

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

Laurence Cholvy¹ and Robert Demolombe¹ and Andrew Jones²

¹ ONERA/CERT, Toulouse, France

² 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.

keywords: Logic for Artificial Intelligence, Modal Logic, Reliability.

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 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 $\text{Safe}(a, A \rightarrow B)$ (resp. $\text{Safe}(a, A \rightarrow \neg B)$). It is important to notice that the system knows $\text{Safe}(a, A \rightarrow B)$ and $\text{Safe}(a, A \rightarrow \neg B)$ 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 unemployment 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: $\text{Safe}(b, B \wedge C \rightarrow D)$.

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

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:

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

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: $A, A \rightarrow B, B, B \rightarrow D, D$ 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.

We have developed 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. 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 $: \text{in.DB}(m)$, and its intended meaning is “the message m is stored in DB”. E is closed under logical equivalence:

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

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

$$(ET) \quad E_a p \rightarrow p$$

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

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

Since we accept (RE) and (R¬N) we must reject the distribution 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:

$$(Ec) \quad (E_a p_1 \wedge E_a p_2) \rightarrow E_a(p_1 \wedge p_2).$$

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 $: \text{in.DB}(m_i)$, 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 :

$$(BEL) \quad K_s B_a q \rightarrow K_s B q$$

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

Concept of Safety.

Now the concept of Safety is defined by :

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

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:

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

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 :

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

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 $K_a q \stackrel{\text{def}}{=} (B_a q) \wedge q$.
- 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 \neg 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) $K_s(E_a(\text{in.DB}('q'))) \rightarrow B_a q$
 - (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 $\text{Safe}(a, q)$, which are defined by:

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

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:

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

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: $E_a(\text{In.DB}('q'))$ are replaced by sentences of the form $EB_a q$, where EB_a is a new modality. $EB_a q$ 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 $EB_a q$ we do not consider the syntactical form of q , but only its meaning, we have to accept the following inference rule:

$$(RE') \frac{q_1 \leftrightarrow q_2}{EB_a q_1 \leftrightarrow EB_a q_2}$$

However the modality $EB_a q$ 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 $K_s q \stackrel{\text{def}}{=} (B_s q) \wedge q$.
- For each modality of the form EB_a we have:
 $(RE') \frac{q_1 \leftrightarrow q_2}{EB_a q_1 \leftrightarrow EB_a q_2}$
- 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 $\text{Safe}'(a, q)$ 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:

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

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:
 $(RE') \frac{q_1 \leftrightarrow q_2}{EB_a q_1 \leftrightarrow EB_a q_2}$
- The link between the modalities EB_a and B is defined by:
 $(BEL'') EB_a q \rightarrow Bq$
- 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:

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

It might be noticed that from $\text{EB}_a q$ 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 $\neg \text{EB}_a(\neg q)$. Then, due to properties of material implication we have $\text{Safe}''(a, \neg q)$. 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 $\text{Safe}''(a, \neg q)$, we have to have $\text{EB}_a(\neg q)$, which together with $\text{EB}_a q$ leads to an inconsistency.

We do not have this problem with the definition of Safe' because to have $\text{Safe}'(a, \neg q)$, $\neg \text{EB}_a(\neg q)$ 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:

$$\text{st}'' = \{ \text{Safe}''(a, q_1), \text{Safe}''(a, q_2), \dots, \text{Safe}''(b, q'_1), \text{Safe}''(b, q'_2), \dots, \text{EB}_a q_i, \text{EB}_a q_j, \dots, \text{EB}_b q_k, \text{EB}_b q_l, \dots \}$$

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

2.4 Semantics of the S'' logic

A model M for the S'' logic is a tuple $M = \langle W, R, f_{a_1} f_{a_2}, \dots, f_{a_n}, P \rangle$, where :

- W is a non empty set of worlds,
- R is a relation on $W \times W$, 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:

$$\begin{array}{lll} M, w \models p & \text{iff} & w \in P(p), \text{ if } p \text{ is a propositional variable} \\ M, w \models \neg p & \text{iff} & M, w \not\models p \\ M, w \models p \vee q & \text{iff} & M, w \models p \text{ or } M, w \models q \\ M, w \models Bp & \text{iff} & \forall w' (wRw' \Rightarrow M, w' \models p) \\ M, w \models \text{EB}_{a_i} p & \text{iff} & \|p\| \in f_{a_i}(w) \end{array}$$

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 $\forall M = \langle W, R, \dots, P \rangle, \forall w \in W, M, w \models p$

Theorem 1. *The S'' logic is valid and complete.*

Proof. 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 $E_a(\text{In.DB}(q'))$ the sentence $EB_a q$. If T is extended to sets of sentences in the natural way, then, if $st' = T(st)$ we should have the property:

$$\vdash_S st \rightarrow K_s Bq \text{ iff } \vdash_{S'} st' \rightarrow K_s Bq, \text{ and } : \vdash_S st \rightarrow K_s q \text{ iff } \vdash_{S'} st' \rightarrow K_s q$$

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 $K_s(EB_a q)$ sentences of the form $EB_a q$. Then, if $st'' = T'(st')$ we should have the property:

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

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.

3 Operational definition

In this section is specified, 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 which 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.

The specification is given by means of meta axioms presented in 3.1. In 3.2, we show how to represent the database and the queries at the meta level. The soundness and completeness of the meta axioms are presented in 3.3.

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

The predicate $B(s, q)$ (resp. $K(s, q)$) means that $s \rightarrow q$ is a database belief (resp. a true database belief). $Comp(q, q')$ means that q' is a complementary literal of q . The meta axiom (1) says that if l is a literal in the clause s , then $s \rightarrow l$ is a database belief. The meta axiom (2) says that if $conj \rightarrow q$ is in the set of agent's explicit beliefs and $\neg q \wedge s \rightarrow conj$ is a database belief, then $s \rightarrow q$ is a database belief. Axiom (3) says that $s \rightarrow true$ is a database belief. Axiom (4) says that if $s \rightarrow b$ and $s \rightarrow b_1$ are database beliefs then $s \rightarrow b_1 \wedge b$ is a database belief. Axioms (5) to (8) have a similar meaning as axioms (1) to (4).

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

Horn clauses associated to a sentence. A sentence q is represented by its clausal form $cl(q)$, and clauses are represented by sets of Horn clauses by renaming negative literals. This set of Hornclauses is denoted by $H(cl(q))$. For instance, if q is the sentence $A \vee B$, we have: $H(cl(q)) = \{\text{not}A \rightarrow B, \text{not}B \rightarrow A\}$.

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: $st'' = \{ Safe''(a, q_1), Safe''(a, q_2), \dots, Safe''(b, q'_1), Safe''(b, q'_2), \dots, EB_a q_i, EB_a q_j, \dots, EB_b q_k, EB_b q_l, \dots \}$ where q_i 's and q'_i 's 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:

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

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.

3.3 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:*

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

Proof. We have shown that deriving, in S'' logic, formulas like Bf or f from st'' comes to derive, in classical logic, formula f from the set of clauses inserted by the agents, and we re-use properties of a prover that we had defined in previous work [1].

4 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 $EB_a p$ and $EB_a q$. 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 $EB_a p$ 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].

Acknowledgement : This work was partially supported by the ESPRIT BRA project MEDLAR2.

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.