

# A Framework for Fusing Consistent Knowledge Bases Automatically

Éric Grégoire

CRIL & IRCICA  
Université d'Artois  
F-62307 Lens  
France

gregoire@cril.univ-arts.fr

Du Zhang

Department of Computer Science  
California State University  
Sacramento, CA 95819-6021  
USA

zhangd@ecs.csus.edu

## Abstract

Knowledge-based systems play a pivotal role in the information economies. Knowledge base inconsistencies can adversely affect a system's correctness and performance. In this paper, we propose a framework for fusing several knowledge bases together to produce a consistent one. The framework is based on two cornerstones: detecting inconsistencies using algorithmic techniques to compute minimally unsatisfiable formulas, and adopting a logic-based weakening approach to restore consistency for the fused knowledge base. We study the dynamics in the framework in terms of both the model-theoretic and the fixpoint semantics for the fused knowledge base.

**Keywords:** knowledge base fusion, minimally unsatisfiable formulas, inconsistency detection and resolution.

## 1. Introduction

Knowledge-based systems play a pivotal role in today's information economies. Knowledge-based technology has found its way into so many problem domains [5] and has been the cornerstone to numerous successful applications [29, 16]. A crucial component of a knowledge-based system is its knowledge base (KB) that contains knowledge about a problem domain [8, 20]. Knowledge base development involves domain analysis, context space definition, ontological specification, and knowledge acquisition, codification and verification (Figure 1).

It is important to recognize the context under which domain-specific knowledge is formulated and reasoned about when developing a KB for an application. A context is a region in some  $n$ -dimensional space [19]. In the common sense KB, Cyc [5], there are thousands of contexts. Some of its top dimensions of the context space include time, space, logic, relations, paths, physical objects, artifacts, movement, and so forth. In a KB development process, domain analysis should result in identification of the region of interest in the context space.

Specifying a context entails defining or locating a point or region along each of those  $n$  dimensions.

Clearly specifying the context space within which a KB is developed has a number of benefits [19]. It suppresses the bulk of irrelevant knowledge so as to concentrate on the relevant one to the problem at hand, which results in better inference and search performance. It allows for terse and sloppy communications among users and developers, and accommodates contradictory information through separating inconsistent information into different contexts. Similar or relevant assertions can be entered in the same or nearby context in terse and simpler form.

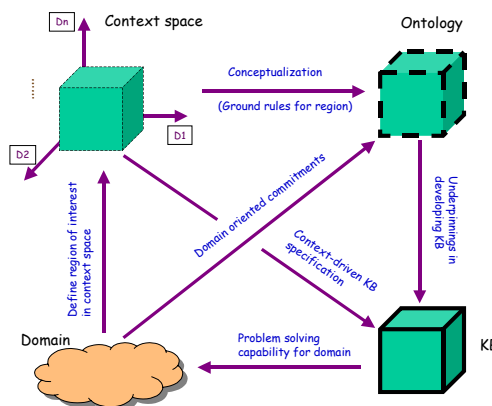


Figure 1. KB development process.

To categorize knowledge into contexts, we need to avoid the following granularity pitfalls: too coarse-grained is likely having no contexts at all, and too fine-grained makes it hard to find the right context for a new piece of knowledge [19].

After determining the context (or contexts) for a problem domain, ontological development is in order. An ontology is a formal, explicit specification of a shared conceptualization [3, 10, 24]. Conceptualization refers to an abstract model consisting of relevant concepts about a problem domain. Explicitly defined are the following: concepts and concept taxonomies, relationships between

and properties of concepts, and the constraints on concept use. The “shared” implies that an ontology captures consensual conceptualization accepted by groups of people. Having an ontology in place before developing a *KB* promotes reuse and sharing. There are many contemporary ontologies, methodologies and languages for building ontologies, and ontology tools [10].

After the ontology is in place, knowledge acquisition, codification and verification can be carried out to build the *KB* for a given application. Several *KBs* may need to be fused to produce an amalgamated one. Inevitably, there will be anomalies in a *KB* as a result of existing practices in its development process. Anomalies such as inconsistency can affect the correctness and performance of an intelligent system, though some systems are robust enough to perform rationally in the presence of the anomalies.

In this paper, we consider the issue of how to detect *KB* inconsistency during *KB* fusion process and how to restore consistency in the fused *KB*. We propose a framework for fusing several *KBs* together to produce a consistent one. The framework is based on two cornerstones: isolating inconsistencies using algorithmic techniques to compute *minimally unsatisfiable formulas* (MUSes), and adopting a logic-based weakening approach to restore consistency for the fused *KB*.

