Reasoning about the safety of information : from logical formalization to operational definition

Laurence Cholvy and Robert Demolombe
ONERA/CERT, Toulouse, France

Andrew Jones
Department of Philosophy and
Norwegian Research Centre for Computers and Law
University of Oslo, Norway *

Abstract

We assume that safety of information stored in a database depends on the reliability of the agents who have performed the insertions in the database. We present a logic S to represent information safety, and to derive answers to standard queries and to safety queries. The design of this logic is based on signaling act theory. Two strong simplifications lead to a logic S" with two modalities to represent explicit beliefs and implicit beliefs. Then, we present an operational view of S" in terms of First Order Logic, with meta predicates, which is implemented by a Prolog meta program. It is proved that answers derived in S" and computed by the meta program are identical. This property gives a clear meaning to computed answers.

Content areas: Epistemological foundations, Theorem proving, Logic programming, Multi-agent systems.

1 Introduction

The content of a database is usually considered as a set of database beliefs, and the only part which is recognized to represent true beliefs is the set of integrity

[0]e-mail addresses, L.Cholvy : cholvy@tls-cs.cert.fr, R.Demolombe: demolomb@tls-cs.cert.fr , A. Jones: jones@hedda.uio.no.

constraints [10, 3]. However there are many situations where we can have a more refined information about database safety. Indeed, we may know for some particular sentences that they represent true beliefs if they are inserted by reliable agents. In that case the derivation of answers to standard queries may involve both beliefs and true beliefs, and it is not easy to know what is the status of the answer. In this context safety queries are defined to determine the part of the standard answers that represent true beliefs. Such a situation is illustrated by the following small example.

Let a, b, and c be three agents each capable of inserting information in a database. Agent a is an expert in economics and he knows some empirical rules which hold during particular periods of time. In particular a is an expert judge of the respective circumstances (not formally specifiable) under which the following two rules hold: "If taxes increase then unemployment increases", and, "If taxes increase then unemployment does not increase". If the propositions "taxes increase" and "unemployment increases" are respectively denoted by A and B, then the formal representation of these two rules is: $A \rightarrow B$ and $A \rightarrow \neg B$.

The fact that the system which manages the database knows that agent a is an expert for these two rules means that if, at a given time, a asserts $A \rightarrow B$ (resp. $A \rightarrow \neg B$), that is, if he inserts the rule $A \rightarrow B$ (resp. $A \rightarrow \neg B$) in the database, then $A \rightarrow B$ (resp. $A \rightarrow \neg B$) is guaranteed to be true. This fact is denoted by $Safe(a, A \rightarrow B)$ (resp. $Safe(a, A \rightarrow \neg B)$). It is important to notice that the system knows $Safe(a, A \rightarrow B)$ and $Safe(a, A \rightarrow \neg B)$ independently of the fact that agent a has inserted, or not, one of the two rules $A \rightarrow B$ or $A \rightarrow \neg B$.

Agent b is an expert in sociology and he knows that under some circumstances the following rule holds: "If unemployement increases and social rights are restricted, then criminality increases". If propositions "social rights are restricted" and "criminality increases" are respectively denoted by C and D, the rule is represented by: $B \wedge C \rightarrow D$, and reliability of b in regard to this rule is represented by: $Safe(b, B \wedge C \rightarrow D)$.

Finally agent c is a representative of the governement and he knows whether taxes increase, or not, and he knows if social rights are restricted, or not. So, we have: Safe(c,A), Safe(c,$\neg$A), Safe(c,C) and Safe(c,$\neg$C).

Let's consider now a situation where the database content is the result of the following insertions:

a has inserted:	$A \rightarrow B$ and $B \rightarrow D$
b has inserted:	$B \wedge C \rightarrow D$ and C
c has inserted:	A

and the information about agent reliability is represented by:

Safe(a, $A \rightarrow B$), Safe(a, $A \rightarrow \neg B$), Safe(b, $B \wedge C \rightarrow D$), Safe(c,A), Safe(c,$\neg$A), Safe(c,C) and Safe(c,$\neg$C).

First it may be observed that fact C is not guaranteed to be true since it was inserted by agent b who is not known to be safe regarding C ; and $B \rightarrow D$ is not guaranteed to be true since it was inserted by the agent a, who is not known to be an expert in sociology.

Now, if we put to the database the query: "is it true that unemployment increases?", formally represented by: B?, the answer is "yes", since the database contains A and $A \rightarrow B$, and both of them are guaranteed to be true, then B is true. However if we put the query: "is it true that criminality increases?", formally represented by: D?, then we note that there are just two ways to derive D from the database. The first one is represented by the derivation: A,$A \rightarrow$ B,B,$\mathbf{B \rightarrow D}$,D and the second one is: A,$A \rightarrow$ B,B,C,$B \wedge C \rightarrow$ D,D. The first derivation contains $B \rightarrow D$ which is not guaranteed to be true, and the second one contains C which is also not guaranteed to be true. So, D is not guaranteed to be true, and must therefore be considered to be merely a belief of the database.

This little toy example exhibits some interesting features of reasoning about the safety of information. The first is that one and the same sentence may be assigned one or more epistemic statuses, which need to be made explicit in a logical formalization. For instance, the rule $A \rightarrow B$ may represent a database belief, or it may represent information guaranteed to be true, or it may represent a piece of information regarding which some agent is known to be reliable. A second important feature is that the system which manages the database has to keep trace of the agents who have performed the insertions, because information safety depends on the reliability of these agents. In particular - returning to our example - even though the rule $B \wedge C \rightarrow D$ is logically redundant with respect to $B \rightarrow D$ (assuming that "$\rightarrow$" validates strengthening of the antecedent), we should not remove it, since it is guaranteed to be true, whilst $B \rightarrow D$ is merely a database belief.

This example also shows that even for a small-scale case it is not be easy to

draw valid conclusions from the description of the given situation ; and we can easily imagine that in real examples, with stores of rules, and thousands of facts we need a well defined logic to guarantee the validity of conclusions.

We have developped for this purpose a logical framework in [4] which is recalled in the next section. Some possible options have been selected, whose results lead to the definition of the S logic where insertions are represented by a particular action operator. In section 2.2 a simplified version of S, the S' logic, ignores insertion actions, and only represents the effect of these actions, that is the inserted sentences and the agents who have performed the insertions. A second simplification, presented in section 2.3, is to only consider two modalities: EB for the representation of database explicit beliefs and B for database implicit beliefs. That leads to a third logic S". The axiomatics of the logic S, S' and S" are presented. We present the semantics of S", and it is proved that S" is valid and complete.

Then we present in section 3 an operational view of S" in terms of First Order Logic, where meta predicates are intended to represent the same information as modalities EB and B. A corresponding Prolog meta program is given. The small example we have presented in the introduction is represented both in S" and in the Prolog program. It is also proved that answers derived in S" and computed by the meta program are identical.

The main contribution of the paper is to show the different steps from a general logical formalization of information safety to an implemented Prolog program whose results have a clear meaning defined in a modal logical framework. Moreover this program is a resulting tradeoff between the generality of a powerful logical framework that was defined to help to understand complex phenomena, and the efficiency of an implementation that can manage non trivial applications.

