Disabling Subsumptions in a Logic-Based Component

Éric Grégoire Sébastien Ramon
CRIL CNRS UMR 8188 - Université d'Artois
Rue Jean Souvraz SP18
F-62307 Lens, France
{gregoire,ramon}@cril.fr

Abstract

In this paper, we address a problem that is often overlooked in the gradual construction of a logic-based knowledge component: how can a new piece of information g be added into a component KC so that g is not subsumed by KC? More precisely, the focus is on iterating this subsumption-freeness enforcement process when the component is increased step by step. Interestingly, it is shown that the approach and results initially related to standard Boolean logic can directly apply to some non-monotonic frameworks.

1 Introduction

The representation and handling of evolving knowledge has long been a major topic of Artificial Intelligence and knowledge engineering research, especially when knowledge is expressed in logic [1, 2, 3, 4, 5]. In this respect, updating and revising knowledge and beliefs remain very fertile domains of research [6]. This paper is concerned with a problem that has often been overlooked so far in such a context. When a new piece of knowledge g is not logically conflicting with a knowledge component KC, i.e. when g is consistent with KC, it is often assumed that g should simply be added into KC. This postulate is shared by most theoretical studies about the dynamics of knowledge, like knowledge and belief revision or update [6, 7, 8, 9, 10], and knowledge merging or fusion [11, 12]. All the aforementioned studies and the abundant subsequent literature concentrate on handing inconsistent situations [13]. Recently, we have shown that in real-life circumstances, inserting g can require further modifications of KC when g must not be subsumed by KC [14, 15] and when g is consistent with KC. For example, assume that a rule $\mathcal{R}_1$ "When the switch is on and the lamp bulb is ok then the light is on" is to be inserted within KC, which already contains $\mathcal{R}_2$ "When the switch is on then the light is on". Clearly $\mathcal{R}_1$ and $\mathcal{R}_2$ are not logically conflicting. Actually, $\mathcal{R}_1$ is *subsumed* by $\mathcal{R}_2$ in the sense that, from a logical point of view, $\mathcal{R}_1$ is a mere strict deductive consequence of $\mathcal{R}_2$. Accordingly, both rules cannot co-exist in a same knowledge component KC if we want $\mathcal{R}_1$ to prevail. Otherwise, nothing would prevent $\mathcal{R}_2$ to apply and conclude that the light is on, provided that the switch is on, even when the lamp bulb is not ok. Another example is as follows. Assume that an agent believes that Peter is in his office or at home. A new, more informative piece of knowledge comes up: "Peter is in his office or at home or in his car", leaving open the additional possibility that Peter is actually in his car. This last piece of information is however again a mere strict deductive consequence of the former knowledge, not conflicting with it. In some way, the former knowledge should be expelled from KC to really enforce the contents of the new -more informative- one. However, subsumption between formulas is not always that apparent; in the worst case, subsumption between two formulas can only be established by making use of all formulas of KC.

An original approach addressing this problem has been introduced in [14]: both its algorithmic facets and formal aspects have been investigated. This former study has been extended to a general non-monotonic setting in [15] and applied to the legal domain in [16]. In this paper, the focus is on the dynamics of such a process: how could the insertion of knowledge be iterated, taking into account the above treatment of information that must not be subsumed. Interestingly, this leads us to define a new form of *integrity constraints*: knowledge that cannot be logically strengthened. A second contribution of the paper is in showing that the involved standard logic apparatus offers a sufficient framework for expressing the most important aspects of this subsumption-related issue for an interesting case of non-monotonic logics.

The formal framework in this paper is standard Boolean logic. On the one hand, it is the simplest possible setting for presenting and discussing the above subsumption-related issues. On the other hand, recent dramatic progress

in Boolean search and reasoning has now revived Boolean logic as a realistic framework for representing large knowledge sets and solving numerous complex reasoning artificial intelligence tasks [17]. Furthermore, as noted earlier, the results in this paper will be shown relevant to some more expressive non-monotonic logics.

