A Metalogic Programming Approach to Reasoning about Time in Knowledge Bases

S.M.Sripada
European Computer-Industry Research Centre
Arabellastrasse 17, 8000 Munich 81, Germany
email: sripada@ecrc.de

Abstract

The problem of representing and reasoning about two notions of time that are relevant in the context of knowledge bases is addressed. These are called *historical time* and *belief time* respectively. Historical time denotes the time for which information models reality/Belief time denotes the time lor which a belief is held (by an agent or a knowledge base). We formalize an appropriate theory of time using logic as a meta-language. We then present a metalogic program derived from this theory through fold/unfold transformations. The metalogic program enables the temporal reasoning required for knowledge base applications to be carried out efficiently. The metalogic program is directly implementable as a Prolog program and hence the need for a more complex theorem prover is obviated. The approach is applicable for such knowledge base applications as legislation and legal reasoning and in the context of multi-agent reasoning where an agent reasons about the beliefs of another agent.

1 Introduction

Temporal information arises naturally in many practical knowledge base applications. A knowledge base dealing with legislative matters, for example, has to cope with (legal) rules applicable during different periods of time (eg. [Sergot 86]). Given a database of facts concerning the citizens of a nation, the status of a person (eg. whether the person is a taxpayer, whether the person is eligible for a particular kind of benefit etc.) may vary from time to time, depending upon both the person's history as well as the rules that are applicable in inferring their status as illustrated by Example 1.

Example 1 Consider a knowledge base dealing with legislation which has to reason about the following information :

A law came into effect on *1 Jan 88*, that every one earning more than £5000 per annum and who is not a student should pay tax. This information was recorded in a knowledge base on *1 Jan 88* in the form of extended Horn clause: tp(X) ← ic(X,Y), Y >5000, ~s(X)
where, tp(X) denotes that X is a taxpayer, ic(X,Y) denotes that income of X is Y and ~ represents nagation-as-failure.

John started having an income of £6000 per annum from *10 Apr 88*. This information was recorded in the knowledge base on *10 Apr 88*.

John enrolled as a student in a course which starts on *1* Oct 88 and finishes on *27 Sep 89.* This information was recorded in the knowledge base on *1 Oct 88.*

Mary took up a job for an annual pay of £5001 on 2 *Feb 89.* This information was recorded in the knowledge base on 2 *Feb 89.*

John has enrolled as a student again on *1 Oct 90.* This information was recorded in the knowledge base on *17 Oct 90.*

The government changed its tax policies on 1 Nov 91. It passed a legislation that every one who earns more than £5500 per annum should pay tax, irrespective of whether s/he is a student or not. The legislation was passed with retrospective effect from *1 Apr 91.* This information was recorded in the knowledge base on *1 Nov 91.*

Query: John has not paid any tax since April 91. Has he violated the law? Does he need to pay it now? (In other words, find out who has to pay tax and for what time.)

According to the latest state of the knowledge base, say on 2 Nov 91, the history of the world is as shown in Fig la. It may be seen from Fig la that John was a taxpayer during the periods [10 Apr 88,1 Oct 88), [28 Sep 89,1 Oct 90) and [1 Apr 91, o)i. Similarly, Mary was a taxpayer for the period [2 Feb 89, 1 Apr 91).

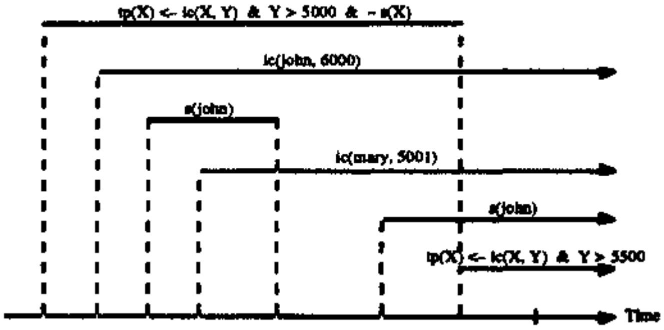

Fig 1a The world as of the KB on 2 Nov 91

However, consider the state of affairs on 20 Oct 91, before the new legislation has been passed. This is represented pictorially in Fig 1b. It may be observed that John was a taxpayer during the periods [10 Apr 88, 1 Oct 88) and [28 Sep 89, 1 Oct 90). Similarly, Mary was a taxpayer during the period [2 Feb 89, ∞).