2 Axiomatic definition of the logic

The concept of Safety is deeply related to the reliability of the sources of information. For this reason we have adopted the general framework of signalling acts [7], in order to have an explicit representation of the different types of agents involved in the process of information storage and retrieval. In our approach the DB is considered as a means for communication. Some agents, called "information sources", which may be users or sensors, bring it about that messages are stored in DB. These messages can be read by another agent, namely the DB management

system, called the "system" for short, who knows the meaning of every stored message. The system is able to derive consequences of the information read in DB in order to compute answers to standard queries. There is another agent, called DB administrator, who plays a special role in the communication process. He is the agent who has information about source reliability. This meta-information, called Safety information, is stored by the administrator in a meta-database MDB, and it can be read by the system to compute answers to Safety queries.

A general formalization is presented that can also be used to investigate other issues than computing answers to Safety queries, such as database updates, or other issues related to database security.

Action operator for signalling acts.

An action modality E is introduced to represent acts of transmitting messages to the DB. Wffs of the form $E_a p$ will be read "the agent a brings it about that p", where p is a sentence about messages that are stored in DB. In the case where p is an atomic formula, it is of the form : in.DB(m), and its intended meaning is "the message m is stored in DB". E is closed under logical equivalence:

$$(\mathrm{RE}) \quad \frac{p_1 \leftrightarrow p_2}{E_a p_1 \leftrightarrow E_a p_2}$$

and it will be a "success" operator, in the sense that all instances of the schema (ET) are assumed true :

$$(\mathrm{ET}) \quad E_a p \rightarrow p$$

Since we take tautologies to be outside the scope of anyone's agency, we accept the schema (E¬N) [9]:

$$(\mathrm{E}\neg\mathrm{N}) \quad \neg E_a \top \qquad \text{(where } \top \text{ is any tautology)}$$

Since we accept (RE) and (R¬N) we must reject the distibution principle: (Em) $E_a(p_1 \wedge p_2) \rightarrow (E_a p_1 \wedge E_a p_2)$. A case can be made (cf. Elgesem, [5] for discussion of this), for accepting the converse principle: (Ec) $(E_a p_1 \wedge E_a p_2) \rightarrow E_a(p_1 \wedge p_2)$.

Interpretation of signalling acts by the system.

We must find a way of representing how the result of each signalling act (the messages stored in DB) is interpreted by the system.

In a general approach we can consider that messages have no structure, and that they might be just strings of characters, or strings of bits. In that case the meaning of each message m_i is represented by an axiom (S_i) of the form :

$$(S_i) \quad K_s(E_a p_i \rightarrow B_a q_i)$$

If p_i is the sentence : in.DB(m_i), axiom (S_i) can be rephrased as "the system knows that, if the agent a brings it about that the message m_i is stored in DB, then a believes q_i". Formulas of the form $E_a p_i \rightarrow B_a q_i$ specify, in their consequents, the meanings assigned to signalling acts by the agent a.

In most of the existing databases the autonaming convention is adopted. That is, sentences are represented by strings of characters that are the sentences themselves. In this situation, if 'q' is a message the system interprets as meaning that q, then the meaning of messages may be represented by the unique axiom schema (S) instead of a set of axioms (S_i) :

$$(S) \quad K_s(E_a(\text{in.DB}('q')) \rightarrow B_a q)$$

For simplicity the core modality K will be defined by : $K_a q \stackrel{\text{def}}{=} (B_a q) \wedge q$, and the modality B will be assigned a logic of type KD45 [2].

We have assumed that the system is observing the DB, and that it is informed about every signalling act performed by information-transmitting sources. In other words, the system knows the DB content, and who stored the messages in DB. This assumption is formally represented by the axiom schema :

$$(OBS) \quad E_a p \rightarrow K_s E_a p$$

It may also be necessary for the system to collate the beliefs of the various agents who have inserted messages into DB, and to draw conclusions from this collection of beliefs. Where a is any agent, we thus introduce :

$$(\text{BEL}) \quad K_s B_a q \rightarrow K_s B q$$

where "$K_s Bq$" is read "the system knows that it is believed that q".

Concept of Safety.

Now the concept of Safety is defined by :

$$\text{Safe}(a, p, q) \stackrel{\text{def}}{=} K_{adm}(E_a p \rightarrow q)$$

where "q" is the meaning of the signalling act "$E_a p$".

Sentences of the form Safe(a,p,q) can be read "the agent a is reliable when he performs an action whose result is p, and whose meaning is q". The formal definition of Safe(a,p,q) means that the administrator, called "adm", knows that if the agent a brings it about that p then q is true.

If the autonaming convention is accepted the definition of Safe takes the form :

$$\text{Safe}(a, q) \stackrel{\text{def}}{=} K_{adm}(E_a(\text{in.DB}('q')) \rightarrow q)$$

The axiom schema (SAF) says that all the safety information is known by the system, or, in other words, that the system knows the content of the meta-database MDB :

$$(\text{SAF}) \quad K_{adm}(E_a p \rightarrow q) \rightarrow K_s(E_a p \rightarrow q)$$

2.1 Axiomatics of the S logic

The logic we have adopted will be called the S logic. In summary, it is formally defined by the following axiom schemas and inference rules:

- All the axiom schemas of Classical Propositional Calculus.

- For any modality of the form B_a, and for B, we have the axiom schemas of KD45, and any modality of the form K_a is defined as $K_a q \stackrel{def}{=} (B_a q) \wedge q$.
- For the modality E we have:
  - (RE) $\frac{p_1 \leftrightarrow p_2}{E_a p_1 \leftrightarrow E_a p_2}$
  - (ET) $E_a p \rightarrow p$
  - (E¬N) $\neg E_a \top$
- The links between the modalities E_a, B_a, K_s and K_{adm} are defined by:
  - (OBS) $E_a p \rightarrow K_s E_a p$
  - (S) $K_s(E_a(in.DB('q')) \rightarrow B_a q)$
  - (BEL) $K_s B_a q \rightarrow K_s B q$
  - (SAF) $K_{adm}(E_a p \rightarrow q) \rightarrow K_s(E_a p \rightarrow q)$
- The inference rules Modus Ponens and (for all modalities except E_a) Necessitation.

The semantics of the S logic is not presented here, but the semantics of each modal operator is well defined in terms of Kripke models or minimal modal models. For more details see [2].

Assumptions about agent safety are represented by sentences of the form Safe(a,q), which are defined by:

$$Safe(a, q) \stackrel{def}{=} K_{adm}(E_a(in.DB('q')) \rightarrow q)$$

A given state of the database is represented by the conjunction of a set of assumptions st defining safety of agents, and the insertion actions that have been performed. Then st is the conjunction of a set of sentences of the form:

$st = \{ Safe(a, q_1),\ Safe(a, q_2),\ \ldots, Safe(b, q'_1),\ Safe(b, q'_2), \ldots$
$E_a(In.DB('q'_i)),\ E_a(In.DB('q'_j)), \ldots, E_b(In.DB('q'_k)),\ E_b(In.DB('q'_l)),\ \ldots \}$

For a query represented by the sentence q, the answer to the standard query is "yes" iff we have: $\vdash_S st \rightarrow K_s B q$, and the answer to the safety query is "yes" iff we have: $\vdash_S st \rightarrow K_s q$.

2.2 Axiomatics of the S' logic

We shall now define a simplified version of the S logic, called S', which will be an intermediate step toward the logic S" ; the latter will be powerful enough to represent the phenomena we want to consider in the context of Data and Knowledge Bases in this paper.