The paper is organized as follows. In the next Section, basic notions about Boolean logic are recalled. The subsumption issue is presented in Section 3. In Section 4, the iteration of the subsumption-freeness enforcement process is investigated. Computational experimental results are provided and discussed in Section 5. Section 6 focuses on extending the results to a generic form of non-monotonic logic. The paper ends with perspectives and promising paths for further research.

2 Logic-Based Framework

To concentrate on the aforementioned conceptual problems, we consider the very simple framework of standard Boolean logic. Let $\mathcal{L}$ be a language of formulas over a finite alphabet $\mathcal{P}$ of Boolean variables, also called *atoms*. Atoms are noted $a, b, c, \ldots$ The $\wedge, \vee, \neg, \Rightarrow$ and $\Leftrightarrow$ symbols represent the standard conjunctive, disjunctive, negation, material implication and equivalence connectives, respectively. A *literal* is an atom or a negated atom. Formulas are built in the usual way from atoms, connectives and parentheses; they are noted $f, g, h, \ldots$ A formula is in conjunctive normal form (CNF) when expressed as a conjunction of clauses, where a clause is a disjunction of literals. The semantical concepts needed in the paper are as follows. Let Ω denote the set of all interpretations of $\mathcal{L}$, which are functions assigning either *true* or *false* to every atom. A *model* of a set of formulas KC is an interpretation of Ω that satisfies every formula of KC. KC is *consistent* when its set of models is not empty. $KC \vdash f$ expresses that the formula f can be deduced from KC, i.e. that it is *true* in all models of KC.

From a syntactical point of view, a *knowledge component* KC will thus be a set of formulas of $\mathcal{L}$. We opt for a semantical (vs. a purely syntactical) regard of KC. Under this point of view, KC is identified with the set of all its deductive consequences. Anyway, we will still distinguish between the formulas that are *explicitly* present in KC vs. formulas that are only *implicit* deductive consequences of the formulas that are explicitly present in KC. For convenience purpose, KC will be represented by its set of explicit formulas.

A word of caution might also be needed for readers who are familiar with rule-based systems but not with logic. We exploit the full (sound and complete) inferential capability of Boolean logic, i.e. we do not only simply allow for mere forward and backward chaining on $\Rightarrow$ as in traditional rule-based systems. For example, from the rule $a \Rightarrow b$ and $\neg b$, we infer $\neg a$ using contraposition. Also, the reader not familiar with logic should always keep in mind that a rule of the form $(a \wedge b \wedge \neg c) \Rightarrow (d \vee e)$ is logically equivalent to $\neg a \vee \neg b \vee c \vee d \vee e$, and will be treated as such. Actually, in the following, we consider clausal KCs and clauses, only.

3 Preempting Subsumption

In this Section, the approach in [14] for preempting subsumption in the Boolean framework is briefly summarized.

Two central concepts in this paper are the *strict implicant* and the *subsumption* ones, defined as follows.

Definition 1. *Let f and g be two formulas. f is a strict implicant of g iff $f \vdash g$ but $g \nvdash f$. KC subsumes g iff $KC \vdash f$ for some strict implicant f of g.*

For example, $KC = \{office \vee home\}$ subsumes $office \vee home \vee car$. Indeed, $office \vee home$ is a strict implicant of $office \vee home \vee car$. Obviously, f (as mentioned in $KC \vdash f$) needs not be explicit in KC but can be a mere implicit formula in KC, i.e. a deductive consequence f of the explicit formulas of KC such that f is not itself explicit in KC.

