

About the Incremental Validation of First-Order Stratified Knowledge-Based Decision-Support Systems

Éric Grégoire, Bertrand Mazure

*CRIL – Université d’Artois
rue de l’Université SP16
F-62307 Lens Cédex France*

Abstract

In this paper, a new efficient technique is introduced to check the logical consistency of first-order function-free stratified knowledge-based decision-support systems (*KBs*). It is based on a progressive instantiation schema that enables us to benefit from the power of local search techniques for propositional satisfiability and search. It combines a knowledge preference pre-ordering with a concept of depth-limited reasoning. An algorithm that proves efficient very often is proposed. It delivers a good approximation of Benferhat et al.’s preferred maximal inclusion-based consistent sub-bases. The approach is extended to cover forms of nonmonotonic *KBs*, as well.

Key words: knowledge bases, validation, verification, logical consistency

1 Introduction

Checking the logical consistency of knowledge-based systems (*KBs*) is a task that is both vital and computationally heavy. Indeed, from any contradictory set of information, a logical agent will be able to deduce any conclusion (and its contrary). Even in the propositional setting, consistency checking can be time-consuming since checking the satisfiability of a set of propositional clauses is NP-complete. However, recent impressive practical progress has been obtained

Email addresses:

gregoire@cril.univ-artois.fr (Éric Grégoire),
mazure@cril.univ-artois.fr (Bertrand Mazure).

thanks to the use of local search techniques, making it possible to check the (in)satisfiability of very large propositional actual knowledge bases [1–5].

This progress is so impressive that it becomes viable to check also the (in)satisfiability of large first-order (finite) KB s by considering their instantiated counterparts [5,6]. Obviously enough, to avoid an exponential size blow-up, carefully restricted instantiation schemata are required. In this context, specific instantiation schemata have been proposed in [6] and shown computationally viable, in particular for checking the consistency of finite clausal (function-free) first-order multi-sorted KB s when they are built in an incremental way, i.e. checking the consistency of $KB' = KB \cup \{f\}$, where KB and $\{f\}$ are consistent finite sets of function-free (multi-sorted) clauses.

In this paper, such a technique is revisited when KB' is stratified according to a knowledge preference pre-ordering. When KB' is inconsistent, the goal is now to give the user a minimal set of clauses that should be retracted from KB so that KB' becomes consistent, while the preference ordering between the information is respected.

Our approach is based on the following heuristic finding: local search is often a good oracle to detect which information is conflicting in a KB , when this latter one is actually inconsistent. More precisely, the clauses that are most often falsified during a failed local search for consistency most likely belong to an inconsistent subpart of KB . The algorithm delivers a good approximation of Benferhat et al. maximal consistent preferred sub-bases [7].

The paper is organized as follows. First, the difficult interaction between the following two tasks is discussed in an intuitive way: on the one hand, we should instantiate KB' step by step and as little as possible, while, on the other hand, in case of inconsistency, we need to detect the preferred minimal set of clauses to be retracted from KB' to restore consistency. A depth-limited instantiation schema for (in)consistency checking is presented. It is then explained how the duality between consistency and inconsistency can play a useful role, before it is shown how the approach can handle forms of nonmonotonic KB s, as well. After some experimental illustrations are given, the main benefits and limits of the approach are then summarized.

2 Incremental instantiation and preference

Assume that KB is a consistent set of first-order clauses and that $\{f\}$ is a clause. Assume that both KB and f contain at least one constant and that they do not contain any non-constant function symbol. Our goal is to prove that $KB' = KB \cup \{f\}$ is consistent or find one smallest set of formulas to be

retracted from KB in order for KB' to be consistent.

Assume also that we have devised an instantiation schema that provides us with increasing partial instantiated knowledge bases from KB' , namely a series of KB_i such that $KB_0 \subseteq \dots \subseteq KB_i \subseteq KB_{i+1} \dots \subseteq KB_{end}$, where KB_{end} is the totally instantiated Herbrand counterpart of KB' (i.e. using all constants occurring in KB').