The first simplification in the definition of the S' logic is to consider only the result of insertion actions performed by the agents. So, the action operators of the form E_a are omitted, and sentences of the form: $E_a(In.DB('q'))$ are replaced by sentences of the form EB_aq, where EB_a is a new modality. EB_aq can be read "the database explicitly believes q, and a sentence whose meaning is q has been inserted by the agent a". A consequence of this interpretation is that formulas in the scope of the EB_a modality are restricted to propositional formulas. To reflect the fact that in EB_aq we do not consider the syntactical form of q, but only its meaning, we have to accept the following inference rule:

$$(RE') \quad \frac{q_1 \leftrightarrow q_2}{EB_aq_1 \leftrightarrow EB_aq_2}$$

However the modality EB_aq obeys none of the axiom schemas of KD45. The consequence of this first simplification is that (RE), (ET), (E¬N), (OBS), (S) and (SAF) must be removed, since each involves the operator E_a.

The second simplification is to drop the distinction between the agents called "administrator" and "system", replacing the modalities K_s and K_{adm} by the one modality K_s.

To summarize the formal definition of the S' logic is:

- All the axiom schemas of Classical Propositional Calculus.
- For the modalities B and B_s we have the axiom schemas of KD45, and the modality K_s is defined as $K_sq \stackrel{def}{=} (B_sq) \wedge q$.
- For each modality of the form EB_a we have:
  $(RE') \quad \frac{q_1 \leftrightarrow q_2}{EB_aq_1 \leftrightarrow EB_aq_2}$

- The link between the modalities EB_a, B and K_s is defined by:
  (BEL') $K_s(EB_a q) \rightarrow K_s(Bq)$
- The inference rules Modus Ponens and (for all modalities except EB_a) Necessitation.

Assumptions about the reliability of agents are now represented by sentences of the form Safe'(a,q) which are defined by:

$$\text{Safe}'(a, q) \stackrel{\text{def}}{=} K_s(EB_a q \rightarrow q)$$

A given state of the database is now defined by the conjunction of a set st' of sentences of the form:

st' = { $\text{Safe}'(a, q_1)$, $\text{Safe}'(a, q_2)$, ..., $\text{Safe}'(b, q'_1)$, $\text{Safe}'(b, q'_2)$, ...
$K_s(EB_a q_i)$, $K_s(EB_a q_j)$, ..., $K_s(EB_b q_k)$, $K_s(EB_b q_l)$, ... }

For a query represented by the sentence q the standard answer is "yes" iff we have: $\vdash_{S'} st' \rightarrow K_s Bq$, and the answer to the safety query is "yes" iff we have: $\vdash_{S'} st' \rightarrow K_s q$.

2.3 Axiomatics of the S" logic

It is noticeable that in the S' logic derivation of formulas of the form $K_s Bq$ and $K_s q$ from st' only involves formulas that are prefixed by the modality K_s. That means that these derivations reflect derivations performed by the agent called the "system". So, the fact that we are in the context of the "system" knowledge could be understood to be implicit in these derivations, and that leads to a further simplification where the modality K_s is abandoned. Then the formal definition we get for the S" logic is:

- All the axiom schemas of Classical Propositional Calculus.
- For the modality B we have the axiom schemas of KD45.
- For each modality of the form EB_a we have:
  (RE') $\dfrac{q_1 \leftrightarrow q_2}{EB_a q_1 \leftrightarrow EB_a q_2}$

- The link between the modalities EB_a and B is defined by:
  (BEL") $EB_aq \rightarrow Bq$
- The inference rules Modus Ponens and Necessitation for the modality B.

Assumptions about the reliability of agent are represented in the S" logic by sentences of the form Safe"(a,q) which are defined by:

$$\text{Safe}''(a, q) \stackrel{\text{def}}{=} EB_aq \rightarrow q$$

It might be noticed that from EB_aq and (BEL") we can infer Bq, and from the axiom schema (D) we have $\neg B(\neg q)$, and from the contrapositive form of (BEL") for $\neg q$, we have $\neg EB_a(\neg q)$. Then, due to properties of material implication we have Safe"(a,$\neg$q). It is quite counter-intuitive to be able to infer that agent a is reliable in regard to $\neg q$ from the fact that a has inserted q in the database, and this shows that the definition of Safe" is not a correct formal representation of the concept of safety.

However, from an operational point of view, this fact has no dramatic consequences, in as much as we are not interested in deriving from st" conclusions regarding agent reliability. Indeed, if we only consider the derivation of answers to queries, to infer $\neg q$ from Safe"(a,$\neg$q), we have to have $EB_a(\neg q)$, which together with EB_aq leads to an inconsistency.

We do not have this problem with the definition of Safe' because to have Safe'(a,$\neg$q), $\neg EB_a(\neg q)$ has to be true in every world where the knowledge of the system S' is true, and not just in one particular world.

A given state of the database is defined by the conjunction of a set st" of sentences of the form:

st" = { $\text{Safe}''(a, q_1)$, $\text{Safe}''(a, q_2)$, $\ldots, \text{Safe}''(b, q_1')$, $\text{Safe}''(b, q_2'), \ldots$
EB_aq_i, EB_aq_j, $\ldots, EB_bq_k$, $EB_bq_l, \ldots$ }

For a query represented by the sentence q the standard answer is "yes" iff we have: $\vdash_{S''} st'' \rightarrow Bq$, and the answer to the safety query is "yes" iff we have: $\vdash_{S''} st'' \rightarrow q$.

For the example presented in the introduction st" is:

$EB_a(A \rightarrow B)$, $EB_a(B \rightarrow D)$, $EB_b(B \wedge C \rightarrow D)$, $EB_b(C)$, $EB_c(A)$,

Safe(a, $A \rightarrow B$), Safe(a, $A \rightarrow \neg B$), Safe(b, $B \wedge C \rightarrow D$), Safe(c,A), Safe(c,$\neg$A), Safe(c,C), Safe(c,$\neg$C).

2.4 Semantics of the S" logic

A model M for the S" logic is a tuple $M =< W, R, f_{a_1} f_{a_2}, \ldots, f_{a_n}, P >$, where :

- W is a non empty set of worlds,
- R is a relation on WxW, transitive and euclidean,
- each f_{a_i} is a function from W to 2^{2^W}, and
- P is a function from propositional variables to 2^W.

The following constraint C_i is imposed to each function f_{a_i}:

C_i : if r(w) denotes the set of worlds {w' : wRw' }, then for each X such that $X \in f_{a_i}(w)$ we must have $r(w) \subseteq X$.

This constraint is intended to validate the (BEL") axiom schema.

The satisfiability relation is defined as usual:

$M, w \models p$	iff	$w \in P(p)$, if p is a propositional variable
$M, w \models \neg p$	iff	$M, w \not\models p$
$M, w \models p \vee q$	iff	$M, w \models p$ or $M, w \models q$
$M, w \models Bp$	iff	$\forall w'(wRw' \Rightarrow M, w' \models p)$
$M, w \models EB_{a_i} p$	iff	$\|p\| \in f_{a_i}(w)$

where $\|p\|$ denotes the truth set of p: { w' : $M, w' \models p$ }.

The notion of valid formulas is defined as usual by :

$\models_{S''} p$ iff $\forall M =< W, R, .., P >, \forall w \in W, M, w \models p$

Theorem 1: The S" logic is valid and complete.

Sketch of proof [1] The proof of validity is obvious. The proof of completeness uses the technique of canonical minimal models [2].