In [14], it has been shown that if the choice is to modify KC before inserting a new formula g that must not be subsumed, then it is necessary to first delete from KC all possibilities to deduced $g \Rightarrow f_i$ for any strict implicant f_i of g. Alternatively, g can be introduced inside KC; then, all strict implicants of g must be expelled from the augmented KC. Interestingly, when the CNF format of the formulas is considered, it is sufficient to expel the longest strict implicants, in terms of the number of involved literals. In the above example, e.g. ensuring that the strict implicant $office \vee home$ of $office \vee home \vee car$ is expelled is sufficient to guarantee that e.g. the smaller implicant $office$ is also expelled (otherwise we would have $office \vdash office \vee home \vdash office \vee home \vee car$).

In order to check whether a clause is subsumed or not, a first straightforward preprocessing step could check whether any of its n strict longest sub-clauses is actually explicit in the component. This can be performed efficiently in $O(n, m)$, where m is the total number of clauses in the component. This check can actually easily look for the presence of smaller implicants as well. Obviously, this would not detect more complex subsumption paths. From a computational point of view, checking whether a formula subsumes another one is $coNP$-complete, and thus intractable in the worst case. However, recent dramatic progress in Boolean and search makes it often possible to get answers within seconds, especially thanks to powerful SAT solvers [17, 18], which check whether a set of clauses is consistent or not.

In [14], we have experimented an original method to address the subsumption issue. The goal is not only to answer whether the formula is subsumed or not, but also to deliver clauses that can be required to be expelled in order to eliminate the subsumption links. The technique considers clausal KCs and clauses, only. It is based on the SAT-solvers technology and on methods [17, 18] for delivering so-called MUSEs (Minimal Unsatisfiable Subsets). More precisely: assume we need to check whether KC (that contains g) subsumes g through a strict implicant f of g. The solver considers the new set of formulas $KC \cup \{\neg f\}$, which can only be inconsistent in case of subsumption. Then, the solver looks for the MUSes, namely the (cardinality-)minimal sets of clauses that are inconsistent. Making sure that at least one clause in each of the MUSes is dropped ensures that the subsumption link disappears. Expelling such clauses can be automatic, or the knowledge engineer can be asked whether she (he) really wants to drop them, or even be given the choice of selecting the clauses to be dropped in the MUSes. When he (she) prefers keeping these MUSes intact, she (he) is then conducted to revise his (her) former requirement about the subsumption-freeness status of g. This solver provides efficient results even for huge KCs, provided that the number of MUSes remains small [19]. As an alternative to computing all MUSes, the solver also allows to compute a cover of MUSes, which is formed of enough MUSes to explain all inconsistencies.

At this point, it is important to stress that the above subsumption-freeness enforcement process should often only concern the subpart of KC that concerns complex rules rather than mere facts. Indeed, for example, assume that the fact "Light is on" is in KC. Then this fact subsumes e.g. any rule having "Light is on" as a consequence, like "If the switch is on and the lamp bulb is ok then the light is on". Accordingly, like in the first expert systems, we distinguish between a working memory (made of basic assertions translated as literals) and the so-called rule-based one, made of more complex formulas translating generic knowledge that is expected to be more permanent. In the following, KC will *always* denote this latter part of the set of formulas that must be taken into account to investigate and solve the subsumption issue.

Finally, the process can take into account a traditional concept of *integrity constraints*: these formulas cannot be expelled from KC to ensure the subsumption-freeness of a formula. The set of integrity constraints of KC is noted KC_{IC}.

4 Iterating the Process

When g is inserted inside KC and arrangements are made so that g is not subsumed by KC, two basic questions arise if we iterate the process. Assume that we now introduce another additional formula h in KC.

- If h must not be subsumed, what should be done if g is an obstacle for achieving this other subsumption-freeness insertion? In case h and g cannot be subsumption-free at the same time, what policy should be observed: do we need to give a higher priority to one of the formulas? In the positive case, which one?
- What should be done if the insertion of h conducts KC to subsume g?

Clearly, to address these questions, we do not only need to know, at each insertion step, which formulas in KC can be altered by the additional formulas to obey the subsumption-freeness requirements. We also need to know if all formulas that are to be subsumption-free must receive an equal treatment, or if some priorities should arbitrate between them.