A first straightforward result is that, for any i , when KB_i is inconsistent then KB' itself is inconsistent. This means that any further instantiation step is useless with respect to inconsistency proving. Moreover, some clauses from KB_i must be dropped (or weakened [8]) if we want consistency to be restored. These results make the progressive instantiation schema attractive. Under the condition that they are guided by the additional formula f , we can hope that some depth-limited instantiation steps will prove the inconsistency (or guarantee that inconsistency cannot occur within this limited depth of possible reasoning).

Now, assume that a preference pre-ordering applies to all clauses from KB' , meaning for instance that, if $(i < j)$, then we prefer to drop clause c_j instead of c_i from KB' in order to recover consistency. Accordingly, clauses from KB' are partitioned into a series of strata. Assume that we have shown that KB_k is inconsistent and that we have found that dropping one of the two clauses c_{124} and c_{235} allows us to restore consistency. Accordingly, we shall drop c_{235} . Imagine now that there are several minimal conflicting sets of clauses in KB_i and that we discover a second set of conflicting clauses, namely $\{c_{43}, c_{124}\}$. In this case, we shall also drop c_{124} . Thus, dropping c_{235} was useless since our preference relation requires us to drop c_{124} , which allows the first set of conflicting information to be solved, too. In other words, we need to handle the whole KB_i , or at least all its minimal inconsistent sets of clauses in order to be sure that the preference is respected. In [9], an algorithm is provided that solves this problem in the propositional framework. In the partial instantiation framework, it should be noted that each time a subsequent KB_{i+1} is considered, then the set of clauses that were previously proposed as clauses to be dropped must be reconsidered. This, in order to make sure that the preference ordering is obeyed.

3 Stratified knowledge bases

Assume that the m clauses in $KB' = KB \cup \{f\}$ are partitioned inside n strata $S_1 \cup \dots \cup S_n$. For simplicity of presentation, we assume that the m clauses are numbered in a way that follows the strata, i.e. when $c_i \in S_k$, $c_j \in S_r$ and $(k > r)$, then $(i > j)$. When $(i > j)$ then c_j is preferred over c_i , i.e. if we must

choose to drop one of the two clauses then we drop c_i . Accordingly, the strata translate a preference complete pre-ordering on the clauses of KB' .

Several approaches have been proposed to characterize the preferred maximal consistent sub-bases of an inconsistent stratified KB' . Here we take Benferhat et al. inclusion-based ordering as a case study [7], which is linked to previous works by [10]. Namely, let $A = A_1 \cup \dots \cup A_n$ and $B = B_1 \cup \dots \cup B_n$ be two consistent subsets of KB' , where $A_i = A \cap S_i$ and $B_i = B \cap S_i$.

Definition 1 [7]

$A \ll_{\{S_1 \cup \dots \cup S_n\}} B$ iff $\exists i$ s.t. $A_i \subset B_i$ and $\forall j < i, A_j = B_j$ (\subset denotes strict inclusion). A consistent sub-base of KB' that is maximal w.r.t. $\ll_{\{S_1 \cup \dots \cup S_n\}}$ is called a maximal consistent sub-base of KB' (w.r.t. $S_1 \cup \dots \cup S_n$).

An equivalent constructive definition can be established easily.

Proposition 1 [7]

$A = A_1 \cup \dots \cup A_n$ is a maximal consistent sub-base of KB' (w.r.t. $S_1 \cup \dots \cup S_n$) iff $A_1 \cup \dots \cup A_i$ is a maximal consistent sub-base of $S = S_1 \cup \dots \cup S_i$, $\forall i \in [1..n]$ in increasing order, successively.

For clarity of presentation, we also recall the dual concept of minimal inconsistent kernel.