2.5 Links between S, S' and S"

To show how the S' logic relates to the S logic we define a transformation T that assigns to any sentence of the form Safe(a,q) the sentence Safe'(a,q), and to any sentence of the form E_a(In.DB('q')) the sentence EB_aq. If T is extended to sets of sentences in the natural way, then, if st'=T(st) we should have the property:

$$\vdash_S st \rightarrow K_sBq \quad \text{iff} \quad \vdash_{S'} st' \rightarrow K_sBq, \quad \text{and:} \quad \vdash_S st \rightarrow K_sq \quad \text{iff} \quad \vdash_{S'} st' \rightarrow K_sq$$

To show how the S" logic relates to the S' logic we define a transformation T' that assigns to sentences of the form Safe'(a,q) sentences of the form Safe"(a,q), and to sentences of the form $K_s(EB_aq)$ sentences of the form EB_aq. Then, if st"=T'(st') we should have the property:

$$\vdash_{S'} st' \rightarrow K_sBq \quad \text{iff} \quad \vdash_{S''} st'' \rightarrow Bq, \quad \text{and:} \quad \vdash_{S'} st' \rightarrow K_sq \quad \text{iff} \quad \vdash_{S''} st'' \rightarrow q$$

Notice that the above two properties require only that if we are in the same context we should get the same answers whether we are in the S logic, or in the S' logic, or in the S" logic.

In our work to date we have not formally proved these two properties. So, we can only consider the S logic and the S' logic as intuitive justifications for the definition of the S" logic, and accept the S" logic as a basis for the definition of answers to standard or safety queries.

Nevertheless we can easily understand why we get the same answers in the S' logic and in the S logic.

Let's consider the following derivation of an answer to a safety query regarding q in the S logic:

[1] The reader who is interested by the detailed proofs can found them in the Annex.

(1)	$E_a(In.DB('q'))$	in st
(2)	$K_sE_a(In.DB('q'))$	(1) and (OBS)
(3)	Safe(a,q)	in st
(4)	$K_{adm}(E_a(In.DB('q')) \rightarrow q)$	definition of Safe
(5)	$K_s(E_a(In.DB('q')) \rightarrow q)$	(4) and (SAF)
(6)	K_sq	(2), (5) and (K)

we have in the S' logic the corresponding derivation:

(1')	$K_s(EB_aq)$	in st'
(2')	Safe'(a,q)	in st'
(3')	$K_s(EB_aq \rightarrow q)$	definition of Safe'
(4')	K_sq	(1'), (3') and (K)

Next consider the derivation of an answer to a standard query q in the S logic:

(1)	$E_a(In.DB('q'))$	in st
(2)	$K_sE_a(In.DB('q'))$	(1) and (OBS)
(3)	$K_s(E_a(In.DB('q')) \rightarrow B_aq)$	instance of (S)
(4)	K_sB_aq	(2), (3) and (K)
(5)	K_sBq	(4) and (BEL)

we have in S' the corresponding derivation:

(1')	$K_s(EB_aq)$	in st'
(2')	K_sBq	(1') and (BEL')

Finally, as was remarked at the begining of section 2.3, derivations of answers in the S' and S" logics differ from each other only by the fact that, in the S' logic, the sentences in the derivations are prefixed by K_s.

Moreover if we have $\vdash_S st \rightarrow K_sq$ because q is a tautology, and because q is a tautology implies by necessitation $\vdash_S K_sq$ and $\vdash_S st \rightarrow K_sq$,we also have for the same reasons $\vdash_{S'} st' \rightarrow K_sq$ and $\vdash_{S''} st'' \rightarrow q$. By the the same reasoning, if q is a tautology, we have $\vdash_S K_sBq$, $\vdash_{S'} K_sBq$ and $\vdash_{S''} Bq$.

3 Operational definition

In this section is specifyied, at the meta level, a theorem prover that allows us to generate answers to standard queries as well as answers to safety queries addressed to a database. This specification is defined in First Order Logic by Horn clauses and their translation in Prolog is a simple exercise. An important feature is that the modalities EB and B of S" are respectively represented by two meta predicates Bexp and B, and true beliefs of S" are represented by the meta predicate K.

We assume that the agents do not insert formulas in the database, but they insert sets of clauses which represent these formulas. In the same way, we assume that the reliability of agents is not described in terms of propositional formulas but in term of sets of clauses.

We note cl(f) the set of clauses wich represent a formula f. We assume that the function cl satisfies the following property: two equivalent formulas are associated by cl with two sets of clauses such that any clause in one set is variant of a clause in the other set (.i.e has the same literals than that clause).

We do not restrict ourselves to Horn clauses but we allow agents to insert sets of general clauses.

The prover described in this section, is based on the prover which has been defined previously in [1], for general clauses : the main trick is to transform each general clause into a set of Horn clauses, by renaming negative literals, and to take the factorisation into account.

In 3.1, we define the meta language, called MetaL, that is needed to specify the prover. The specification is given by means of meta axioms presented in 3.2. In 3.3, we show how to represent the database and the queries at the meta level. The soundness and completeness of the meta axioms are proved in 3.4.

3.1 The meta language

Let us denote L a language for Classical Propositional Logic in which the agent beliefs are expressed. We define the first order meta language MetaL in that way :

- there is a constant symbol denoted by "true",
- there are as many constant symbols as agents : $a_1, a_2, \ldots, a_n$,
- there is a constant symbol denoted by "user",
- for any propositional variable P of L, there are two constant symbols denoted by P and notP,
- there is a constant symbol noted Q.
- there is a binary function symbol denoted $\wedge$, whose first argument is a constant symbol P or notP, and whose second argument is the constant "true" or a term built from $\wedge$,
- there is a binary function symbol denoted $\rightarrow$, whose first argument is a term built from $\wedge$, and whose second argument is a constant symbol P or notP; terms of the form $l_1 \wedge (l_2 \wedge (\ldots (l_n \wedge \text{true}) \ldots)$ are abreviated into $l_1 \wedge l_2 \wedge \ldots \wedge l_n \wedge \text{true}$,
- there are predicate symbols B, Bconj, Bexp, K, Kconj, Safe, comp.

The intuitive semantics of these symbols is the following :

- "true" denotes the truth value "true",
- the a_is represent agents,
- "user" represents the user (who is a new agent),
- constants P and notP represent, at the meta level, the literals of the language L. Any positive literal P of L is associated to the constant P in MetaL ; any negative literal $\neg$ P of L is associated to a constant notP in MetaL,
- constant Q represents the query,
- function $\wedge$ is used to represent conjunctions of literals,
- function $\rightarrow$ is used to represent, at the meta level, Horn clauses; the first argument is the antecedent and the second is the consequent; for instance to the clause $A \vee \neg B \vee C$ are associated several Horn clauses like: $\text{notA} \wedge B \rightarrow C$, $\text{notA} \wedge \text{notC} \rightarrow \text{notB}$, or $B \wedge \text{notC} \rightarrow A$,

- $B(l_1 \wedge l_2 \wedge \ldots \wedge l_n \wedge true, l)$, $n \geq 0$ means that $l_1 \wedge \ldots \wedge l_n \rightarrow l$ is a database belief,
- $Bconj(l_1 \wedge \ldots \wedge l_n, l_{n+1} \wedge \ldots \wedge l_m)$ $n \geq 1, m \geq n$ means that $l_1 \wedge \ldots \wedge l_n \rightarrow l_{n+1} \wedge \ldots \wedge l_m$ is a database belief,
- $Bexp(a_i, sc)$ means that the agent a_i has inserted the set of clauses sc,
- $K(l_1 \wedge \ldots \wedge l_n \wedge true, l)$, $n \geq 0$, means that $l_1 \wedge \ldots \wedge l_n \rightarrow l$ is a true belief,
- $Kconj(l_1 \wedge \ldots \wedge l_n, l_{n+1} \wedge \ldots \wedge l_m)$ $n \geq 1, m \geq n$ means that $l_1 \wedge \ldots \wedge l_n \rightarrow l_{n+1} \wedge \ldots \wedge l_m$ is a true belief,
- Safe(a_i, sc) means that the agent a_i is safe as regard to the set of clauses sc,
- comp(l,l') means that l' denotes the complementary literal of l; for instance if l denotes notP, l' denotes P, and vice versa,
- $in(t, t_1 \wedge \ldots \wedge t_n \wedge true)$, $n \geq 1$ means that the literal (or the Horn clause) t is one of the t_is.