Therefore, *John was a taxpayer on 15 Apr 91 as of 2 Nov 91* but *John was not a taxpayer on 15 Apr 91 as of 20 Oct 91.*

[1]The symbol ∞ denotes an unkown value or persistence upto infinity in the case of historical time. The same symbol denotes "current time" or *now* in the case of belief time.

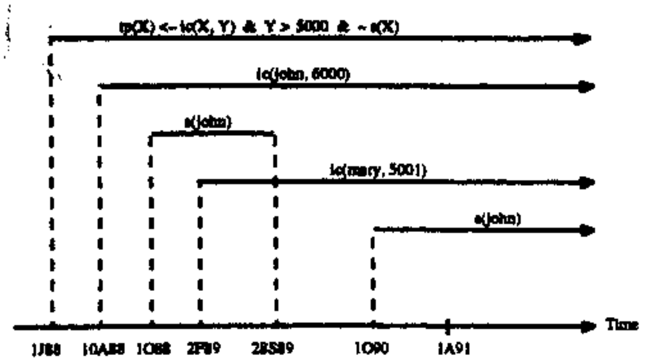

Fig 1b The world as of the KB on 20 Oct 91

Again, Mary was not a taxpayer on 15 Apr 91 as of 2 Nov 91 but Mary was a taxpayer on 15 Apr 91 as of 20 Oct 91. Conversely, Mary was a taxpayer on 15 Apr 91 as of the knowledge base that existed during the period [2 Feb 89 to 1 Nov 91) and Mary was not a taxpayer on 15 Apr 91 as of the knowledge base that existed during the period [1 Nov 91 to ∞).

More generally, the following inferences may be made: Mary was a taxpayer for the period (2 Feb 89, o) as of the knowledge base that existed during the period [2 Feb 89,1 Nov 91); Mary was not a taxpayer for the period [1 Apr 91, ∞) as of the knowledge base that existed during the period [1 Nov 91, ∞).

Answer: John did not violate the law. However, according to the new regulations, John has to pay tax for the period [1 April 91, now). On the other hand, the tax Mary has paid for the period [1 April, now) should be refunded. □

Our goal is to develop an approach to represent and reason with the kind of knowledge described in Example 1. Notice that relationships such as student, income etc. may themselves be derived from other facts and rules as illustrated by the domain rules

student(X) ← enrolled(X,Y), recognised(Y)

colleague(X,Y) ← colleague(X,Y), colleague(Y,Z)

where the relations enrollment, recognition and colleague change in time. Thus domain rules can be fully recursive.

The time at which a piece of information is recorded in the knowledge base, such as, 1 Nov 91 in Example 1, is called belief time. As opposed to this notion of time, the time at which the piece of information models reality (i.e. represents the state of affairs in the real-world being modelled), such as, 1 April 91, is called historical time. As is evident from Example 1, both these times interact with each other. In Example 1, the enactment of the legislation has been made with retrospective effect. It may be noted that this process is identical to overriding some past information with new information (i.e. belief revision for correcting errors etc).

It may also be noted that historical times may be nested. For example, From 1985 to 1989, it has been the case that one would get a visa on the fifth day after an application has been made. Belief time, on the other hand, denotes the time for which a belief is held (by an agent). For example, John believed during the period from 30 March 1990 to 4 August 1991 that Mary was a lecturer during the period from 1 April 1976 to 1 April 1980. As the above examples illustrate, a belief may concern information which itself contains the notion of historical time. Hence historical times may be nested within belief times. Beliefs may themselves involve other beliefs (nested beliefs). Hence belief times may be nested more than once (i.e. an arbitrary number of levels).

In this paper, we address the problem of representing and reasoning with both these notions of time in knowledge bases. We assume that knowledge bases are implemented within the framework of logic programming [Lloyd 87]. The domain knowledge (such as the legislative rules of Example 1) is assumed to be represented in the Horn clause subset of classical logic augmented with negation as failure (as range-restricted rules).

2 Representation of temporal information

We use the metalogic approach for representing temporal knowledge in knowledge bases. We use two meta predicates holds(r, p) and holdsAt(s, t) to represent respectively that the rule or relationship named r is applicable during the period p and the rule or relationship named s is applicable at time point t.