Definition 2

A minimal inconsistent kernel (in short, kernel) of KB' is a subset of clauses of KB' that is both inconsistent and minimal with respect to set-theoretic inclusion.

Clearly, KB' might contain several different kernels and their set-theoretic intersection contains f since KB is consistent. Also, dropping one clause belonging to a kernel of KB' is enough to break the kernel, i.e. to suppress it from the set of kernels of KB' .

Definition 3

The inconsistent part of KB' , noted $IP(KB')$, is the set-theoretic union of all kernels of KB' .

In the following, we adopt the natural assumption that any instantiated clause and its parent first-order clause belong to the same stratum. Accordingly, all definitions in this section also apply to any partially instantiated KB_i . We also assume that we do not want to discard f to restore consistency; accordingly f is placed in the lowest stratum.

4 An incremental instantiation schema

Current local search techniques apply to propositional KB s. One straightforward way to tackle finite function-free first-order clausal KB s would require a full instantiation of the KB into its Herbrand counterpart, namely all possible instantiations using the constants occurring in KB . Obviously enough, this cannot be done in the general case without a combinatorial space blow-up. To avoid this drawback as much as possible, some compromises have to be made. Here, we adopt two restrictions. On the one hand, we just consider the restricted problem of incrementally building a consistent KB . More precisely, KB is assumed consistent and a clause f is to be introduced into KB . We want to show that $KB' = KB \cup \{f\}$ is itself consistent or find the involved conflicting information. On the other hand, we make do with a progressive instantiation of KB' inside a propositional setting that ensures that (in)consistency cannot occur using less than a given number i of reasoning steps. From a practical point of view, this policy can be justified by the following assumptions about practical situations; on the one hand, inconsistency is often due to a limited number of rules and facts, limiting the reasoning steps that are necessary to prove it. On the other hand, if the available time-resources are limited, we can make do with a guarantee that inconsistency will not occur within a preset limited depth in the reasoning steps. Let us revisit the instantiation schema that was initially proposed in [6].

Assume KB is a consistent finite set of function-free clauses and that KB contains at least one constant. Assume f is a clause. Let $KB' = KB \cup \{f\}$. As KB is consistent, any inconsistency is due in part from information (constants and/or predicates) from f . Accordingly, we shall expand the information from f inside KB and check for inconsistency step by step.

More formally, let \mathcal{T} be the (non-empty) set of constants occurring in KB' . Actually, a form of multi-sorted logic is required here to avoid a direct combinatorial blow up; namely any constant must be given its exhaustive list of predicate arguments that it can instantiate. For clarity of presentation, this is left implicit in the following. However, let us stress that this is a really important requirement since it will avoid exhaustive instantiations using the whole set \mathcal{T} . Let C_0 be the singleton $\{f\}$. Let KB_0 contain all ground clauses that can be obtained by instantiating clauses from C_0 using \mathcal{T} . From these starting points, we shall augment KB_0 to mimic the possible information links between f and the contents of KB , translating reasoning steps starting from f and using information from KB .

Let us stress that this approach differs from [6] in the sense that we do not put in KB_0 all the ground clauses from KB' , since they might not directly take part in the inconsistency when this latter one exists w.r.t. KB' . Let us

note $Ground(C)$ the set of ground clauses obtained by instantiating clauses in C using \mathcal{T} as instantiation domain. Depth- i consistency/inconsistency is defined in the following way.

Definition 4 [6]

Let C_i be the set of clauses from KB' containing at least one predicate from C_{i-1} . Let $KB_i =_{def} KB_{i-1} \cup \{Ground(C_i)\}$. We say that: KB is depth- i consistent (resp. inconsistent) iff KB_i is consistent (resp. inconsistent).

This concept converges on the standard definitions of consistency and inconsistency.

Proposition 2 [6]

$\exists n$ s.t. KB_n is stable, i.e. $KB_n = KB_{n+1}$. (Such a fixed-point KB_n is noted KB_{end}).