Remark : "literals" and "clauses" refer here to the object level. They are thus represented at the meta level by constants or terms of MetaL.

3.2 The meta axioms

Let us consider the following meta axioms where all the variables are assumed to be universally quantified:

(1) $in(l, s) \rightarrow B(s, l)$

(2) $Bexp(a, sc) \wedge in(conj \rightarrow q, sc) \wedge comp(q, q') \wedge Bconj(q' \wedge s, conj) \rightarrow B(s, q)$

(3) Bconj(s,true)

(4) $B(s, b_1) \wedge Bconj(s, b) \rightarrow Bconj(s, b_1 \wedge b)$

(5) $in(l, s) \rightarrow K(s, l)$

(6) $Bexp(a, sc) \wedge Safe(a, sc) \wedge in(conj \rightarrow q, sc) \wedge comp(q, q') \wedge Kconj(q' \wedge s, conj) \rightarrow K(s, q)$

(7) Kconj(s,true)

(8) $K(s, b_1) \wedge Kconj(s, b) \rightarrow Kconj(s, b_1 \wedge b)$

3.3 Description of the database and of the query at the meta level

3.3.1 Hornization

Renaming :

Let L' be an extension of the language L that contains a new propositional variable notP for each propositional variable P of L. We define a mapping pos from sentences of L of the form P or $\neg$P or $\neg\neg$P, to propositional variables of L' which is defined by:

- for each propositional variable P of L: pos(P)=P, pos($\neg\neg$P)=P, pos($\neg$P)=notP.

Horn clauses associated to a clause :

Let $c = l_1 \vee .. \vee l_n$, $n \geq 1$ be a clause of L. Let c_{i_n} be any reformulation $\neg l_{i_1} \wedge \neg l_{i_2} \wedge \ldots \wedge \neg l_{i_{n-1}} \rightarrow l_{i_n}$ of c in an implicative form. Let h_{i_n} be the Horn clause $L_{i_1} \wedge L_{i_2} \wedge \ldots \wedge L_{i_{n-1}} \rightarrow M_{i_n}$ of L' obtained from c_{i_n}, where L_{i_j} denotes pos($\neg l_{i_j}$) and M_{i_n} denotes pos(l_{i_n}); h_{i_n} is called an "hornized form" of c.

We denote by H(c) the conjunction of the Horn clauses which are the hornized forms of c. In the same way, we will denote by **H(cl(q))**, the conjunction of the Horn clauses which are the hornized forms of the clauses which represent a formula q.

3.3.2 State of affairs and query

Let us consider a propositional formula f, which represents a query asked by a user. We introduce in L a new propositional variable, noted Q. Let us recall that cl(f$\leftrightarrow$Q) denotes the clausal form of the formula f$\leftrightarrow$Q.

If st" is a set of sentence of the form: st" = { Safe"(a, q_1), Safe"(a, q_2), ..., Safe"(b, q'_1), Safe"(b, q'_2), ... $EB_a q_i$, $EB_a q_j$, ..., $EB_b q_k$, $EB_b q_l$, ... } where q_is and q'_js are propositional formulas, we define the set meta(st",f) from st" and f by replacing Safe(a,q) in st" by the sentence Safe(a, H(cl(q)) and by replacing $EB_a(q)$ by Bexp(a, H(cl(q)), and by inserting Safe(user, H(cl(f $\leftrightarrow$ Q))) and Bexp(user, H(cl(f $\leftrightarrow$ Q))). So, in formal terms we have:

$$\begin{aligned}\text{meta(st",f)} = &\{ \text{Safe(a, H(cl(q))) : Safe(a,q)} \in \text{st"} \} \cup \\ &\{ \text{Bexp(a, H(cl(q))) : } EB_a q \in \text{st"} \} \cup \\ &\{\text{Safe(user, H(cl(f} \leftrightarrow \text{Q}))), \text{ Bexp(user, H(cl(f} \leftrightarrow \text{Q})))\}\end{aligned}$$

In other terms, meta(st",f) contains the meta information which represent :
- the fact that such an agent has inserted such a formula,
- the fact that such an agent is safe as regard to such a formula,
- the fact that the user is asking such a query and
- the fact that this user is considered to be safe as regard to his formulation of the query.

3.4 Soundness and completeness of the meta axioms

Theorem 2 : Let f be a formula L. Let Q be the new propositional variable introduced previously, we have:

$\vdash_{S''} st'' \rightarrow Bf \Leftrightarrow \{(1), ..., (8)\} \vdash meta(st'', f) \rightarrow B(true, Q)$, and

$\vdash_{S''} st'' \rightarrow f \Leftrightarrow \{(1), ..., (8)\} \vdash meta(st'', f) \rightarrow K(true, Q)$

Sketch of proof. [2]

The first equivalence is proved in three steps :

First of all, we show that deriving, in S" logic, formulas Bf from st" comes to derive, in classical logic, formula f from the set of clauses inserted by the agents.

[2]The reader who is interested by the detailed proofs can found them in the Annex.

Secondly, we re-use a prover that we had defined in previous work and which allows us to derive consequences of a set of general clauses : the main trick is that clauses must be "hornized", and factorization must be taken into account. That prover is , up to minor syntactical details, described by meta-axioms (1),...,(4).

The last step consists in ensuring that when considering more axioms (i.e axioms (5),...,(8)) and additional data (those describing safety) the prover still provides correct answers.

The second equivalence has the same sketch : the first step consists in showing that deriving, in S" logic, formulas f from st", comes to derive, in classical logic, formula f, from the set of all the clauses which have been inserted by some agents who are safe as regard to these clauses. The second step and the third steps are the same as before.

4 A running example

Let us come back to the example given in the introduction. The meta clauses which represents the state of affairs are :

$Bexp(a, (A \rightarrow B) \wedge (notB \rightarrow notA))$
$Bexp(a, (B \rightarrow D) \wedge (notD \rightarrow notB))$
$Bexp(b, (B \wedge C \rightarrow D) \wedge (notD \wedge C \rightarrow notB) \wedge (B \wedge notD \rightarrow notC))$
Bexp(b,C), Bexp(c,A)

$Safe(a, (A \rightarrow B) \wedge (notB \rightarrow notA))$
$Safe(a, (A \rightarrow notB) \wedge (B \rightarrow notA))$
$Safe(b, (B \wedge C \rightarrow D) \wedge (notD \wedge C \rightarrow notB) \wedge (B \wedge notD \rightarrow notC))$
Safe(c,A), Safe(c,notA)
Safe(c,C), Safe(c,notC)

The meta clauses which represent the query $A \rightarrow D$ are represented through the clausal form of: $(A \rightarrow D) \leftrightarrow Q$, i.e. $(A \vee Q) \wedge (\neg D \vee Q) \wedge (\neg A \vee D \vee \neg Q)$:

$Bexp(user, (notA \rightarrow Q) \wedge (notQ \rightarrow A) \wedge (D \rightarrow Q) \wedge (notQ \rightarrow notD) \wedge (A \wedge notD \rightarrow notQ) \wedge (A \wedge Q \rightarrow D) \wedge (notD \wedge Q \rightarrow notA))$