Metalogic has been used in the past for a variety of knowledge representation and reasoning problems such as formalizing "reasoning about reasoning" (eg. [Aiello 88], [Kowalski 92]), for structuring knowledge ([Brogi 91]) etc. In this approach, the fact that John is a taxpayer at time t is expressed by a statement such as holdsAt(taxpayer(john), t). In the context of logic programming, this is a metalevel statement in which taxpayer(john) is a metalevel term denoting the object level formula taxpayer(john). A traditional treatment of the metalevel would require the naming of object level terms and variables by ground/variable-free terms at the metalevel. However, as in the vanilla metainterpreters [Hill 89], we neither use quotation marks nor any other explicit naming devices. Although initially motivated by practical considerations, it has been recently shown that such non-ground representations may be derived from ground representations. The semantic underpinning of this approach may be found in [Kowalski 90; de Schreye 92].

In our approach, the fact that the domain rule

eligible(X)← lonePar(X), income(X,Y), Y<5000

is applicable during the period p1 is recorded in the knowledge base as a metalevel statement

holds(eligible(X)← lonePar(X)&income(X,Y)&Y<5000, p1)

The domain rule eligible(X)← lonePar(X), income(X,Y), Y<5000 is referred to as the object level rule or object rule. Thus an object rule is implicitly represented by a metalevel statement in the knowledge base. Therefore, the knowledge base itself is a meta-knowledge base.

2.1 Syntax and Notation

Domain knowledge is assumed to be in the form of range-restricted Horn clauses extended with negation-as-failure and as simple atomic facts. Predicate and function symbols and constants start in the lower case. Variables start in the upper case. To avoid confusion between the object level and the metalevel, a conjunction in the body of an object rule is represented by &. We use holds(a(b), <10,25>) as shorthand for the three statements hoids(a(b), q), start(q, 10), end(q, 25) where, q is a symbolic name for the time period during which the relationship a(b) holds. Therefore, without loss of generality, all the input to the knowledge base (i.e.the information inside the knowledge base) is assumed to be in the form of metalevel statements such as holds(r, <tl,t2>).

For simplicity, we use numerical values and discrete time in all the examples in the rest of the paper. However, both the theory of time and the technique that we develop in this paper (Sections 3, 4 and 5) are independent of this assumption (eg. the approach is also valid for continuous time), In all our examples we assume that all intervals are closed at the lower end and open at the upper end However, the theory and the technique developed in this paper are neutral with respect to this choice. Lower level predicates such as intersect, union etc should be defined suitably by the users in accordance with their choice of intervals.

The domain information of Example 1 is represented in the knowledge base as shown below. Since the problem involves 2 levels of time, the knowledge base contains meta-metalevel statements:

Name	Meta-metalevel statements in the KB for Example 1
I1	holds(holds(tp(X)←ic(X,Y)&y>5000&~s(X), <1Jan88, ∞>), <1Jan88, 1Nov91>)
I2	holds(holds(tp(X)←ic(X,Y)&y>5000&~s(X), <1Jan88, 1Apr91>), <1Nov91, ∞>)
I3	holds(holds(ic(john,6000), <10Apr88,∞>),<10Apr88, ∞>)
I4	holds(holds(s(john),<1Oct88,28Sep89>),<1Oct88, ∞>)
I5	holds(holds(ic(mary,5001), <2Feb89,∞>),<2Feb89, ∞>)
I6	holds(holds(s(john),<1Oct90,∞>),<17Oct90,∞>)
I7	holds(holds(tp(X)←ic(X,Y)&y>5500, <1Apr91,∞>),<1Nov91, ∞>)

Given such a knowledge base, we now require the ability to derive not only the conclusions that were derivable from the object level rules but also the (historical and belief) time periods for which such conclusions hold.

3 A Theory of Time for Knowledge Bases

We shall adopt both time points and time intervals in our ontology. A discussion of the philosophical issues lying behind this choice may be found in [Galton 90; van Benthem 83]. The following specifications (axioms) expressed in classical logic as a meta-language serve as a theory of time for our purpose. Where variables are not explicitly quantified, they are assumed to be universally quantified in front of the axiom. These specifications are equally valid for historical time as well as belief time.