The paper is organized as follows. In the next section, we provide some necessary background information about Boolean forward chaining *KBs*, their semantics and inconsistencies. In Section 3, we discuss how inconsistencies can be identified through detecting MUSes in a given *KB*. Section 4 describes a technique that can resolve the detected inconsistencies in the fused *KB*. The main properties of the dynamics of this technique are then presented in Section 5. Finally in Section 6, we conclude the paper with remarks on possible future work.

## 2. Technical background

The focus in this paper is on forward-chaining rule-based *KBs*. A *KB* has a set of facts that is stored in a working memory (WM) and a set of rules stored in a rule base (RB). Rules represent general knowledge about an application domain. Facts in a WM provide specific information about the problem at hand and may be elicited either dynamically from the user during each problem-solving session, or statically from the domain expert during knowledge acquisition, or derived through rule deduction.

Rules and facts are formulas belonging to a Boolean language  $\mathbf{L}$  of formulas over a finite alphabet  $\mathbf{P}$  of Boolean variables, also called *atoms* or *propositions*. The  $\wedge$ ,  $\vee$ ,  $\neg$  and  $\Rightarrow$  symbols represent the standard conjunctive, disjunctive, negation and implication

connectives, respectively. A *literal* is an atom or its negation.  $\top$ <sup>1</sup> represents inconsistency. From a syntactical point of view, a *KB* will be a multi-set of formulas of  $\mathbf{L}$  (we shall often abuse words and mention set for multi-set). In Section 4, we will consider a multi-set of rule bases  $E = \{KB_1, \dots, KB_n\}$  to be fused ( $n > 1$ ).

We assume that rules in a *KB* have the following format:  $P_1 \wedge \dots \wedge P_n \Rightarrow R$ , where  $P_i$ s are the conditions (collectively, the left-hand side, LHS, of a rule),  $R$  is the conclusion (or right-hand side, RHS, of a rule). The  $P_i$ s and  $R$  are literals. If the conditions of a rule are satisfied by facts in WM, then the rule is enabled and its firing deposits its conclusion into WM. A fact is represented as a atom. It specifies a proposition in a given problem domain. WM contains a collection of positive facts which are deposited through either assertion (initial or dynamically), or rule deduction using forward chaining.

A negated condition  $\neg P$  in the LHS of a rule is satisfied if  $P$  is not in WM. A negated conclusion  $\neg R$  in the RHS of a rule results in the removal of  $R$  from WM when  $R$  is in WM. Rules and negated literals can be utilized by the inference system, but never deposited into WM.

Let  $WM_0$  denote the initial state for WM. We use  $WM_i$  ( $i=1,2,3,\dots$ ) to represent subsequent states of WM as a result of firing all enabled rules under the state of  $WM_{i-1}$ .

Actually, a language used in developing a *KB* is frequently enriched with other valuable epistemological concepts. For instance, mutual exclusive literals that cannot be true at the same time (such as “animal” and “vegetable”) and incompatible ones, which are complementary pairs of synonymous literals (such as “expensive” and “-high price”), are often explicitly or implicitly asserted [35]. In the following, the aforementioned concepts are accommodated in our proposed framework.

The semantics of *KB* can be described using several techniques. The first one is the *fixpoint semantics* [34]. For a given *KB*, we can define a transformation  $T_{KB}$  in the following way. A single step of  $T_{KB}$  to *KB* amounts to generating a set of ground literals, denoted as  $\vdash_{WM_i} RB$ , which is obtained by firing all enabled rules in RB under  $WM_i$ . It can be shown that  $T_{KB}$  is monotonic and has a least fixpoint  $lfp(T_{KB})$  with regard to the partial order  $\leq_k$  (knowledge ordering) [9].  $lfp(T_{KB})$  contains all the derivable conclusions and is denoted as  $\vdash_{lfp}$ . Let  $g$  be a literal,  $KB \vdash_{lfp} g$  when  $g \in lfp(T_{KB})$ . When both  $g$  and  $\neg g$  (or a pair of contradicting literals) are in  $lfp(T_{KB})$ , the *KB*

<sup>1</sup> Here, we adopt the usual convention from many four-valued logics that makes use of  $\top$  to denote *overdefined* or *contradiction* in the truth value set of  $\{true, false, \perp, \top\}$ . Let us stress that the bottom symbol  $\perp$  is often used to represent contradiction in standard binary logic.

is said to be (*lfp*-)inconsistent or contains (*lfp*-)inconsistent knowledge. This is denoted as  $KB \vdash_{\text{lfp}} \top$ .  $KB$  is (*lfp*-)consistent iff  $KB \not\vdash_{\text{lfp}} \top$ .