Proposition 3 [6]

KB' is consistent (resp. inconsistent) iff KB_{end} is consistent (resp. inconsistent).

Since $KB_i \subseteq KB_{end}$ for any i , we know that whenever KB_i is inconsistent then KB_{end} is inconsistent. Accordingly, at each instantiation step, when an inconsistency is encountered, it can be reported to the knowledge engineer as no further instantiation step will allow consistency to be recovered. On the contrary, no definitive proof of consistency is obtained until KB_{end} is shown consistent.

5 Handling a partially instantiated KB_i

In this section, we briefly present our previous results [9] in the propositional setting that can be applied to partially instantiated KB_i . In the incremental approach to build a consistent KB , we can hope that the new clause f will not conflict in several minimal ways with the clauses already in KB . Accordingly, we distinguish between two possible situations when KB_i is inconsistent.

In the first one, which we call the single kernel assumption, there is exactly one kernel in KB_i whereas, in the general case, there can be several different kernels in KB_i . To some extent the single kernel assumption can be related to the single diagnosis situation in consistency-based diagnosis [11]. Clearly, the single kernel situation will be encountered more often in our incremental building of a consistent stratified KB than in the process of building the KB globally before restoring consistency only once. The main idea is as follows [9]. In case of inconsistency, dropping one clause can restore consistency. Such

a clause is looked for among the clauses that were more often falsified during the previous failed local search for a model. Candidate clauses are selected according to the preference relation, ensuring that there is no clause that is less preferred and that belongs to $IP(KB_i)$.

In the general case, we cannot assume the existence of a unique kernel when KB_i is inconsistent. In [9], a procedure is described that computes a set of clauses to be discarded to restore consistency and that uses the above heuristic about local search, which proves often accurate. Accordingly, a correct set of clauses to be dropped is found very often. Remarkably enough, our experiments show that this procedure often requires a number of calls to a satisfiability check that is close to the number of kernels of $IP(KB_i)$, which is often a very small number in most realistic situations of incremental construction of KB s. When the heuristic about local search works nicely, this procedure delivers clauses of an increasing preference, each of them allowing at least one kernel to be broken when the clause is dropped.

6 Using the duality between consistency and inconsistency

The above instantiation schema mimics reasoning steps starting from f and linking information from KB . However, it should be noted that it diverges from implementing the standard logical resolution principle in an important way. Each time we include a clause in KB_i because it contains a predicate that also belongs to KB_{i-1} , we do not require that these two occurrences of the same predicate are of opposite signs, as it would be a basic requirement for the resolution principle to apply. This last feature can be exploited in order to exploit the duality between consistency and inconsistency in a better way. Let us explain this in more details.

In the previous section, we have seen that whenever a ground KB_i is shown inconsistent then KB' is itself inconsistent. On the other hand, we can only be sure that KB' is consistent when KB_{end} is reached, which can often be out of reach. But consider the following dual problems. Let KB be a consistent set of clauses, \neg stand for the logical negation connective and f be a clause.

- Is $KB' = KB \cup \{f\}$ consistent?
- Is $KB'' = KB \cup \{\neg f\}$ consistent?

Actually, the above expansion schema yields propositional subsets KB_i and KB_i'' for both problems that differ only on either instantiations of f or instantiations of $\neg f$ are included. Accordingly, we can define a double check policy. At each instantiation level, we test KB_i and KB_i'' for consistency. Whenever one of these checks yields inconsistency, we know that we have got a definitive

answer about the consistency of KB' . In particular, when KB_i'' is inconsistent, that means that KB'' is inconsistent and thus that KB' is consistent.

Accordingly, this is a significant way to soften the requirement asserting that no definitive consistency result can be established about KB' until KB_{end} is reached. On the other hand, from a logical point of view, showing that KB'' is inconsistent amounts to proving that f is an actual logical consequence of KB . Accordingly, when such a situation is encountered, it is worth telling the user that the additional information f to be introduced in KB is redundant w.r.t. KB , at least from a logical point of view.