The holds predicate is defined in terms of holdsAt as follows:

$$\text{holds}(R,P) \leftrightarrow \forall T\,\{T \in P \rightarrow \text{holdsAt}(R,T)\},\ \text{period}(P) \qquad \text{S 1}$$

The predicate ∈ denotes that the time point T belongs to the time period P. A period is defined as follows:

$$\text{period}(P) \leftrightarrow$$
$$\forall T_1 T_2 T_3\,\{T_1 \in P,\ T_3 \in P,\ T_1 < T_2,\ T_2 < T_3 \rightarrow T_2 \in P\} \qquad \text{S 2}$$

S2 captures the convexity property. In other words, there are no gaps in P and therefore P is a contiguous interval.

The conclusion of a rule holds at time T iff the rule is applicable at time T and all the conditions are true at time T:

$$\text{holdsAt}(R, T) \leftrightarrow \text{holdsAt}(R \leftarrow C, T),\ \text{holdsAt}(C, T) \qquad \text{S 3}$$

A conjunction of goals holds at time T iff each of the goals is true at time T:

$$\text{holdsAt}(R1\&R2, T) \leftrightarrow \text{holdsAt}(R1,T),\ \text{holdsAt}(R2,T) \qquad \text{S 4}$$

Note that according to the definition of holds, the period P may not be a maximal time period for which the relationship R holds. We shall now define maximal time periods for which a relationship holds (this notion is needed to take into account the fact that multiple, alternative derivations may exist for the same relationship or property). Intuitively, P is a maximal period during which R holds iff there does not exist an interval containing P during which R holds. The definition of a maximal period is as follows:

$$\text{maxholds}(R,P) \leftrightarrow \text{holds}(R,P),$$
$$\neg \exists P'\,[\text{holds}(R, P'),\ P \subseteq P'] \qquad \text{S 5}$$

The union of two time periods is defined iff they are joint:

$$[\text{joint}(P1,P2) \leftrightarrow \forall T\,\{\min(T1,T2)<T,\ T<\max(T1',T2')$$
$$\rightarrow (T \in P1 \vee T \in P2)\,\}] \leftarrow \text{start}(P1,T1),\ \text{end}(P1,T1'),$$
$$\text{start}(P2,T2),\ \text{end}(P2,T2'),\ \text{period}(P1),\ \text{period}(P2) \qquad \text{S 6}$$

where, min and max are functions denoting the minimum and maximum elements of their arguments respectively. Two time periods intersect iff they have at least one time point in common. The predicate ⊆ denotes the sub-interval relationship between two time periods:

$$[P1 \subseteq P2 \leftrightarrow \forall T\,\{T \in P1 \rightarrow T \in P2\,\}] \leftarrow$$
$$\text{period}(P1)\ \&\ \text{period}(P2) \qquad \text{S 7}$$

We also need to formulate negation in the temporal context:

$$\text{holdsAt}(\sim R, T) \leftrightarrow \neg\text{holdsAt}(R, T) \qquad \text{S 8}$$

In general, a time period has two complements: a complement to the left of the period (chronologically earlier than) and a complement to the right of the period(chronologically later than). We shall use two partial functions *leftcomp* and *rightcomp* to denote these complements respectively, which are defined only when the respective complements exist.

4 The Metalogic Program

It may be observed that the specifications S1-S8 are difficult and sometimes impossible to execute (eg. SI for continuous time). The set of clauses PI-PII can be derived from the specifications (together with some auxiliary predicate definitions) using fold/unfold transformations [Tamaki 84]. The derivations may be found in [Sripada 91]. These clauses have the syntactic form of a logic program and are therefore executable. These clauses manipulate time intervals instead of time points. Hence they are more efficient than the specifications which manipulate time points. Completion semantics and the SLDNF proof procedure form an adequate basis for these logic programs [Lloyd 87].

The time period for which the conclusion of a rule holds is the intersection of the time periods for which the rule holds and the time period for which the conditions of the rule hold:

$$\text{holds}(R,P) \leftarrow \text{holds}(R \leftarrow C,P1),\ \text{holds}(C,P2),$$
$$\text{intersect}(P1,P2,P) \qquad \text{PI}$$