If we treat a  $KB$  as a set of logic formulas, we can then apply the model-theoretic approach to define the semantics for the  $KB$ . Let  $\Omega$  denote the set of all *interpretations* of  $\mathbf{L}$ , which are functions assigning either *true* or *false* to every atom. A *model*  $\xi$  of  $KB$  is an interpretation of  $\Omega$  that satisfies every formula of  $KB$ . The set of models of  $KB$  will be denoted  $[[KB]]$ .  $KB \models g$  when the literal  $g$  is entailed from  $KB$ , i.e., when  $g$  is true in all models of  $KB$ .  $KB$  is *inconsistent* or *unsatisfiable* when  $[[KB]]$  is empty, also denoted by  $KB \models \top$ .

Though we use the terms of rules/facts and formulas interchangeably, the two semantics of  $\vdash_{\text{lfp}}$  and  $\models$  do not necessarily coincide for a given  $KB$  in the sense that we do not have  $KB \vdash_{\text{lfp}} g$  iff  $KB \models g$ . For example, let  $KB = \{a, b \Rightarrow \neg a\}$ .  $KB \models \neg b$ , but we do not have  $KB \vdash_{\text{lfp}} \neg b$ . Indeed, using a forward chaining mechanism, we are not able to derive  $\neg b$  from  $KB$  although  $\neg b$  is a logical consequence of  $KB$ .

To ascertain the presence of inconsistency in a  $KB$ , we utilize a concept called *minimally unsatisfiable formulas* or MUSes for a  $KB$ . A MUS captures a smallest possible set of formulas in a  $KB$  that encode some contradiction.

**Definition 2.1.** A MUS  $\Gamma$  of a  $KB$  is a set of formulas s.t.  $(\Gamma \subseteq KB)$  and  $([[\Gamma]] = \emptyset)$  and  $(\forall S \subset \Gamma: [[S]] \neq \emptyset)$

Identifying MUSes for a  $KB$  can exhibit a high worst case computational complexity. Indeed, even in the simple Boolean clausal framework, determining whether a set of clauses is a MUS is  $D^P$ -complete [26], and checking whether a formula belongs to the set of MUSes of contradictory sets of clauses is in  $\Sigma_2^P$  [7]. Moreover, the number of MUSes for a set of  $n$  clauses is  $C_n^{n/2}$ . However, this high complexity often deflates to a tractable one in real-life. New techniques to compute MUSes are now available [14, 15]. They are based on the use of a (failed) local search for consistency as an oracle to locate MUSes. More precisely, they make use of a *critical clause* concept and a heuristic stating that the formulas that are most often falsified during this failed search for consistency belong most probably to these MUSes. Extensive computational studies [14, 15] on difficult benchmarks [6, 30] show that this approach is frequently the most efficient one with respect to all current competing ones.

Though defined under  $\models$ , MUS can be adapted with respect to the *lfp*-semantics as well. To this end, the (un)satisfiable condition is to be replaced by the absence (presence) of contradictory facts in the least fixpoint. Throughout the remainder of the paper, we shall use  $\text{MUS}_{\vdash_{\text{lfp}}}$  and  $\text{MUS}_{\models}$  to distinguish the two concepts.

When there is no confusion, we use the term MUS as a generic one for both  $\text{MUS}_{\models}$  and  $\text{MUS}_{\vdash_{\text{lfp}}}$ .

A  $KB$  can contain several MUSes. The inconsistent part of  $KB$  is denoted  $\cup\text{MIN-UNSAT}(KB)$  and is defined as follows.

**Definition 2.2.** The *inconsistent part* of a  $KB$ , denoted  $\cup\text{MIN-UNSAT}(KB)$ , is the set-theoretic union of all MUSes of  $KB$ .

Since MUSes can exhibit non-empty intersections, the concept of *inconsistent cover* [15] proves valuable in that it does not require all MUSes to be made explicit, but provides us with enough mutually independent inconsistency causes that need to be addressed to restore consistency. This concept can be defined with both the  $\text{MUS}_{\vdash_{\text{lfp}}}$  and  $\text{MUS}_{\models}$  characterizations.