First, we define two possible statuses for formulas in KC: a formula can be either *permissive* or *restrictive* with respect to subsumption. A restrictive formula in KC is an explicit formula in KC that is not only subsumption-free in KC but must remain subsumption-free if a new piece of information is then added into KC. We note KC_R the set of restrictive formulas in KC. No formula in KC_R is thus subsumed in KC. To some extent, elements of KC_R represent integrity constraints of a new type, made of formulas that cannot be logically strengthened. Also, like KC_{IC}, it is natural to expect KC_R to be small with respect to KC. Furthermore a small cardinality will ease the process from a computational point of view.

Let us stress that there is no *a priori* specific links between KC_R and KC_{IC}. It might happen e.g. that some or all elements of KC_R belong to KC_{IC} but those sets are independent in the general case.

Now, the set of permissive formulas in KC is given by $KC_P = KC \setminus KC_R$. Note that formulas in KC_P can be implicit and can (vs. must) be subsumed in KC.

As a case study, we investigate a form of preference for the more recently introduced information when two expectedly restrictive formulas cannot be subsumption-free at the same time. Accordingly, we suppose that KC_R is actually a LIFO-stack structure.

In Algorithm 1, the general skeleton of the algorithm for the iterated introduction of formulas involving the subsumption-freeness enforcement is provided, when all formulas are restricted to clauses.

Interestingly, the traversal of the stack of restrictive formulas is unique, since, because standard logic is monotonic, any deletion of formulas cannot alter the subsumption-freeness of remaining formulas. The subsumption-freeness enforcement policy considers the global set of MUSes for all longest strict implicants of g and requires each element

to be broken by expelling one of its clauses, in interaction with the user or not. Alternative policies could require each MUS to be broken as soon as it is extracted, or compute and handle a cover of MUSes for each implicant. Also integrity constraints from KC_{IC} are protected from being expelled, although this is not detailed in these algorithms.

Obviously enough, the actual implementation of the algorithm should not be direct. Especially, redundant or useless computations should be avoided. Let us just stress on two features in that respect. First, to show that a formula is not subsumed by one of its longest strict implicants, models or counter-models are derived. They should be recorded so that they are checked when the same question arises at a next step of the process. Second, KC can be made of unrelated components. The newly introduced piece of information g does not need to be checked against parts of KC having no possible connection with g.

Algorithm 1: Main

Data: –
begin
  $KC \longleftarrow \emptyset$; $KC_{IC} \longleftarrow \emptyset$;
  $KC_R \longleftarrow empty\_stack()$;
  $finished \longleftarrow false$;
  while $finished$ *is* $false$ **do**
    write("New clause: "); read(g);
    write("Integrity constraint?: "); read(is_ic);
    write("Restrictive?: "); read(is_res);
    construct($g, is\_ic, is\_res, KC, KC_{IC}, KC_R$);
    write(Finished?:); read($finished$);
end

Algorithm 2: construct

Data: $g, is\_ic, is\_res, KC, KC_{IC}, KC_R$
begin
  if is_ic *is* $true$ **then**
    $ic\_ok \longleftarrow$ check_IC($KC \cup \{g\}, KC_{IC} \cup \{g\}$);
    if ic_ok *is* $true$ **then**
      $KC_{IC} \longleftarrow KC_{IC} \cup \{g\}$;
    else
      interactWithUser(g, KC, KC_{IC}, KC_R);
      exit;
  if $KC \cup \{g\}$ *is inconsistent* **then**
    revise(g, KC, KC_{IC}, KC_R);
  $KC \longleftarrow KC \cup \{g\}$;
  if is_res *is* $true$ **then**
    push(g, KC_R);
  enforce(KC, KC_{IC}, KC_R);
end