$Safe(user, (notA \rightarrow Q) \wedge (notQ \rightarrow A) \wedge (D \rightarrow Q) \wedge (notQ \rightarrow notD) \wedge (A \wedge notD \rightarrow$

notQ) $\wedge$ (A $\wedge$ Q $\rightarrow$ D) $\wedge$ (notD $\wedge$ Q $\rightarrow$ notA))

The PROLOG clauses corresponding to meta axioms (1) to (8) are:

B(s,l) :- in(l,s)
B(s,q) :- comp(q,q'),Bexp(a, sc), in(conj$\rightarrow$q, sc),Bconj(q'$\wedge$s,conj)
Bconj(s,true)
Bconj(s,$b_1 \wedge b$) :- B(s,b_1),Bconj(s,b)
K(s,l) :- in(l,s)
K(s,q) :- comp(q,q'),Bexp(a, sc),Safe(a, sc), in(conj$\rightarrow$q, sc),Kconj(q'$\wedge$s,conj)
Kconj(s,true)
Kconj(s,$b_1 \wedge b$) :- K(s,b_1),Kconj(s,b)

5 Conclusion

We have presented a general logical framework for reasoning about the safety of information stored in a database, and its simplified version, the S" logic, to derive answers to standard queries and safety queries. The S" logic allows to derive different safety answers from several databases that represent the same theory in different syntactical forms. For instance answers derived from a database that contains $EB_a(p \wedge q)$ are not necessarily the same as answers derived from another database containing EB_ap and EB_aq. However $EB_a(p \wedge q)$ and $EB_a(q \wedge p)$ lead to the same answer. That means that the insertion of two sentences that represent the same proposition are considered to be equivalent.

The operational view of the S" logic is relatively simple and efficient since it is defined by Horn clauses in First Order Logic. There is a limited restriction about the form of inserted sentences since they must be a conjunction of clauses. The reason for this is to avoid to check equivalence of two sentences p and p' that occur in EB_ap and in Safe(a,p'), in the general case.

In future works it will be worth investigating potential applications of the S" logic to the problem of belief revision. Indeed database representation is not purely syntactic like in [8] and it is not a belief set like in [6].

References

[1] A. Bauval and L. Cholvy. Automated reasoning in case of inconsistency. In *Proceedings of WOCFAI*, Paris, France, 1991.

[2] B. F. Chellas. *Modal Logic: An introduction.* Cambridge University Press, 1988.

[3] R. Demolombe and A. Jones. Integrity Constraints Revisited. In A. Olive, editor, *4th International Workshop on the Deductive Approach to Information Systems and Databases.* Universitat Politecnica de Barcelona, 1993.

[4] R. Demolombe and A. Jones. Deriving answers to safety queries. In R. Demolombe and T. Imielinski, editor, *Non Standard Queries and Answers*, Oxford, To appear. Oxford University Press.

[5] D. Elgesem. *Action Theory and Modal Logic.* PhD thesis, University of Oslo, Department of Philosophy, 1992.

[6] P. Gardenfors. *Knowledge in flux : modeling the dynamics of epistemic states.* The MIT Press, 1988.

[7] A. Jones. Toward a Formal Theory of Communication and Speech Acts. In P. Cohen, J. Morgan, and M. Pollack, editors, *Intentions in Communications.* The MIT Press, 1990.

[8] G.M. Kupper, J.D. Ullman, and M. Vardi. On the equivalence of logical databases. In *Proc of ACM-PODS*, 1984.

[9] I. Porn. Action Theory and Social Science. Some Formal Models. *Synthese Library*, 120, 1977.

[10] R. Reiter. What Should a Database Know? *Journal of Logic Programming*, To appear.

Annex

Theorem 1.

The S" logic is valid and complete.

Proof. The proof of validity is quite obvious, then we only present a sketch of the proof of completeness.

The proof technique is based on canonical minimal models (see [2]). We define a canonical minimal model $M^c =< W^c, R^c, f^c_{a_1} f^c_{a_2}, \ldots, f^c_{a_n}, P^c >$ by:

- W^c : set of all the maximal consistent sets of formulas of the S" logic,
- R^c satisfies the property: $Bp \in w$ iff $\forall w'(wRw' \Rightarrow p \in w')$,
- $f^c_{a_i}$ satifies the property: $EB_{a_i}p \in w$ iff $|p| \in f^c_{a_i}(w)$, where $|p|$ denotes the proof set of p: $\{ w : p \in w \}$; $f^c_{a_i}$ is not ambiguous since $|p| = |q|$ implies $\vdash p \leftrightarrow q$, and by RE" we have: $\vdash EB_{a_i}p \leftrightarrow EB_{a_i}q$; therefore we have: $EB_{a_i}p \in w$ iff $EB_{a_i}q \in w$.
- P^c assigns to each propositional variable p the set of worlds w such that $p \in w$.

We first prove that M^c is a model for the S" logic. That comes to prove that the constraint C_i is satified by each function $f^c_{a_i}$.

Let $r^c(w)$ be the set $\{ w' : wR^cw' \}$. If $X \in f^c_{a_i}(w)$, by definition of $f^c_{a_i}$ there exists some sentence p such that $|p| = X$ and $EB_{a_i}p \in w$. By (BEL") we have $EB_{a_i}p \rightarrow Bp$, and, since w is a maximal consistent set for S", we have $Bp \in w$; $Bp \in w$ implies $\forall w'(wRw' \Rightarrow p \in w')$, and from the definition of $r^c(w)$ we have $p \in w'$ for every w' in $r^c(w)$, and therefore we have $r^c(w) \subseteq |p|$. So we have $r^c(w) \subseteq X$.

We prove the property by induction on the complexity of the formulas:

$$(1)\quad M^c, w \models p \quad \text{iff} \quad p \in w$$

For propositional variables the property (1) directly follows from P^c definition.

For $\neg p$ we have:
$M^c, w \models \neg p$ iff $M^c, w \not\models p$, by definition of satisfiability
$M^c, w \not\models p$ iff $p \notin w$, by induction property
$p \notin w$ iff $\neg p \in w$, from maximal consistent sets properties.

For $p \vee q$ we have a similar proof as for $\neg p$.

For Bp we have:
$M^c, w \vdash Bp$ iff $\forall w'(wRw' \Rightarrow M^c, w' \models p)$ by definition of satisfiability
$\forall w'(wRw' \Rightarrow M^c, w' \models p)$ iff $\forall w'(wRw' \Rightarrow p \in w')$ by induction property
$\forall w'(wRw' \Rightarrow p \in w')$ iff $Bp \in w$ by definition of R^c.

For $EB_{a_i}p$ we have:
$M^c, w \models EB_{a_i}p$ iff $\|p\| \in f^c_{a_i}(w)$ by definition of satisfiability
$\|p\| \in f^c_{a_i}(w)$ iff $|p| \in f^c_{a_i}(w)$ since, by induction property $\|p\| = |p|$
$|p| \in f^c_{a_i}(w)$ iff $EB_{a_i}p \in w$ by definition of $f^c_{a_i}$.

Finally if p is a valid sentence of S", i.e. $\models p$, we have $\forall w \in W^c$ $M^c, w \models p$. Therefore, from property (1), we have $\forall w \in W^c$ $p \in w$, and by properties of maximal consistent sets we have $\vdash p$.

Lemma 1.