**Definition 2.3.** An *inconsistent cover*  $IC$  of a  $KB$  is a set-theoretic union of MUSes of  $KB$  s.t.  $KB \setminus IC$  is consistent.

### 3. Detecting inconsistency via MUSes

As illustrated earlier, the relations  $\vdash_{\text{lfp}}$  and  $\models$  do not necessarily coincide. What this illuminates is the discrepancy between the inconsistency characterizations under the two. Given a  $KB = \{a, b \Rightarrow \neg a, d, \neg b \Rightarrow \neg d\}$ . Clearly,  $KB$  is logically inconsistent w.r.t.  $\models$ , but not under  $\vdash_{\text{lfp}}$ . In this specific case, although the forward chaining mechanism will not allow us to derive any contradiction, we argue that the knowledge in the  $KB$  is inherently inconsistent<sup>2</sup> and this information will be important to the knowledge engineer. Thus, inconsistency under the *lfp* semantics is a sufficient but not necessary condition to inconsistency under the model-theoretic semantics.

**Property 3.1.** Whenever  $KB \vdash_{\text{lfp}} \top$ , we also have  $KB \models \top$ , but not conversely. Whenever  $\Gamma$  is a  $\text{MUS}_{\vdash_{\text{lfp}}}$  of  $KB$ , we also have that  $\Gamma$  is a  $\text{MUS}_{\models}$  of  $KB$ , but not conversely.

Because algorithmic techniques are already in place to determining  $\text{MUS}_{\models}$  [14, 15], the aforementioned property affords us a direct way to compute MUSes under  $\vdash_{\text{lfp}}$ . To this end,  $\text{MUS}_{\models}$  are first computed. Any  $\text{MUS}_{\models}$  that exhibits a least fixpoint that contains contradictory literals is also a  $\text{MUS}_{\vdash_{\text{lfp}}}$ . Let us stress that the cost of this second step is negligible from a computational point of view since  $\text{MUS}_{\vdash_{\text{lfp}}}$  can be computed in polynomial time with respect to the size of the  $\text{MUS}_{\models}$ , which is often small in real-life applications [14].

<sup>2</sup> Conceivably,  $KB$  can be evolved through some equivalence preserving transformations (say, contrapositive law) into  $KB' = \{a, a \Rightarrow \neg b, d, d \Rightarrow b\}$  where  $KB' \vdash_{\text{lfp}} \top$ .

Table 1 illustrates the results of MUS detection algorithm in [14, 15] on some *difficult* KBs from the Dimacs [6] and SATLIB [30] benchmarks.

Table 1. MUS detection algorithm benchmark results.

KB	#atoms	#clauses	(#atoms,#clauses) in MUS
dp02u01	213	376	(47,51)
dp03u02	478	1007	(327,760)
23.cnf	198	474	(165,221)
42.cnf	378	904	(315,421)
fpga10_11_uns_rer	220	1122	(110,561)(110,561)
fpga11_12_uns_rer	264	1476	(132,738)(132,738)
ca008	130	370	(110,255)
C202_FW_UT_2814	2038	11352	(15,18)

These are very difficult KBs with respect to the MUSes detecting problem. For example, the fpga11\_12\_uns\_rer contains two MUSes with each involving 738 different clauses, i.e., the *minimal* number of clauses that can be involved in a proof of inconsistency is 738. Although the computational time to solve these benchmarks remains tractable, it should be noted that the size of MUSes for real-life instances is often very small, making the computation even more time-efficient [15].

#### 4. Consistency restoring

When several consistent KBs are fused, additional inconsistencies can occur due to the interaction of the KBs. Once inconsistencies are detected through MUSes, the next step is how to resolve the inconsistencies. In this section, we describe a technique that allows the consistency to be restored in an automatic way in the fused KB without the intervention of the knowledge engineer. In particular, our focus is on the dynamics of restoring consistency during the fusion process when several KBs are fused successively.

Many techniques have been proposed to fuse knowledge bases, in the logic programming context (see e.g. [1, 31]) or in other logic-based settings (see e.g. [4, 13, 17, 18, 21, 22]). In [11, 12], an original way has been introduced to fuse standard logic KBs in an automatic manner. This approach restores consistency in the fused KB if MUSes are detected during the fusion process. Its main feature lies in a generic technique to weaken formulas belonging to MUSes. This contrasts with the common practice of dropping conflicting formulas by existing logic-based techniques to fuse KBs [2, 13].

