf]	P if demo (T, P) and agent (T)
[Common 3]	demo (wise0, demo (T,P) if P and agent (T))
[Common 2]	agent (wise0)

and the domain-specific knowledge of wise 3

(1) white2
(2) demo (wise0, agent (wise2))

From these wise3 can derive

(3)	demo (T, P) if P and agent (T)	by [Conf, Common 2, Common 3]
(4)	agent (wise2)	by [Conf, (2), Common 2]
(5)	demo (wise2, white2) which is <u>incorrect!</u>	by [(1), (3), (4)]

Clearly the problem lies with the derivation of (3) using [Common 3].

$$demo(T, P)$$

whenever the object level representation contains clauses of the form

$$P\ and\ \ agent(T).$$

In addition to its ability to perform metalevel reasoning, an important characteristic of an intelligent agent is its ability to recognise that every other agent can also perform metalevel reasoning, etc. This property can be captured simply by assuming that the confidence axiom and the axioms *pr* that define provability are all common knowledge. In fact, in the proof below it is necessary to use only the assumptions that *Reason*, *Conf*, *Common*3 and *Comp*3 belong to *wise*0.

8 The proof

The following proof is presented from the point of view of the third wise man, using the axioms *wise*3. These axioms contain the purely object level knowledge

$$white1$$
$$white2$$

but these are not necessary for the proof. Although the proof is presented in a forward direction, it is generated backward in the Prolog implementation.

The proof has two main parts. In the first part, *wise*3 reasons that *white*2 *or white*3 is common knowledge. This reasoning could be performed entirely within the theory *wise*3 by simulating *wise*0. However, it is simpler and potentially more efficient to use the attachment rule to reason directly within an object language representation of *wise*0. The object level representation is constructed by collecting all sentences of the form

$$demo(wise0, P)$$

in *wise*3 and for each such sentence adding

$$P$$

to the object level representation of *wise*0. It is then possible to reason within this representation:

(1)	*white*1 *or white*2 *or white*3	*by F*7
(2)	*agent*(*wise*1)	*by F*12
(3)	*demo*(*wise*1, *white*1 *or white*2 *or white*3)	*by* (1), (2), *Common*3
(4)	*not demo*(*wise*1, *white*1)	*by F*8
(5)	*complete*(*wise*1, *white*2)	*by F*1
(6)	*complete*(*wise*1, *white*3)	*by F*2
(7)	*complete*(*wise*1, *white*2 *or white*3)	*by* (5), (6), *Comp*3
(8)	*demo*(*wise*1, *white*2 *or white*3)	*by* (3), (4), (7), *Reason*
(9)	*white*2 *or white*3	*by* (2), (8), *Conf*

Having proved *white*2 *or white*3 from the object level representation of *wise*0, it is then possible for *wise*3 to reason by attachment that

$$demo(wise0, white2\ or\ white3).$$

This provides the starting point for the second part of the proof, where *wise*3 reasons for himself that *wise*2 knows *white*3, and therefore that *white*3 is true:

(10)	*demo*(*wise*0, *white*2 *or white*3)	*by* (9), *RP*1
(11)	*agent*(*wise*2)	*by F*13, *Common*2, *Conf*
(12)	*demo*(*wise*2, *white*2 *or white*3)	*by* (10), (11), *Common*1
(13)	*not demo*(*wise*2, *white*2)	*by F*10, *Common*2, *Conf*
(14)	*complete*(*wise*2, *white*3)	*by F*4, *Common*2, *Conf*
(15)	*demo*(*wise*2, *white*3)	*by* (12), (13), (14), *Reason*
(16)	*white*3	*by* (11), (15), *Conf*

9 Conclusion

The example of the wise man problem shows that the metareasoning capabilities of logic programming can be extended to formalise reasoning about knowledge and belief. The resulting extension combines the naturalness of modal logic, with the power of metalogic and simplicity and efficiency of logic programming.

The extension of the *demo* predicate to represent knowledge and belief is greatly facilitated by the amalgamation of object language and metalanguage embodied in such axioms as *Common*3 and such axiom schemas as *Conf*. As can be seen, for example, from the work of Coscia *et al.* [CFL88], such amalgamation also has practical value in the implementation of object-oriented systems. A schema such as

$$P\ if\ demo(Ob1, P)$$

within a theory *Ob*2, for example, allows *Ob*2 to inherit all the beliefs of *Ob*1.

In most conventional applications of metaprogramming it is usual for the first argument T of the *demo* predicate to be a term which has the same structure as the theory it names. This has the limitation that the metaprogram then needs to have complete information about the contents of that theory. This is not generally possible when *demo* represents knowledge or belief. In such cases it is possible instead to name theories by constants or other structurally dissimilar terms. It seems likely that such structurally dissimilar terms can be used in a similar way for other more practical applications.

The amalgamation logic used in this paper has an alternative interpretation as a modal logic similar to Konolige's deduction model of belief [Kon86]. It shares with his modal logic the characteristic that the arguments of *demo* denote sentences rather than propositions, as they do in most modal languages. We prefer, however, the interpretation of the amalgamation logic as a metalogic, with the simplifying features of Richards' semantics [Ric74], whereby sentences name themselves.