Algorithm 3: enforce

Data: KC, KC_{IC}, KC_R
begin
  while $g \longleftarrow$ *nextElement(*KC_R*) is not* $null$ **do**
    $MUSes \longleftarrow \emptyset$;
    forall *longest strict sub-clauses* f_i *of* g **do**
      if $KC \cup \{\neg f_i\}$ *is inconsistent* **then**
        $MUSes \longleftarrow MUSes \cup$ extractMUSes($KC \cup \{\neg f_i\}$);
    if $MUSes \neq \emptyset$ **then**
      breakMUSes($MUSes, KC, KC_{IC}, KC_R$);
end

5 Experimental results

All experimentations have been conducted on a plain PC (IntelCore 2 Quad 2.66GHz - 4Gb Ram) under Linux Ubuntu 11.10 (3.0.0-16-generic). The solver is freely available from `http://www.cril.univ-artois.fr/~ramon/preempte`. In Table 1, a sample of typical experimental results is provided for the central and computationally-heavy part of the algorithm, namely the MUSes detection step. It presents the performance of the algorithm for that procedure, on usual SAT (i.e. set of Boolean clauses) benchmarks from earlier SAT competitions. The columns indicate the name of the benchmark, the number of involved clauses and of different variables, and the number of MUSes. Then, the performance of the extraction of one MUS and a cover of MUSes is then given, together with the size of the MUS and cover. The timeout was set to 250 seconds. Our belief is that in most real-life knowledge components, those MUSes will remain small (smaller than in those benchmarks made for challenging SAT solvers) and of a manageable number, since a new formula generally interacts with a small number of small subparts of the existing KC, only.

6 Pushing the Enveloppe

The present work has been developed within the standard Boolean logical setting. Interestingly, it also applies -with no additional computational cost- to a generic form of non-monotonic logic that is suitable for representing rules with exceptions, where these exceptions can depend on consistency checks.

In the following, we resort to McCarthys Abnormality notation [20] to emphasize and encode possible exceptions to rules. Let $\mathcal{A}b$ be a subset of $\mathcal{P}$. Its elements are noted $Ab_1, \ldots, Ab_m$ and called abnormality variables. They are intended to represent exceptions to rules that can be either

instances	#cla	#var	#MUSes	Finding all MUSes		Finding a cover of MUSes	
				#sec	#cla in MUSes	#sec	#cla in MUSes
battleship-5-8-unsat	105	40	1	0.23	105	0.18	105
battleship-6-9-unsat	171	54	1	1.88	171	0.48	171
battleship-10-10-unsat	550	100	-	*timeout*		19.58	417
battleship-11-11-unsat	726	121	-	*timeout*		82.21	561
battleship-12-12-unsat	936	144	-	*timeout*		172.93	711
5cnf_3500_3900_30f1.shuffled	420	30	-	*timeout*		17.01	194
5cnf_3900_3900_060.shuffled	936	60	-	*timeout*		33.99	777
marg3x3add8ch.shuffled-as.sat03-1448	272	41	1	177.15	272	7.10	272
marg3x3add8.shuffled-as.sat03-1449	224	41	1	8.79	224	3.48	224
php_010_008.shuffled-as.sat05-1171	370	80	-	*timeout*		1.14	370
rand_net60-30-1.shuffled	10 681	3 600	-	*timeout*		233.46	10 681
C208_FA_UT_3254	6 153	1 876	17 408	2.94	98	17.49	40
C208_FA_UT_3255	6 156	1 876	52 736	6.67	102	18.97	40

Table 1. A Sample of Typical Experimental Results for SAT Benchmarks (MUSes detecting step).

deduced through the standard logic deductive apparatus, or assumed inexistent by default.

For example, the rule "If the switch is on and the lamp bub is ok and if it can be consistently assumed that the switch is ok then the light is on" can be represented by the knowledge component: $KC = \{switch\_on \wedge \neg Ab_1 \wedge \neg Ab_2 \Rightarrow light\_on, \neg switch\_ok \Rightarrow Ab_1, lamp\_bulb\_ok \Leftrightarrow \neg Ab_2\}$.