We adopt the technique in [11, 12] as part of our framework. The essence of this technique involves weakening the contradictory formulas in MUSes by augmenting them with additional literals. Some properties in the fusion process can be observed through reasoning with those literals. In the rest of the section, we give a brief overview of the approach, and then describe how it is incorporated into the forward chaining rule-based systems.

Finally we describe some properties in terms of the  $\vdash_{\text{fip}}$  and  $\models$  semantics.

We assume here that every  $KB_i$  to be fused is given an enumeration of its  $m$  formulas. In the general case, this general enumeration has no meaning and is not intended to represent any respective importance of formulas in  $KB_i$ . An ordered list  $(Flag^{i-1}, \dots, Flag^{i-m})$  of atoms, called *Flags*, is associated to this enumeration. Flags will be introduced inside KBs using our fusion process or in a way to enforce a preference of one KB over the others, only.

Now, assume that we need to fuse two consistent KBs, namely  $KB_1$  and  $KB_2$  and that  $[[KB_1 \cup KB_2]] = \emptyset$ . To restore consistency, we weaken *some* formulas in  $\cup\text{MIN-UNSAT}(KB_1 \cup KB_2)$ . Depending on the way to select those formulas, we obtain a series of weakening policies.

**Definition 4.1.** The set of weakened formulas of  $KB$ , noted  $WKF(KB)$ , is a subset of  $\cup\text{MIN-UNSAT}(KB)$  s.t. for every MUS of  $KB$ , it contains at least one formula from each  $KB$  taking part in that MUS.

Actually, there can often exist many possible candidate  $WKF(KB)$ . In the following, we assume that such a set is selected, and concentrate on the common properties of all corresponding approaches. Accordingly, a  $\cup_{\text{weaken}1}$  fusion by weakening operator was defined as follows in a CNF setting [12].

**Definition 4.2.** When  $[[KB_1 \cup KB_2]] \neq \emptyset$ ,  
then  $KB_2 \cup_{\text{weaken}1} KB_1$  is defined as  $KB_1 \cup KB_2$ .  
When  $[[KB_1 \cup KB_2]] = \emptyset$ ,  
then  $KB_2 \cup_{\text{weaken}1} KB_1$  is defined as  
 $(KB_1 \cup KB_2) \setminus WKF(KB_1 \cup KB_2) \cup$   
 $\{f \vee Flag^{i-1} \text{ s.t. } f \in WKF(KB_1 \cup KB_2) \cap KB_1$   
and  $i$  is the rank of  $f$  in  $KB_1\} \cup$   
 $\{f \vee Flag^{2-j} \text{ s.t. } f \in WKF(KB_1 \cup KB_2) \cap KB_2$   
and  $j$  is the rank of  $f$  in  $KB_2\}$

The  $\cup_{\text{weaken}1}$  operator is easily iterated.

**Definition 4.3.** Let  $n > 2$ . We denote  $KB$  for  
 $KB_n \cup (KB_{n-1} \cup_{\text{weaken}1} \dots \cup_{\text{weaken}1} KB_1)$   
When  $[[KB]] \neq \emptyset$ , then  $KB_n \cup_{\text{weaken}1} (KB_{n-1} \cup_{\text{weaken}1} \dots$   
 $\cup_{\text{weaken}1} KB_1)$  is defined as  $KB$   
When  $[[KB]] = \emptyset$ , then  $KB_n \cup_{\text{weaken}1} (KB_{n-1} \cup_{\text{weaken}1} \dots$   
 $\cup_{\text{weaken}1} KB_1)$  is defined as  
 $KB \setminus WKF(KB)$   
 $\cup \{f \vee Flag^{i-1} \text{ s.t. } f \in WKF(KB) \cap KB_1$   
and  $i$  is the rank of  $f$  in  $KB_1\}$   
 $\cup \dots$   
 $\cup \{f \vee Flag^{n-j} \text{ s.t. } f \in WKF(KB) \cap KB_n$   
and  $j$  is the rank of  $f$  in  $KB_n\}$

Now, it is easy to reformulate these definitions in the context of forward chaining rule-based systems and  $\vdash_{\text{fip}}$ ,

and get a  $\cup_{\text{weaken2}}$  operator. For instance, we can decide to mark rules that occur in the inconsistent part of the set-theoretic union of the  $KB$ s by inserting their corresponding flags in the LHS. In such a way, provided that  $WM_0$  is consistent,  $\vdash_{\text{lfp}}$  consistency of the merged  $KB$ s is ensured. From a computational point of view, it is also sufficient to consider one inconsistency cover to regain consistency. However, in this case, a unique possible fused  $KB$  is guaranteed only when a unique IC exists.