Let us also stress that switching from KB_i to KB_{i+1} can take advantage of previous computations. Mainly, the initial interpretation selected by the local search algorithm to find a model of KB_{i+1} is not obtained randomly but should be the previously found model of KB_i .

7 Covering forms of nonmonotonic KB s

In many applications, the representation formalism of the KB allows negation by failure or limited forms of default reasoning to be represented. Our local-based search technique is very well-suited to handle forms of negation by failure operators. Imagine that we want to represent the rule asserting that any *resident* must pay *taxes* (except *ambassadors*). This can be represented by $: (resident(x) \wedge \text{not}(ambassador(x)) \Rightarrow taxes(x))$, or in clausal form by $\neg resident(x) \vee \neg \text{not}(ambassador(x)) \vee taxes(x)$, where \wedge , \vee and \Rightarrow represent the standard conjunctive, disjunctive and material implication connectives while “not” represents the negation as failure operator. In intuitive terms, if we can assume consistently that an individual a is not ambassador then $\text{not}(ambassador(a))$ is set to *true*.

We can take advantage of our heuristics about the trace of local search when it fails to find a model, to handle this form of non-monotonic reasoning efficiently, at least very often. The idea is as follows. First, assume that all ground literals that are prefixed by the “not” operator in KB_i are *false*. Accordingly, their truth value is fixed and is not allowed to change during the local search. If this latter search leads to a model, then we were obviously right in ensuring this truth value. Assume now that it fails to give us a model and also that a further complete logically search also fails to find one. Now, we may look at the clauses with the highest scores that contain one atom prefixed by the “not” operator and force the truth value of this atom to *true*. In this way, we satisfy this clause. Very often, running a local search or a complete search on this new KB leads to a model. This approach is discussed more formally and in a variety of nonmonotonic frameworks in [12,13]. The main idea is that

we can start assuming *false* all that is *normally false*; if this fails to deliver a model then the trace of the local search provides us with a good oracle of the assumptions that are wrong, most probably.

8 Experimental illustrations for the nonmonotonic case

As the adequacy and the good performance of the instantiation schema has been discussed elsewhere [6], we shall not repeat it here. Simply, the current size of the instantiated KB_i should not currently overpass 100000 clauses on a standard Pentium PC, which allows many applications to be handled. On the other hand, the depth of reasoning leading to inconsistency should not require an instantiation that leads to a larger number of clauses. Let us thus simply illustrate the extensive tests that we have conducted and of the results that we have obtained by means of a typical example in the nonmonotonic case, under the single kernel assumption. All tests were conducted on a 450 Mhz Pentium III, under Linux, using a straightforward (i.e. non optimized) implementation of a Davis, Logemann and Loveland procedure [14] and TSAT [1,15]. In order to simulate realistic KB s that are difficult w.r.t. consistency checking, we have built new benchmarks starting from DIMACS [16] standard Boolean consistency ones. For instance, we have mixed consistent `ii16` problems with the so-called `aim-50-1_6-yes` and with an inconsistent `Dubois16`, all from [16]. These benchmarks have been rewritten using a shared set of Boolean variables. The generated instances are made of more than 19000 clauses using more than 1500 Boolean variables and are clearly inconsistent. Then, we have introduced additional atomic propositions Ab_i (called *markers* or *abnormality propositions*) representing unexpected exceptions in the clauses in a random fashion, with at most one marker per clause and a ($\#markers / (\#other\ Boolean\ variables)$) ratio being 1/3. These additional atoms are assumed to be interpreted *false* by default and are numbered. The introduction of these propositions was however operated in such a way that new instances were consistent, which was easily established using TSAT [1,15]. These markers simulate the preordering between clauses. They are normally *false*, but when inconsistency occurs, we prefer dropping a clause with a marker of a low index (or assuming that the marker that it contains is *true*). Setting these additional propositions to *false*, we let TSAT run during 30 sec. No model could be found (indeed, such simplified instances are inconsistent). We then assigned the truth value *true* to the marker occurring in the clause with the highest score. TSAT exhibited a model in less than 8 sec. on average, showing that KB is inconsistent in the absence of exceptions, whereas including the exception represented by this marker restores consistency. Thus, under the single kernel assumption, dropping this clause would yield a maximal preferred sub-base, where the underlying ordering is given by the numbering of the markers.