The time period for which a conjunction of goals hold is the intersection of the time periods for which each of the goals hold:

$$\text{holds}(R1\&R2, P) \leftarrow \text{holds}(R1, P1),\ \text{holds}(R2, P2),$$

intersect(P1, P2, P) **P2**

Two overlapping time periods for which a goal holds combine to form a longer period:

holds(R, P) ← holds(R, P1), holds(R, P2),
union(P1, P2, P) **P3**

The program clauses for computing the maximal time period for which a goal holds are as follows:

maxholds(R, P) ← holds(R, P),
NOT ∃ P' [holds(R, P') , P ⊊ P'] **P4**

A property holds in all the subintervals of an interval in which it holds:

holds(R, P) ← holds(R, P'), P ⊊ P' **P5**

holdsAt(R, T) ← holds(R, P), T∈P **P6**

Negation is inferred through the following clauses. The predicate earliest(Goal, P) means that the period P is the earliest time period during which Goal holds.

holds(~R, leftcomp(P)) ← earliest(R, P) **P7**

The predicate latest(Goal, P) means that the period P is the latest time period during which Goal holds.

holds(~R, rightcomp(P)) ← latest(Goal, P) **P8**

The predicate adjacent(Goal, P1 , P2) means that P1 and P2 are two disjoint adjacent time periods during which the goal holds such that goal does not hold at any time point in the gap between these two time periods.

holds(~R, P) ← adjacent(Goal, P1 , P2) ,
intersect(rightcomp(P1) , leftcomp(P2) , P) **P9**

holds(~R, always) ← NOT ∃P' holds(R, P') **P10**

The constant *always* denotes an infinite time period that has no beginning and no end.

Finally, downward reflection [Weyhrauch 80] is also required for each builtin predicate that appears in the knowledge base, such as " > " of Example 1:

holds(X>Y, always) ← X > Y **P11**

Note that any sound and complete proof procedure may be used on the program clauses, including bottom-up evaluation methods developed in the context of deductive databases. The following theorem establishes the soundness and completeness of the metalogic program described above:

Theorem 1 Let KB denote a set of metalevel statements implicitly containing the object level rules and facts. Let S denote the set of specifications S1-S8 and P denote the set of program clauses P1-P11[1]. Then, for any ground atom of the form holds(r, p) the following holds:

$\{ P \cup KB \vdash holds(r, p) \} \leftrightarrow \{ S \cup KB \vdash holds(r, p) \}$ □

5 Reasoning with multiple levels of time

We call the (use of the) logic program P1-P11 the Derivation Approach, since the proof tree for a relationship is derived in the process of deducing the time period for which the relationship holds. It may be observed from Example 2 that the clauses P1-P11 act as a meta-interpreter on the object level rules and facts. The most important feature of the Derivation Approach is that its program clauses are self applicable. Hence the clauses may be used to interpret themselves. Each interpretation layer reasons about an additional level of time attached to the object level theory. In this section, we describe how to reason about one, two or more levels of time associated with object level rules and facts, using the Derivation Approach.

5.1 The Principle

Let DA denote the set of program clauses P1-P11 (augmented with the appropriate lower level auxiliary clauses) of the previous section. Then DA is a logic program.

If P is a logic program then let P<> denote the set of meta-level statements obtained by time stamping each of the rules in P by their corresponding time periods of applicability.

P<> = {holds(r, p) | period(p) & r∈P & r is applicable (resp. believed) during the period p }

Clearly, P<> is also a (meta) logic program.

If DK is an object level logic program (domain knowledge), then { DA ∪ DK<> } is a (meta) logic program from which conclusions of the form holds(ρ, π) may be deduced, where ρ is an atom in the Herbrand Base of DK .

Similarly, { DA ∪ (DA ∪ DK<>)<> } is a (meta-meta) logic program from which conclusions of the form holds(holds(ρ, π), π1) may be deduced.

This process may be repeated as many times as there are levels of time. □

The approach is illustrated by considering Example 1 again.

Example 1 revisited : Two levels of time

DK is the set of object level rules and facts such as

tp(X) ← ic(X,Y) & y>5000 & ~s(X)

DK<> is the set of all beliefs (concerning object rules and the historical time periods for which they are applicable) in the form of metalevel statements such as

holds(tp(X) ← ic(X,Y) & y>5000 & ~s(X), <1Jan88, ∞>)
and holds(s(john), <1Oct88, 28Sep89>).

Therefore, { DA ∪ DK<> }consists of the above metalevel statements together with P1-P11.

To add the next level of time we time stamp each member of { DA ∪ DK<> } with appropriate time periods to get { (DA ∪ DK<>)<> }. For members of DK<>, these are the times for which the information was recorded, and for members of DA, these times are "always" since they are supposed to hold always.