**Example 4.1.** Assume  $KB_1 = \{a, a \Rightarrow b\}$  and  $KB_2 = \{c, c \Rightarrow \neg b\}$ . We observe that both of the following results hold:  $[[KB_1 \cup KB_2]] = \emptyset$  and  $[KB_1 \cup KB_2] \vdash_{\text{lfp}} \top$ . There is one unique MUS and thus one unique inconsistent cover IC in  $KB_1 \cup KB_2$ , namely  $KB_1 \cup KB_2$  itself. Accordingly, weakening all rules in the IC, the resulting fused  $KB$  will be given by  $KB_1 \cup_{\text{weaken2}} KB_2 = \{a, a \wedge \text{Flag}^{1-2} \Rightarrow b, c, c \wedge \text{Flag}^{2-2} \Rightarrow \neg b\}$ . Clearly, for the fused  $KB$  to be consistent, it requires that  $\text{Flag}^{1-2}$  and  $\text{Flag}^{2-2}$  must not be true at the same time. Enforcing one of these two flags (e.g.  $\text{Flag}^{1-2}$ ) to be true amounts to preferring its corresponding  $KB$  ( $KB_1$ ) over the other one ( $KB_2$ ).

It is important to note that flags are not mere markers. Since they become part of rules, they can be reasoned about under  $\models$ . However, using the forward chaining mechanism only, we cannot infer any piece of information pertaining to these flags, unless some flags are already in  $WM_0$ . Such a weakening rule schema can be traced to the exception-blocking refinement rule mechanism that has been introduced in the frameworks of non-monotonic logics [23] and in the model-based diagnosis [28]. However, its use to restore consistency w.r.t. both the  $\vdash_{\text{lfp}}$  and  $\models$  semantics, and the study of its dynamics in an iterated fusion process are, to the best of our knowledge, new.

The above operators obey some nice properties.

**Properties 4.1.** Assume  $KB_1, \dots, KB_n$  are  $n$  consistent rule bases.

1.  $[[KB_n \cup_{\text{weaken2}} \dots \cup_{\text{weaken2}} KB_1]] \neq \emptyset$
2. Let  $i \in [1..n]$  we have that  $[[KB_n \cup_{\text{weaken2}} \dots \cup_{\text{weaken2}} KB_1 \cup \{\text{Flag}^{i-j} \forall j\}]] \neq \emptyset$
3.  $\text{WFK}(KB_n \cup KB_{n-1})$  does not have any rule containing a flag.
4.  $\nexists \text{Flag}^{i-j}$  s.t.  $KB_n \cup_{\text{weaken2}} \dots \cup_{\text{weaken2}} KB_1 \models \neg \text{Flag}^{i-j}$
5.  $\nexists \text{Flag}^{i-j}$  s.t.  $KB_n \cup_{\text{weaken2}} \dots \cup_{\text{weaken2}} KB_1 \vdash_{\text{lfp}} \neg \text{Flag}^{i-j}$
6.  $\nexists \text{Flag}^{i-j}$  s.t.  $KB_n \cup_{\text{weaken2}} \dots \cup_{\text{weaken2}} KB_1 \vdash_{\text{lfp}} \text{Flag}^{i-j}$

Property 4.1.1 ensures that the resulting fused  $KB$  is consistent with both the  $\vdash_{\text{lfp}}$  and  $\models$  semantics. Property 4.1.2 shows how a preference for some initial  $KB$  over the other ones can be enforced, by setting all its related flags to true (e.g. by inserting them in  $WM_0$ ). Property 4.1.3

ensures that whenever a formula is weakened, it cannot belong to a future MUS in any subsequent fusion step. Properties 4.1.4 and 5 are best interpreted as follows. Asserting  $\text{Flag}^{i-j}$  to be true enforces a preference for  $KB_i$ , and once this preference is asserted, the fusion process cannot question it in the future. Property 4.1.6 ensures that the fusion process cannot guarantee by itself an explicit preference for a given  $KB_i$  that would modify the set of inferences according to  $\vdash_{\text{lfp}}$  (let us stress that Property 4.1.6 does not hold for  $\models$  [12]).