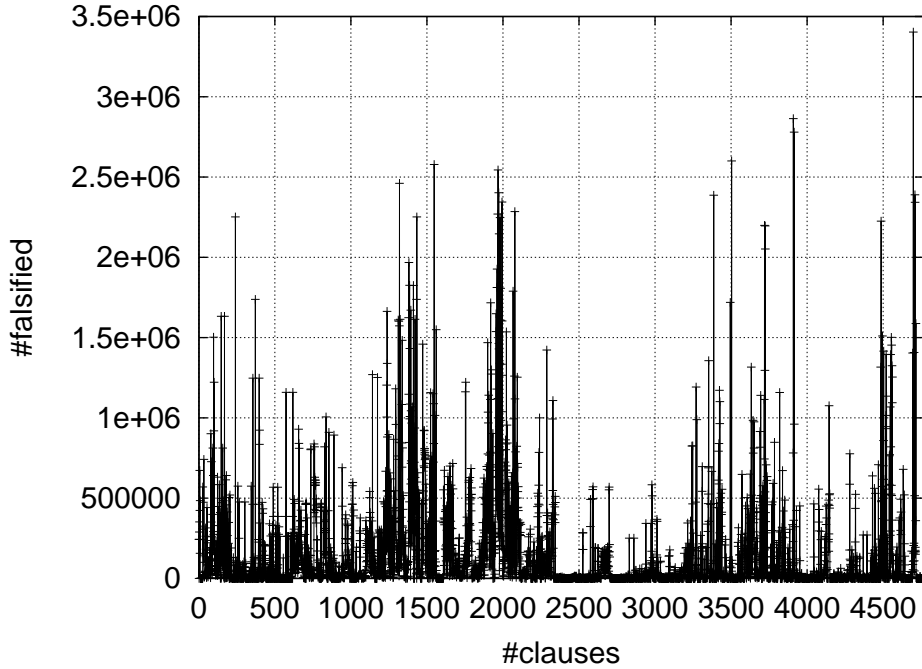


Fig. 1. Trace obtained for **bf2670-130**

More comprehensive experimental results are provided in Figure 1 and Tables 1 & 2. They have been performed on 450 Mhz Pentium III, under linux. We took a series of inconsistent **bf** benchmarks from [16]. Additional Boolean markers Ab_i have been introduced in each benchmark in a random manner, but making sure that they render each benchmark consistent. Assuming that all markers are *false*, the resulting benchmarks are thus becoming inconsistent again. Figure 1 illustrates the number of times each clause has been falsified for the most difficult benchmark (namely **bf2670-130**), during a failed local search for a model, assuming that all markers are *false*. In Tables 1 & 2, all times are expressed in seconds. For two series of **bf** benchmarks, we provide the global time T_{global} that was required to discover the marker that has to be set to *true* to restore consistency, while obeying the preordering preference given by the numbering of the markers. We also give the time T_{trace} spent to obtain the trace allowing us to discover such preferred markers and the time $T_{last-check}$ spent to show that, for the last tested marker, there was no preferable marker that would restore consistency by being set to *true*. Then, we indicate the number $\#Tries$ of markers that we had to consider in a successive manner before finding the most preferred one that restores consistency by being set to *true*. Assuming that all clauses that do not contain a marker are given a high preference (i.e. we do not want to drop them), then the above techniques corresponds to delivering a preferred maximal consistent sub-base of the **bf** benchmarks, under the single kernel assumption. In that respect, the clauses containing markers that are required to be *true* are the clause that should be dropped to obtain these sub-bases.