Add another set of program clauses P1-P11 to the above to get {DA ∪ (DA ∪ DK<>)<>}. Therefore, the final program {DA ∪ (DA ∪ DK<>)<>} consists of

- the domain knowledge in the form of I1-I7 (section 2)
- the set of implicit program clauses P1-P11; they are denoted by P1<>, P2<> etc., and
- the set of program clauses P1-P11

For example, the implicit program clause P1<> is represented as

holds(holds(R,P)← holds(R← B,P1)&holds(B, P2)&
intersect(P1, P2, P), always).

Now consider the query

← holds(holds(tp(mary), H), B)

which has two correct answer substitutions:

H = <2Feb89, ∞> , B = <2Feb89, 1Nov91>
H = <2Feb89, 1Apr91> , B = <1Nov91, ∞>

The main steps in the proofs for these answers are given below in the forward direction. A Prolog implementation,

[1]plus other minor axioms and auxiliary program clauses not given in this paper

however, would generate them backwards.

```
(1) holds( holds(~s(mary), always), always)
                                          from P10, P10<>
(2)  holds( holds(5001>5000, always), always)
                                          from P11, P11<>
(3)  holds(holds(ic(mary,5001),<2Feb89,∞>) &
          holds(5001>5000, always),<2Feb89, ∞>)
                                          from (I5),(2),P2
(4)  holds(holds(ic(mary,5001),<2Feb89,∞>) &
          holds(5001>5000, always) &
          holds(~s(mary), always),<2Feb89, ∞>)
                                          from (1), (3), P2
(5)  holds(holds(ic(mary,5001) & 5001>5000 &
               ~s(mary),<2Feb89, ∞>),<2Feb89, ∞>)
                                          from (4), P1<>, P2<>
(6) holds(
     holds(tp(X)←ic(X,Y)&y>5000&~s(X),<1Jan88,∞>)&
  holds(ic(mary,5001)&5001>5000&~s(mary),<2Feb89,∞>),
        <2Feb89, 1Nov91>)            from (I1),(5),P1<>
(7)  holds(holds(tp(mary),<2Feb89,∞>),<2Feb89,1Nov91>)
                                          from (6), P1
```

Therefore, from (7),

H = <2Feb89, ∞> , B = <2Feb89, 1Nov91>

An alternative derivation after (5), involving (I2),(5),P1<>, P2<> gives

H = <2Feb89, 1Apr91> , B = <1Nov91, ∞> □

5.2 Remarks

(1) The process described above may be repeated for an arbitrary number of levels to build complex theories. This is useful in situations like multi-agent reasoning (see Section 5.4) or when knowledge bases reason about the contents of other knowledge bases.

(2) The Derivation Approach can handle full recursion in domain rules. However, termination depends upon the characteristics of the proof procedure used.

(3) Complex interactions are possible between the domain rules. For example, the rule

holdsAt(right_to_vote(X), T) ← holdsAt(tp(X), T)

may be added to the knowledge base of Example 1 where the condition in the body may deduced from rules that change with time. Similarly, it is possible to have a domain rule of the form

holds(visa(X, T) ← apply(X, T1) & T=T1+7, <50, 5000>)

where historical time is nested.

(4) The above programs have been implemented in CProlog on a VAX 11/750 running under Unix. Prolog's evaluation strategy is sound for the occurrences of the existential quantifiers in the program clauses P1-P11.

(5) The overhead of metainterpretation can be completely eliminated through partial evaluation [Sripada 91].

5.3 Deriving time stamps

So far, we have assumed that the time period for which a domain rule or fact is applicable (or believed) is directly input into the knowledge base. However, this assumption is not necessary. The time information may actually be derived from any other theory of time (eg.[Allen 84, McDermott 82, Kowalski 86]) which deals with such temporal reasoning issues as the frame problem and default persistence ([McCarthy 69, Hanks 87]). For example, the historical time (and belief time) for which the object rule tp(X) ic(X,Y), Y >5500 is applicable may be derived from a description of the event of the enactment of the legislation, using the Event Calculus as described in [Sripada 88].