## 5. Dynamics in the fusion schema

Although Properties 4.1.5 and 6 ensure that no explicit preference for one  $KB_i$  can be inferred using the forward chaining mechanism, this does not force the fusion schema to be unbiased with respect to the several rule-based systems to be fused in an iterated fashion. Indeed, flags are introduced in the formulas themselves and interact with the knowledge that these latter ones represent.

**Example 5.1.** Assume we iterate the fusion process that began in Example 4.1. with  $KB_3 = \{d, d \Rightarrow b\}$ . Clearly,  $KB_3 \cup_{\text{weaken2}} (KB_1 \cup_{\text{weaken2}} KB_2) = \{a, a \wedge \text{Flag}^{1-2} \Rightarrow b, c, c \wedge \text{Flag}^{2-2} \Rightarrow \neg b, d, d \Rightarrow b\}$ . Indeed,  $KB_3 \cup (KB_1 \cup_{\text{weaken2}} KB_2)$  is both  $\models$  and  $\vdash_{\text{lfp}}$  consistent.

The above example illustrates that when a conflict between two  $KB$ s has been neutralized using flags during a previous fusion process, nothing prevents a piece of information from a third  $KB$  that is consistent with the previous resulting fused knowledge from conducting one branch of the alternative of the initial conflict to be adopted.

The following theorem states the conditions for the iterated fusion process to be associative, thus entails no preference for one of the  $KB_i$ . Intuitively, it requires that all MUSes (from the set-theoretic union of the initial  $KB_i$ s) that share a non-empty intersection to consist of formulas coming from the same  $KB_i$ s.

**Theorem 5.1.** Let  $\text{MUS}$  denote either  $\text{MUS}_{\models}$  or  $\text{MUS}_{\vdash_{\text{lfp}}}$ . Let  $\mathbf{K} = \{K \text{ s.t. } K \text{ is a MUS of } KB_n \cup \dots \cup KB_1\}$ ,  $\cup_{\text{weaken1}}$  is associative w.r.t.  $KB_1, \dots, KB_n$  iff  $\nexists (K_i, K_j) \in \mathbf{K} \times \mathbf{K}$  s.t.

- $K_i \cap K_j \neq \emptyset$
- and  $\exists KB_r \in \{KB_1, \dots, KB_n\}$  s.t.  $(\exists f \in KB_r \cap K_i \text{ and } \nexists g \in KB_r \cap K_j)$

## 6. Conclusions and future work

The results presented in this paper have been obtained in the propositional logic framework. From a conceptual point of view, they could be easily extended to the first-order case, i.e. to a framework that allows the use of predicates over finite domains in rules. However, it remains to be seen how the algorithmic techniques to

extract MUSes could be extended accordingly, with the help of an instantiation schema into the Boolean case, provided that any variable is bound to a specific limited-size domain. The technical results presented in this paper could be valuable tools in the pursuit from knowledge base verification and validation perspective (see e.g. [25, 27, 32, 33]). Since logical consistency checking is only a basic tool in that respect, extending the results here into those problem domains is an exciting path for future research, especially in the context of various intelligent systems where rule bases are only a basic component.