We adopt the following notations:

- e_i: set of sentences $EB_{a_i}(p_1), EB_{a_i}(p_2), \ldots, EB_{a_i}(p_m)$,
- E_i: sentence $EB_{a_i}(p_1) \wedge EB_{a_i}(p_2) \wedge \ldots \wedge EB_{a_i}(p_m)$,
- f_i: set of sentences: $(EB_{a_i}p_1 \rightarrow p_1), (EB_{a_i}p_2 \rightarrow p_2), \ldots, (EB_{a_i}p_s \rightarrow p_s)$ where for every p_j in some $EB_{a_i}p_j \rightarrow p_j$ in f_i there is a formula $EB_{a_i}p_k$ in e_i such that $\vdash p_j \leftrightarrow p_k$,
- S_i: sentence $(EB_{a_i}p_1 \rightarrow p_1) \wedge (EB_{a_i}p_2 \rightarrow p_2) \wedge \ldots \wedge (EB_{a_i}p_s \rightarrow p_s)$,
- g_i: set of sentences $(EB_{a_i}p'_1 \rightarrow p'_1), (EB_{a_i}p'_2 \rightarrow p'_2), \ldots, (EB_{a_i}p'_t \rightarrow p'_t)$ where for every p'_j in some $EB_{a_i}p'_j \rightarrow p'_j$ in f_i there is no formula $EB_{a_i}p_k$ in e_i, such that $\vdash p'_j \leftrightarrow p_k$,
- T_i: sentence $(EB_{a_i}p'_1 \rightarrow p'_1) \wedge (EB_{a_i}p'_2 \rightarrow p'_2) \wedge \ldots \wedge (EB_{a_i}p'_t \rightarrow p'_t)$,
- p_{a_i}: sentence $p_1 \wedge p_2 \wedge \ldots \wedge p_m$,
- q_{a_i}: sentence $p_1 \wedge p_2 \wedge \ldots \wedge p_s$,
- st": sentence $(E_1 \wedge S_1 \wedge T_1) \wedge (E_2 \wedge S_2 \wedge T_2) \wedge \ldots \wedge (E_n \wedge S_n \wedge T_n)$.

Using these notations we must have $m > 0$ and $n > 0$, and we may have s=0 and/or t=0.

We have: $\vdash_{S''} st'' \rightarrow Bq$ iff $\vdash p_{a_1} \wedge p_{a_2} \wedge \ldots \wedge p_{a_n} \rightarrow q$.

Proof. We first prove the property in the $\Leftarrow$ direction.

Let's assume (1) $\vdash p_{a_1} \wedge p_{a_2} \wedge \ldots \wedge p_{a_n} \rightarrow q$, by necessitation we have (2) $\vdash_{S''} B(p_{a_1} \wedge p_{a_2} \wedge \ldots \wedge p_{a_n} \rightarrow q)$.

For each a_i and for each j in [1,m] we consider the instance of the axiom schema (BEL") (3) $\vdash_{S''} EB_{a_i}(p_j) \rightarrow Bp_j$, then, according to the definition of E_i, we have (4) $\vdash_{S''} E_i \rightarrow Bp_1 \wedge Bp_2 \wedge \ldots \wedge Bp_m$, and, since B obeys the KD45 axioms, we have (5) $\vdash_{S''} E_i \rightarrow B(p_1 \wedge p_2 \wedge \ldots \wedge p_m)$. According to the definition of p_{a_i}, from (5) we have (6) $\vdash_{S''} E_i \rightarrow Bp_{a_i}$, and since st" contains all the E_is we have (7) $\vdash_{S''} st'' \rightarrow Bp_{a_1} \wedge Bp_{a_2} \wedge \ldots \wedge Bp_{a_n}$, which implies (8) $\vdash_{S''} st'' \rightarrow B(p_{a_1} \wedge p_{a_2} \wedge \ldots \wedge p_{a_n})$ because B satisfies KD45.

From (2) and (8) the axiom K allows to infer (9) $\vdash_{S''} st'' \rightarrow Bq$.

For the $\Rightarrow$ direction we have to prove:

(1') $\vdash_{S''} st'' \rightarrow Bq \Rightarrow \vdash p_{a_1} \wedge p_{a_2} \wedge \ldots \wedge p_{a_n} \rightarrow q$

Since S" has been proved to be valid and complete (1') is equivalent to:

(2') $\models_{S''} st'' \rightarrow Bq \Rightarrow \models p_{a_1} \wedge p_{a_2} \wedge \ldots \wedge p_{a_n} \rightarrow q$

And (2') is equivalent to:

(3') $\{p_{a_1}, p_{a_2}, \ldots, p_{a_n}, \neg q\}$ satisfiable $\Rightarrow \{st'', \neg Bq\}$ satisfiable

The proof of proposition (3') is constructive. We define the model $M^0 = < W^0, R^0, f^0_{a_1} f^0_{a_2}, \ldots, f^0_{a_n}, P^0 >$ as follows:

- W^0 contains one unique world w_0,
- for each $f^o_{a_i}$, $f^o_{a_i}$ is defined by $f^o_{a_i}(w_1) = \{\|p_1\|, \|p_2\|, \ldots, \|p_m\|\}$ where $p_1, p_2, \ldots, p_m$ are in p_{a_i},
- P^0 is such that $w_0 \in P^0(q_{a_1} \wedge q_{a_2} \wedge \ldots \wedge q_{a_n} \wedge \neg q)$

If we assume that (4') $\{p_{a_1}, p_{a_2}, \ldots, p_{a_n}, \neg q\}$ is satisfiable, since all the sentences in the q_{a_i}s are in the p_{a_i}s, we have (5') $\{q_{a_1} \wedge q_{a_2} \wedge \ldots \wedge q_{a_n}, \neg q)$ satisfiable. Then it is possible to define P^0 as it is defined in M^0.

We first prove that M^0 is a model of S" . Indeed, since the relation R^0 is serial, transitive and euclidean, axioms of KD45 are satisfied for B.

Moreover we have $r^0(w_0) = \{w_0\}$. If X is such that $X \in f^o_{a_i}(w_0)$ there exists

some p_j such that $X = \|p_j\|$ and p_j is in q_{a_i}. From the definition of P^0 and of q_{a_i} we have $\|p_j\| = \{w_0\}$, therefore we have $r^0(w_0) \subseteq X$. Since there is only one world w_0 in W^0 the constraint C_i is satisfied. Therefore M^0 is a model of S".

For each $EB_{a_i}p_j$ in e_i, from the definition of $f^o_{a_i}$ we have $M^0, w_0 \models_{S''} EB_{a_i}p_j$. Then we have $M^0, w_0 \models_{S''} E_i$.

For each $EB_{a_i}p_j \rightarrow p_j$ in f_i, there exists p_k such that $\vdash p_j \leftrightarrow p_k$ and $EB_{a_i}p_k \in e_i$. Therefore p_k is in q_{a_i}, and from the definition of P^0 we have $M^0, w_0 \models_{S''} p_k$ which implies $M^0, w_0 \models_{S''} p_j$. This fact implies $M^0, w_0 \models_{S''} EB_{a_i}p_j \rightarrow p_j$. Therefore we havee (7') $M^0, w_0 \models_{S''} S_i$.