Instance	$\#Var$	$\#Cla$	$\#Ab_i$	T_{global}	T_{trace}	$T_{last-check}$	$\#Tries$
bf0432-005	1040	3668	178	7.68	0.90	3.21	4
bf0432-007	1040	3668	126	11.50	0.98	5.05	5
bf0432-011	1053	3743	201	8.63	0.94	3.08	9
bf0432-015	1044	3685	143	9.85	0.85	4.38	3
bf0432-017	1044	3685	178	11.87	0.93	5.02	10
bf0432-033	865	2784	106	7.13	0.64	3.09	6
bf0432-035	865	2784	140	8.84	0.68	3.97	4
bf0432-039	864	2790	153	7.55	0.67	3.29	4
bf0432-041	865	2783	144	7.38	0.67	3.17	6
bf0432-043	865	2783	112	8.62	0.63	3.93	2
bf0432-053	1057	3766	186	8.55	1.13	3.60	2
bf0432-070	1057	3761	200	8.21	1.11	3.14	5
bf0432-077	1044	3685	129	11.17	0.92	4.58	12
bf0432-079	1044	3685	113	10.89	0.88	4.52	11
bf0432-091	1057	3761	175	8.08	0.95	3.18	9
bf0432-103	1055	3746	124	7.51	0.92	3.15	3
bf0432-122	1057	3763	197	8.08	0.93	3.12	7
bf0432-135	424	1031	75	4.83	0.22	2.24	4
bf0432-138	421	1000	67	5.02	0.21	2.25	8

Table 1: bf0432 series

Instance	$\#Var$	$\#Cla$	$\#Ab_i$	T_{global}	T_{trace}	$T_{last-check}$	$\#Tries$
bf2670-001	1393	3434	179	7.14	1.03	2.63	9
bf2670-051	1389	3440	256	7.51	1.10	2.94	7
bf2670-052	1389	3440	224	7.26	1.04	2.88	5
bf2670-130	1784	4766	234	16.65	1.65	4.93	1

continued on next page

Instance	$\#Var$	$\#Cla$	$\#Ab_i$	T_{global}	T_{trace}	$T_{last-check}$	$\#Tries$
bf2670-189	1379	3417	189	147.60	0.91	73.10	5
bf2670-202	1387	3439	175	240.34	1.04	119.70	7
bf2670-240	1734	4941	247	365.86	2.15	188.41	5
bf2670-241	1734	4941	249	1605.65	1.80	805.63	12
bf2670-244	1670	4269	191	12443.18	1.30	6104.31	5
bf2670-248	1674	4324	283	1614.16	1.49	815.64	7
bf2670-269	1407	3496	257	6.62	1.03	2.80	1
bf2670-400	1339	3249	156	7914.76	0.90	3991.10	4
bf2670-404	985	2324	142	624.74	0.58	311.39	5
bf2670-468	1488	3859	154	8.43	1.22	3.34	4
bf2670-487	1423	3609	162	8.21	1.03	3.33	5
bf2670-491	1363	3361	159	357.01	1.03	184.08	7
bf2670-492	1363	3361	192	9.99	1.08	3.75	8
bf2670-493	1371	3383	263	462.53	1.01	228.88	5
bf2670-494	1371	3383	192	10.17	0.99	4.44	4
bf2670-501	1734	4941	341	565.30	1.91	284.76	8
bf2670-502	1734	4941	267	1163.69	1.73	566.15	4
bf2670-510	1734	4941	315	1262.59	1.68	624.23	7
bf2670-511	1734	4941	216	48.04	1.73	22.98	5
bf2670-522	1378	3414	142	219.35	1.17	105.79	6
bf2670-524	1386	3436	173	167.72	0.96	83.28	4
bf2670-532	1476	3841	263	175.47	1.21	86.90	8
bf2670-533	1476	3841	169	174.58	1.30	86.60	4
bf2670-538	1668	4712	320	7.52	1.56	2.84	2
bf2670-546	1464	3737	273	60.34	1.19	29.30	6
bf2670-547	1464	3737	224	290.67	1.19	144.64	1