## References

1. C. Baral, S. Kraus, J. Minker and V.S. Subrahmanian, "Combining knowledge bases consisting of first-order theories", *Computational Intelligence*, vol. 8(1), pp. 45-71, 1992.
2. I. Bloch and A. Hunter (guest eds), "Fusion: general concepts and characteristics", *Int. Journ. of Intelligent Systems*, vol. 16, 2000.
3. B. Chandrasekaran, J.R. Josephson and V.R. Benjamins, "What are ontologies, and why do we need them?", *IEEE Intelligent Systems*, vol. 14(1), pp. 20-26, 1999.
4. L. Cholvy, "Reasoning about merging information", *Handbook of Defeasible Reasoning and Uncertainty Management Systems*, vol. 3, pp. 233-263, 1998.
5. Cycorp, <http://www.cyc.com/>
6. Dimacs Benchmarks on SAT (<ftp://dimacs.rutgers.edu/pub/challenges/satisfiability/>)
7. T. Eiter and G. Gottlob, "On the complexity of propositional knowledge base revision, updates and counterfactual", *Artificial Intelligence*, vol. 57, pp. 227-270, 1992.
8. R. Fagin, J.Y. Halpern, Y. Moses and M.Y. Vardi, *Reasoning about knowledge*, The MIT Press, Cambridge Mass., 1995.
9. M. Fitting, "Fixpoint semantics for logic programming: a survey", *Theoretical Computer Science*, vol. 278(1-2), pp. 25-51, 2002.
10. A. Gomez-Perez, M. Fernandez-Lopez and O. Corcho, *Ontological Engineering*, Springer-Verlag, 2004.
11. É. Grégoire, "An unbiased approach to iterated fusion by weakening", *Information Fusion*, vol 7(1), pp. 35-40, 2006.
12. É. Grégoire, "About the dynamics of iterated knowledge fusion by weakening", *Proc. of the IEEE IRI'05 Conference*, D. Zhang et al. (eds), pp. 326-331, 2005.
13. É. Grégoire and S. Konieczny, "Logic-based approaches to information fusion", *Information Fusion*, vol. 7(1), pp. 4-18, 2006.
14. É. Grégoire, B. Mazure and C. Piette, "Extracting MUSes", *Proc. of the 17<sup>th</sup> European Conference on Artificial Intelligence (ECAI'2006)*, Trento, 2006.
15. É. Grégoire, B. Mazure and C. Piette, "Tracking MUSes and strict inconsistent covers", *CRIL Research Report 2006-RR-02*, 2006.
16. IBM's research project on Information Economies, <http://www.research.ibm.com/infoecon/index.html>
17. S. Konieczny and R. Pino Perez, "Merging with integrity constraints", *Proc. of Ecsqaru'99*, pp. 233-244, LNCS 1638, Springer, 1999.
18. S. Konieczny, "On the difference between merging knowledge bases and combining them", *Proc. of KR'2000*, pp. 135-144, 2000.
19. D. Lenat, "The dimensions of context-space", *CYCorp Report*, October 1998.
20. H.J. Levesque and G. Lakemayer, *The Logic of Knowledge Bases*, The MIT Press, Cambridge Mass., 2000.
21. J. Lin, "Integration of weighted knowledge bases", *Artificial Intelligence*, vol. 83, pp. 363-378, 1996.
22. J. Lin and A.O. Mendelson, "Merging databases under constraints", *Int. Journ. of Cooperative Information Systems*, 7(1), pp. 55-76, 1998.
23. J. McCarthy, "Applications of circumscription to formalizing common-sense knowledge", *Artificial Intelligence*, vol. 28, pp. 89-116, 1986.
24. D.E. O'Leary, "Using AI in knowledge management: knowledge bases and ontologies", *IEEE Intelligent Systems*, vol. 13(3), pp. 34-39, 1998.
25. D.E. O'Leary and A. Preece, *AAAI Workshop on Verification and Validation of Knowledge-Based Systems*, 1998.
26. C. Papadimitriou and D. Wolfe, "The complexity of facets resolved", *Journ. of Computer and System Sciences*, vol. 37(1), pp. 2-13, 1988.
27. R. Plant and G. Antoniou, *Knowledge Based Systems (Special issue on verification and validation)*, vol. 12(1-2) 1999.
28. R. Reiter, "A theory of diagnosis from first principles", *Artificial Intelligence*, vol. 32, pp. 57-95, 1987.
29. R.G. Ross, *Principles of the Business Rule Approach*, Addison-Wesley, 2003.
30. SATLIB Benchmarks on SAT (<http://www.intellektik.informatik.tu-darmstadt.de/SATLIB/benchm.html>)
31. V.S. Subrahmanian, "Amalgamating knowledge bases", *ACM Transactions on Database Systems*, vol. 19(2), pp. 291-331, 1994.
32. J. Vanthienen and F. van Harmelen (eds), *Proc. of the Fourth European Symp. on the Validation and Verification of Knowledge-Based Systems EUROAV'97*, Leuven, Belgium, 1997.
33. A. Vermesan and F. Coenen (eds), *Verification and Validation of Knowledge Based Systems: Theory, Tools and Practice*, Kluwer Academic Publishers, 1999.
34. D. Zhang, "Fixpoint semantics for rule-based anomalies", *Proc. of the Fourth IEEE Int. Conf. on Cognitive Informatics*, Irvine, pp.10-17, 2005.
35. D. Zhang and Luqi, "Approximate declarative semantics for rule base anomalies", *Knowledge-based Systems*, vol. 12(7), pp. 341-353, 1999.