5.4 Application to Multi-agent reasoning

The technique developed in this paper may also be used in a multi-agent scenario for dealing with the temporal aspects of nested beliefs. Consider the following example:

An agent A records a history of his beliefs about the beliefs of another agent B. The beliefs of the agent B, as believed by A, are themselves of a temporal nature, involving both belief time of B and historical time. This may be represented by meta-meta-meta-level facts as:

$\text{holds}^A(\text{bolds}^B(\text{holds(rank(bob, professor), hp)}, bp^B), bp^A)$

The above statement represents that the agent A believes during the period bp^A that the agent B believes during the period bp^D that Bob is a professor during the period hp. In order for the agent A to infer what the agent B believes at any given time, we need to able to represent and reason with multiple belief times such as those described above. The self-applicability of the Derivation Approach plays an important role which makes it useful in the context of Artificial Intelligence in general, and in reasoning about multi-agent interactions, in particular.

We do not claim to provide a full solution to the problem of multiagent beliefs and reasoning. However, the approach presented in this paper may be adapted to formulate different notions of belief and reasoning, as desired, and a logic program may be derived from these specifications. For example, axiom S3 may be viewed as the axiom of belief B(p)AB(p->q) -> B(q). This general principle is the contribution of the present paper. However, a formal exposition of the correspondence between the notions of belief used in this paper and those that are standard in AI literature (eg. [Halpern 85, Hintikka 62]) is beyond the scope of this paper.

6 Related work

Several researchers dealt with temporally scoped rules containing a single level of time in a restricted way (eg. LEGOL [Jones 79] and TMM [Dean 87]). Temporally scoped rules with both recursion and negation-as-failure are handled in an extension of Prolog called Temporal Prolog [Hrycej 88]. However, Temporal Prolog is applicable only for one level of time i.e. historical time. The only other work which we are aware of that deals with multiple belief times is that of Isozaki & Shoham [Isozaki 92]. Their work, developed independently, is primarily concerned with the default persistence of nested beliefs and is restricted to atomic propositional beliefs i.e. there are no implicit beliefs and no deduction is involved. They employ persistence clipping in the presence of contradictions, which is not always the best solution to default persistence, as illustrated by Kautz's stolen car problem [Kautz 86]. Our work is more general since it involves the predicate case and also since it involves deduction. In addition, our approach inherits the persistence and consistency maintenance aspects from another theory of time used for deriving the time stamps. For example, the default persistence mechanism of the Event Calculus, based on negation-as-failure, is known to handle correctly both the Hanks-McDermott problem ([Hanks 87, Evans 89]) as well as Kautz's stolen car problem ([Kautz 86, Shanahan 89]). Hence our technique also benefits from these capabilities.

The theory and technique presented in this paper is complementary to the Event Calculus as well as other theories of time in AI. Our theory and technique are not concerned with the default persistence and temporal reasoning issues that are standard in the AI domain. We are mainly concerned with the two notions of time and their interactions when implicit information is deduced. The AI theories of time are mainly concerned with historical time and default persistence. Thus the two issues are complementary.

7 Conclusions

We have developed a technique for representing and reasoning with the notions of historical time and belief time. The technique may be used in knowledge based applications as well as when reasoning about the temporal aspects of multi-agent beliefs, where nested beliefs are involved. The use of metalogic for formalizing the notions of historical time and belief time in knowledge bases and the derivation of a self-applicable metalogic program that can deal with multiple levels of time illustrates the power and practicality of the metalogic programming approach.

Acknowledgements

I am grateful to Bob Kowalski, Frank van Harmelen, Marek Sergot, Murray Shanahan and Paolo Mancarella for many valuable discussions and comments on this work. Many special thanks are due to Paolo Mancarella for spending hours and hours with me verifying all the derivations and proofs involved. I am very grateful to Beat Wiithrich for extensive comments on an earlier draft which helped reduce the length of the paper to fit in the space provided.

This research has been funded in part by a Jawaharlal Nehru Fellowship (J.N.M.Trust, UK) and in part by the ESPRIT Basic Research Project COMPULOG No. 6810.