For each $EB_{a_i}p'_j \rightarrow p'_j$ in g_i, if there exists p_k such that $EB_{a_i}p_k \in e_i$ and $\|p'_j\| = \|p_k\|$, we have from the definition of P^0: $M^0, w_0 \models_{S''} p_k$, then we have $M^0, w_0 \models_{S''} p'_j$. Then we have $M^0, w_0 \models_{S''} EB_{a_i}p'_j \rightarrow p'_j$. If there does not exist such a p_k, for each $EB_{a_i}p_k$ in e_i we have $\|p'_j\| \neq \|p_k\|$, and by definition of $f^o_{a_i}$ we have $M^0, w_1 \models_{S''} \neg EB_{a_i}p'_j$. Then we have $M^0, w_1 \models_{S''} EB_{a_i}p'_j \rightarrow p'_j$. So, we have in both cases $M^0, w_0 \models_{S''} EB_{a_i}p'_j \rightarrow p'_j$, and then we have (8') $M^0, w_0 \models_{S''} T_i$.

From (6'), (7') and (8') we have (9') $M^0, w_0 \models_{S''} st''$.

From the definitions of P^0 and R^0 we have $w_0R^0w_0$ and $M^0, w_0 \models_{S''} \neg q$, then we have (10') $M^0, w_0 \models_{S''} \neg Bq$.

Since there is only one world w_0 in M^0, from (9') and (10') we have $M^0 \models_{S''} st'' \wedge \neg Bq$, wich shows that $\{st'', \neg Bq\}$ is satisfiable.

Lemma 2.

Using the same notations as in Lemma 1 we have:

$\vdash_{S''} st'' \rightarrow q$ iff $\vdash q_{a_1} \wedge q_{a_2} \wedge \ldots \wedge q_{a_n} \rightarrow q$.

Proof. The proof is similar as for Lemma 1. We first prove the property in the $\Leftarrow$ direction.

From the axioms of CPC we have (1) $\vdash_{S''} EB_{a_i}p_j \wedge (EB_{a_i}p_j \rightarrow p_j) \rightarrow p_j$. From the instantiation of (1) for every p_j such that $EB_{a_i}p_j$ is in e_i and such that there exists p_k such that $\vdash p_j \leftrightarrow p_k$ and $EB_{a_i}p_k \rightarrow p_k$ is in f_i, we have (2) $\vdash_{S''} E_i \wedge S_i \rightarrow p_1 \wedge p_2 \wedge \ldots \wedge p_s$. From the definition of q_{a_i} (2) implies (3) $\vdash_{S''} E_i \wedge S_i \rightarrow q_{a_i}$. Then

we have (4) $\vdash_{S''} st'' \rightarrow (q_{a_1} \wedge q_{a_2} \wedge \ldots \wedge q_{a_n})$.

So, if we assume $\vdash q_{a_1} \wedge q_{a_2} \wedge \ldots \wedge q_{a_n} \rightarrow q$, from (4) we infer $\vdash_{S''} st'' \rightarrow q$.

For the proof in the $\Rightarrow$ direction we have to prove:

$\vdash_{S''} st'' \rightarrow q \Rightarrow \vdash q_{a_1} \wedge q_{a_2} \wedge \ldots \wedge q_{a_n} \rightarrow q$

Using the same technique as in the proof of Lemma 1, we have to prove:

$\{q_{a_1}, q_{a_2}, \ldots, q_{a_n}, \neg q\}$ satisfiable $\Rightarrow \{st'', q\}$ satisfiable

If we assume $\{q_{a_1}, q_{a_2}, \ldots, q_{a_n}, \neg q\}$ is satisfiable it is possible to define M^0 as it is defined for Lemma 1. Then we have $M^0 \models_{S''} st''$. From the definition of P^0 we also have $M^0 \models_{S''} \neg q$. So, we have $M^0 \models_{S''} st'' \wedge \neg q$, and {st",$\neg$q} is satisfiable.

Lemma 3.

Let us consider the notations:

$$db(st") = \bigwedge_{EB_a p \in st''} p$$

$$m(st",f) = \{Bexp(a, H(cl(q))) : EB_a q \in st''\} \cup Bexp(user, H(cl(f \leftrightarrow Q)))$$

We have:

$$\vdash db(st'') \rightarrow f \quad \Leftrightarrow \quad \{(1),(2),(3),(4)\} \vdash m(st'', f) \rightarrow B(true, Q)$$

Proof. This lemma is a reformulation of a result given in [1]. In this previous work, we have defined a theorem prover that allows us to prove at the meta level, theorems from a set of general clauses. This prover is defined by meta axioms (1), (2), (3) and (4). We have proved the soundness and the completeness of this prover in the paper cited below, i.e deriving the formula f from a set of general clauses comes to prove formula B(true,Q) with the axioms (1), ... ,(4) from a set of meta clauses which represent the hornized form of the given clauses and of formula $f \leftrightarrow Q$.

In this previous work, we did not pay attention to the agents who have inserted the clauses in the database, but indexing the formulas by their origin is an obvious extension.

Lemma 4.

With the same notations as in Lemma 3 and in section 3.3 we have:

$\{(1),...,(4)\} \vdash m(st'',f) \rightarrow B(true,Q) \Leftrightarrow \{(1),...,(8)\} \vdash meta(st'',f) \rightarrow B(true,Q)$

Proof. The proof for the $\Rightarrow$ direction is obvious.

We prove the $\Leftarrow$ direction in the semantics by showing that if $\{(1),\ldots,(4), m(st'',f), \neg B(true,Q)\}$ is satisfiable, then $\{(1),\ldots,(8), meta(st'',f), \neg B(true,Q)\}$ is satisfiable. Indeed, we can build a (classical first order) model of the second set by extending a model of the first one in the following way : we give valuations of propositions of the type K and Kconj in order to satisfy axioms (5),...,(8). We can notice that their satisfaction does not imply to change the valuations of B(true,Q) (which remains false).

Lemma 5.

Let us consider the notations:

V(q',q): we have V(q',q) iff q and q' are two conjunctions of clauses such that for each clause in q' there is a clause in q formed with the same set of literals, and vice versa. V(q',q) can be read "q' is a variant of q",

$S(st") = \{\ q : EB_a q \in st''$ and $\exists q'$ such that V(q',q) and $Safe(a,q') \in st''\ \}$

$$sdb(st") = \bigwedge_{q \in S(st'')} q$$

$$sm(st",f) = \Big(\bigwedge_{q \in S(st'')} EB_a q\Big) \wedge \Big(\bigwedge_{V(q',q) \wedge q \in S(st'')} Safe(a,q')\Big)$$

With these notations we have:

$\vdash sdb(st'') \rightarrow f \quad \Leftrightarrow \quad \{(5),(6),(7),(8)\} \vdash sm(st'',f) \rightarrow K(true,Q)$

Proof. The proof is, like proof of Lemma 3, a reformulation of the result proved in [1].

Lemma 6.

With the same notations as in Lemma 5 and in section 3.3 we have:

$\{(5),...,(8)\} \vdash sm(st'',f) \rightarrow K(true,Q) \Leftrightarrow \{(1),...,(8)\} \vdash meta(st'',f) \rightarrow K(true,Q)$

Proof: the proof is similar as the proof of Lemma 4.

Theorem 2.

With the same notations as in section 3.3 we have:

$\vdash_{S''} st'' \rightarrow Bf \quad \Leftrightarrow \quad \{(1),...,(8)\} \vdash meta(st'',f) \rightarrow B(true,Q)$ and

$\vdash_{S''} st'' \rightarrow f \quad \Leftrightarrow \quad \{(1),...,(8)\} \vdash meta(st'',f) \rightarrow K(Q,true)$

Proof: the proof of the first equivalence is a direct application of lemmas 1, 3 and 4; the proof of the second equivalence is a direct application of the lemmas 2, 5 and 6.