Intuitively, KC is considered under a non-monotonic reasoning schema in the following way. Assume that we take into account in KC the additional contextual factual information $switch\_on$ and $lamp\_bulb\_ok$. Then, all Ab_i are considered. Roughly, when neither Ab_i nor $\neg Ab_i$ can be established then $\neg Ab_i$ is assumed by default. In the example, $\neg Ab_i$ is assumed by default and KC allows $light\_on$ to be inferred.

More formally, the non-monotonic inferential apparatus can be defined as follows:

Definition 2. *Let us index any model of KC by the set of Ab_i atoms that the model satisfies. A minimal model of KC is a model of KC such that no proper-subset of its index is the index of a model of KC.*

Definition 3. *A formula f can be non-monotonically inferred from KC (noted $KC \mid\sim f$) iff f is true in all minimal models of KC.*

A useful point here is that such a non-monotonic logic covers standard Boolean logic in the sense that whenever $KC \vdash f$ we also have that $KC \mid\sim f$. Now, a question is: in order to move up the subsumption issue to this non-monotonic framework, should we replace the standard entailment relation $\vdash$ by $\mid\sim$ in both the standard-logic definitions (see Definition 1) of subsumption and strict implicant?

Actually, we do not make such a step here for the following reason. First, remember that in the classical logic approach we do not consider the factual part of the knowledge to handle the subsumption issue. Now, in the absence of a proof of Ab_i from KC, we have that $KC \mid\sim \neg Ab_i$. Accordingly, we have that any formula of the form $\neg Ab_i \vee \ldots$ (equivalent to $\ldots \wedge Ab_i \Rightarrow$) is non-monotonically entailed from KC and would thus be subsumed by KC. Classifying all those formulas as subsumed ones would not meet the same objective that motivated our treatment of subsumption in classical logic. Indeed, these formulas would be subsumed because of additional factual assumptions made by default ($\neg Ab_1$ in the example). These assumptions do not necessarily represent the actual context of KC. In the same way that we have left apart from the subsumption issue the factual part of the knowledge translated by literals, we left here also apart the additional augmenting $\neg Ab_i$ assumptions. In this way, we keep our policy coherent with our motivation of separating the knowledge into two separate components: the factual information about actual circumstances on the one side, and more complex permanent rules on the other side. The factual information, either explicit in KC or by default, is not thus considered in this treatment of subsumption, be it classical or non-monotonic.

In this respect, by not taking into account these additional default assumptions, the treatment of subsumption in the non-monotonic setting collapses down to the treatment of subsumption within classical logic.

Obviously enough, this decision to consider subsumption according to the standard logic interpretation of a non-monotonic knowledge component cannot be justified for all non-monotonic logics and all circumstances. For example, in [15], we have refined the implicant and subsumption concepts to fit a finer-grained non-monotonic framework,

where, among other things: 1. The treatment of exceptions is more local in the sense that not every possible default information is necessarily assumed. 2. It is decided to focus on the specific context of KC by including the involved factual information in KC. 3. The factual contextual information around the fired rules involving exceptions default is itself taken into account in the subsumption issue.

7 Conclusions and Perspectives

Enforcing subsumption-free formulas is a difficult task from both conceptual and computational perspectives. However, it is essential to avoid inadequate and unexpected inferences from a knowledge component. The contribution of this paper about this issue is twofold. On the one hand, we have shown how such an enforcement process can be iterated. On the second hand, we have shown that this procedure can be applied to a non-monotonic extension of classical logic that is suitable for representing rules with exceptions. This research opens many perspectives. First, it remains to extend the framework to more expressive logics. Especially, finite fragments of first-order logics are natural candidates in this respect. Also, adapting and grafting the approach to description logics and answer set programs is also a promising path for future research. We intend to pursue these lines of research in the future.