References

[Allen 84]Allen, J.F., Towards a general theory of action and time, Artificial Intelligence 23 (1984), pp. 123-154

[Aiello 88] Aiello, L.C. Nardi, D. & Schaerf, M. Reasoning about knowledge and ignorance. In Proc. FGCS 1988

[Bowen]Bowen, K. & Kowalski, R.A. Amalgamating language and and metalanguage in Logic Programming, in Clark and Tarnlund (eds) Logic Programming, Academic Press, pp.153-173,1982

[Brogi 91] Brogi, A. Turini, F. Metalogic for Knowledge Representation, in Proc. KR'91

[Dean 87]Dean, T. L. & McDermott, D. V. Temporal Data Base Management. Artificial Intelligence 32, No.l (1987), pp. 1-55

[de Schreye 92] de Schreye, D. & Martens, B. A sensible least Herbrand model semantics for untyped vanilla meta-programming and its extension to a limited form of amalgamation. To appear in Meta-92, Uppsala, June 1992

[Evans 89] Evans, C. Negation as failure as an approach to the Hanks and McDermott problem. In Proc. 2nd Int. Symposium on Artificial Intelligence, 1989

[Galton 90]Galton, A. A crictical examination of Allen's theory of action and time. Artificial Intelligence 42 (1990) 159-188

[Halpem 85] Halpern, J.Y., Moses, Y. A guide to the Modal Logics of Knowledge and Belief: Preliminary Draft. In Proc. IJCAI 85.

[Hanks 87] Hanks, S. McDermott, D. Nonmonotonic Logics and Temporal Projection. Artificial Intelligence 33 (1987)

[Hill 89] Hill, P.M. & Lloyd, J.W. Analysis of metaprograms. in Abramson & Rogers (eds) Metaprogramming in Logic Programming, MIT Press, pp. 23-52, 1989

[Hintikka 62] Hintikka, J. Knowledge and Belief: An Introduction to the Logic of the two notions. Cornell University Press, 1962

[Hrycej 88]Hrycej, T., Temporal Prolog, in Proc. European Conference on Artificial Intelligence, 1988, pp.296-301

[Isozaki 92] Isozaki, H., Shoham, Y. A Mechanism for Reasoning about Time and Belief. In Proc. FGCS 92

[Jones 79]Jones, S., Mason, P., Stamper, R., LEGOL 2.0: A Relational Specification Language for Complex Rules, Information Systems, Vol. 4, 1979, pp. 293-305

[Kautz 86] Kautz, H. The logic of persistence. In Proc AAAI86

[Kowalski 86]Kowalski, R.A., Sergot, M.J. A Logic-based Calculus of Events. New Generation Computing, 4, No. 1 (1986), pp. 67-95

[Kowalski 90]Kowalski, R.A. Problems and Promises of Computational Logic. In J.W.Lloyd (ed.) Proc. ESPRIT Symposium in Computational Logic, Brussels, November 1990, Springer Verlag, 1990

[Kowalski 92]Kowalski, R.A. & Kim, J.S. A metalogic programming approach to multi-agent knowledge and belief. Department of Computing, Imperial College, London.

[Lloyd 87]Lloyd, J.W. Foundations of Logic Programming (Second Edition) Springer Verlag, 1987

[McCarthy 69]McCarthy, J.M., and Hayes, P.J., Some philosophical problems from the standpoint of artificial intelligence, in Machine Intelligence, D.Michie (Ed.), 1969

[McDermott 82]McDermott, D. A temporal logic for reasoning about processes and plans. Cognitive Science, Vol. 6,1982, 101-155

[Sergot 86]Sergot, M.J. Sadri, F. Kowalski, R.A. Kriwaczek, F. Hammond, P. & Cory, H.T. The British Nationality Act as a Logic Program. CACM, Vol. 29, No. 5, May 1986, pp. 370-386

[Shanahan 89] Shanahan, M.P. Prediction is deduction but explanation is abduction. In Proc IJCAI 1989

[Sripada 88] Sripada, S.M. A logical framework for Temporal Deductive Databases, in Proc. VLDB'88,1988

[Sripada 91]Sripada, S.M. Temporal Reasoning in Deductive Databases, Ph.D. Thesis, Dept. of Computing, Imperial College, London, 1991

[Tamaki 84]Tamaki, H. & Sato, T. Unfold/Fold Transformation of Logic Programs, in Proceedings Second International Conference on Logic Programming, Uppsala, 1984, pp. 127-138

[van Benthem 83]van Benthem, J.F.A.K. The Logic of Time, Reidel Publishing Company, Holland, 1983

[Weyhrauch 80] Weyhrauch, R. Prologomena to a theory of mechanized reasoning. Artificial Intelligence, 13, 1980, pp.133-17