It is well known from the work of Montague [Mon63] and Thomason [Tho80] that simply treating modal operators as metapredicates can lead to inconsistencies. It has been shown, however, by Perlis [Per88] and by des Rivières and Levesque [dRJL86] that it is possible to extend modal logic or to restrict metalogic so that the two approaches have similar consistency properties.

Another correspondence between modal logic and an amalgamation of object language and metalanguage has been studied by such logicians as Boolos [Boo79] and Smorynski [Smo85]. In their case the amalgamation logic studied is first-order arithmetic, in which Gödel numbers name arithmetic sentences and other syntactic expressions. It was in fact the example of arithmetic which was the original guide for the development of the amalgamation logic of Bowen and Kowalski [BK82] used in this paper. Perhaps it is along these same lines that a more rigorous semantics for the amalgamation logic will be developed in the future.

Acknowledgement

This work was supported in part by the ESPRIT Basic Research Project, COMPULOG, and in part by an ORS(UK) award. We are grateful to our COMPULOG partners, and in particular to Luigia Aiello, Pat Hill, John Lloyd, Paolo Mancarella, Danielle Nardi and Barry Richards for useful discussions about this work.

References

[ANS88] Luigia Aiello, Daniele Nardi and Marco Schaerf. Reasoning about knowledge and ignorance. In *Proc. of the Int'l Conference on FGCS*, pages 618–627, 1988.

[ANS89] Luigia Aiello, Daniele Nardi and Marco Schaerf. Reasoning about knowledge and reasoning in a meta-level architecture. Technical report, Dipartimento di Informatica e Sistemistica, Università di Roma "La Sapienza", 1989.

[BK82] Kenneth Bowen and Robert Kowalski. Amalgamating language and metalanguage in logic programming. In *Logic Programming*, Keith Clark and Stan-Ake Tärnlund (eds.), Academic Press, pages 153–173, 1982.

[Boo79] George Boolos. *The Unprovability of Consistency.* Cambridge University Press, 1979.

[CFL88] Patrizia Coscia, Paola Franceschi, Giorgio Levi, Giuseppe Sardu and Luigia Torre. Object level reflection of inference rules by partial evaluation. In *Meta-Level Architectures and Reflection*, Pattie Maes and Daniele Nardi (eds.), Elsevier Science Publishers, pages 313–327, 1988.

[CL89] Stefania Costantini and Gaetano Lanzarone. A metalogic programming language. In *Logic Programming: Proc. of the Sixth Intl. Conference*, Giorgio Levi and Maurizio Martelli (eds.), pages 218-233. MIT Press 1989.

[KK90] Jin Sang Kim and Robert Kowalski. An application of amalgamated logic to multi-agent belief. In *Proc. of the 2nd Workshop on Meta-programming in Logic*, Maurice Bruynooghe (ed.), pages 272–283, 1990.

[Kon82] Kurt Konolige. A first-order formalization of knowledge and action for a multi-agent planning system. *Machine Intelligence*, vol. 10, Jean Hayes and Donald Michie and Y. Pao (eds), Ellis Horwood, 1982.

[Kon85] Kurt Konolige. Belief and incompleteness. In *Formal Theories of the Commonsense World*, J. Hobbs and Robert Moore (eds.), Ablex Pub. Corp., pages 359–403, 1985.

[Kon86] Kurt Konolige. *A Deduction Model of Belief.* Pitman, London, 1986.

[Kow79] Robert Kowalski. *Logic for Problem Solving.* North-Holland, 1979.

[McC63] John McCarthy. *Towards a mathematical science of computation. Proc. IFIP Congress 1962.* North Holland 1963.

[MSHI78] John McCarthy, Masahiko Sato, Takeshi Hayashi and Shigeru Igrashi. On the model theory of knowledge. Technical report AIM-312, Stanford University, 1978.

[Mon63] Richard Montague. Syntactical treatments of modality, with corollaries on reflection principles and finite axiomatizability. *Acta Philosophica Fennica*, 16, pages 153–167, 1963. Reprinted in *Formal Philosophy*, New Haven, Yale University Press, pages 286–302, 1974.

[Per88] Donald Perlis. Languages with self-reference II: knowledge, belief, and modality. *Artificial Intelligence*, 34, pages 179–212, 1988.

[Ric74] Barry Richards. A point of reference. *Synthese* 28, pages 431–445, 1974.

[dRJL86] des Rivières Jim and Hector Levesque. The consistency of syntactical treatments of knowledge. In *Proc. of the 1986 Conference, Theoretical Aspects of Reasoning about Knowledge*, Joseph Halpern (ed.), pages 115–130, 1986.

[Smo85] Craig Smorynski. *Self-Reference and Modal Logic.* Springer-Verlag, New York, 1985.

[Smu88] Raymond Smullyan. *Forever Undecided.* Oxford University Press, 1988.

[Tho80] Richmond H. Thomason. *A note on syntactic treatments of modality.* Synthese, Vol. 44, pages 391–395.

[Wey80] Richard Weyhrauch, Prolegomena to a theory of mechanized formal reasoning. *Artificial Intelligence*, 13, pages 133–170, 1980.