Acknowledgments

This work has been supported in part by the *Région Nord/Pas-de-Calais* and the EC through a FEDER grant.

References

[1] J. Doyle. A truth maintenance system. *Artificial Intelligence*, 12:231–272, 1979.

[2] M. Dalal. Investigations into a theory of knowledge base revision (preliminary report). In 7^{th} *National Conference on Artificial Intelligence (AAAI'88)*, volume 2, pages 475–479. Morgan Kaufmann, 1988.

[3] P. Z. Revesz. On the semantics of theory change: Arbitration between old and new information (preliminary report). In 12^{th} *ACM SIGACT-SIGMOD-SIGART Symposium on Principles Of Databases Systems (PODS'93)*, pages 71–82. ACM, 1993.

[4] V. S. Subrahmanian. Amalgamating knowledge bases. *ACM Transactions on Database Systems*, 19:291–331, 1994.

[5] R. Fagin, J. D. Ullman, and M. Y. Vardi. On the semantics of updates in databases. In 2^{nd} *ACM SIGACT-SIGMOD Symposium on Principles Of Database Systems (PODS'83)*, pages 352–365. ACM, 1983.

[6] E. Fermé and S. Hansson. AGM 25 years. twenty-five years of research in belief change. *Journal of Philosophical Logic*, 40:295–331, 2011.

[7] C. Alchourrón, P. Gärdenfors, and D. Makinson. On the logic of theory change: Partial meet contraction and revision functions. *Journal of Symbolic Logic*, 50(2):510–530, 1985.

[8] H. Katsuno and A. Mendelzon. On the difference between updating a knowledge base and revising it. In 2^{nd} *International Conference on Principles of Knowledge Representation and Reasoning (KR'91)*, pages 387–394. Cambridge University Press, 1991.

[9] P. Gärdenfors. *Knowledge in Flux: Modeling the Dynamics of Epistemic States*, volume 103. MIT Press, 1988.

[10] S. O. Hansson. *A Textbook of Belief Dynamics. Theory Change and Database Updating*. Kluwer Academic, 1999.

[11] S. Konieczny and R. Pino Pérez. On the logic of merging. In 6^{th} *International Conference on Principles of Knowledge Representation and Reasoning (KR'98)*, pages 488–498. Cambridge University Press, 1998.

[12] S. Konieczny and É. Grégoire. Logic-based information fusion in artificial intelligence. *Information Fusion*, 7(1):4–18, 2006.

[13] D. Zhang and É. Grégoire. The landscape of inconsistency: a perspective. *International Journal of Semantic Computing*, 5(3), 2011.

[14] Ph. Besnard, É. Grégoire, and S. Ramon. Enforcing logically weaker knowledge in classical logic. In 5^{th} *International Conference on Knowledge Science Engineering and Management (KSEM'11)*, pages 44–55. LNAI 7091, Springer, 2011.

[15] Ph. Besnard, É. Grégoire, and S. Ramon. Overriding subsuming rules. In 11^{th} *European Conference on Symbolic and Quantitative Approaches to Reasoning with Uncertainty (ECSQARU'11)*, pages 532–544. LNAI 6717, Springer, 2011.

[16] Ph. Besnard, É. Grégoire, and S. Ramon. Logic-based fusion of legal knowledge. In *(submitted)*, 2012.

[17] 14^{th} *International Conference on Theory and Applications of Satisfiability Testing (SAT'11)*. LNCS 6695, Springer, 2011.

[18] `http://www.satlive.org`.

[19] É. Grégoire, B. Mazure, and C. Piette. Using local search to find msses and muses. *European Journal of Operational Research*, 199(3):640–646, 2009.

[20] J. McCarthy. Applications of circumscription to formalizing common sense knowledge. *Artificial Intelligence*, 28:89–116, 1986.