Table 2: bf2670 series

9 Conclusion

In this paper, we have shown that our previous computing techniques [9] for validating stratified propositional KB s can be combined with the partial instantiation technique proposed in [6], in such a way that the consistency of first-order stratified KB s can be checked and maximal consistent sub-bases be delivered (most often). Let us summarise the salient feature of the technique.

- The partial instantiation schema translates reasoning steps. Accordingly, we can ensure the consistency of KB' progressively with respect to an increasing depth of reasoning
- Inconsistency is discovered as soon as possible.
- The most efficient checking techniques for propositional satisfiability are used.
- It can be extended to forms of nonmonotonic KB s.

Acknowledgements

This work has been supported by grants from the “Région Nord/Pas-de-Calais” and by an EC FEDER grant.

References

- [1] B. Mazure, L. Saïs, E. Grégoire, Tabu search for SAT, in: Proceedings of the Fourteenth National Conference on Artificial Intelligence (AAAI'97), Providence (Rhode Island, USA), 1997, pp. 281–285.
- [2] B. Mazure, L. Saïs, E. Grégoire, Boosting complete techniques thanks to local search methods, *Annals of Mathematics and Artificial Intelligence* 22 (1998) 319–331.
- [3] B. Selman, H. J. Levesque, D. Mitchell, Gsat: A new method for solving hard satisfiability problems, in: Proceedings of the Tenth National Conference on Artificial Intelligence (AAAI'92), 1992, pp. 440–446.
- [4] B. Selman, H. A. Kautz, B. Cohen, Local search strategies for satisfiability testing, in: Working notes of the DIMACS Workshop on Maximum Clique, Graph Coloring, and Satisfiability, 1993.
- [5] B. Selman, H. A. Kautz, D. A. McAllester, Computational challenges in propositional reasoning and search, in: Proceedings of the Fifteenth International Joint Conference on Artificial Intelligence (IJCAI'97), Vol. 1, Nagoya (Japan), 1997, pp. 50–54.

- [6] L. Brisoux, L. Saïs, E. Grégoire, Validation of knowledge-based systems by means of stochastic search, *International Journal of Intelligent Systems* 16 (3) (2001) 319–332.
- [7] S. Benferhat, C. Cayrol, D. Dubois, J. Lang, H. Prade, Inconsistency management and prioritized syntax-based entailment, in: *Proceedings of the Thirteenth International Joint Conference on Artificial Intelligence (IJCAI'93)*, 1993, pp. 640–645.
- [8] B. Bessant, E. Grégoire, P. Marquis, L. Saïs, Combining nonmonotonic reasoning and belief revision: a practical approach, in: F. Giunchiglia (Ed.), *Proceedings of the 8th International Conference on Artificial Intelligence Methodology, Systems and Applications (AIMSA'98)*, Vol. 1480 of *Lecture Notes in Computer Science*, Springer, Sozopol (Bulgaria), 1998, pp. 115–128.
- [9] E. Grégoire, Handling inconsistency efficiently in the incremental construction of stratified belief bases, in: A. Hunter, S. Parsons (Eds.), *Proceedings of the Fifth European Conference on Symbolic and Quantitative Approaches to Reasoning and Uncertainty (ECSQARU'99)*, Vol. 1638 of *Lecture Notes in Computer Science*, Springer, London (UK), 1999, pp. 168–178.
- [10] G. Brewka, Preferred subtheories: an extended logical framework for default reasoning, in: *Proceedings of the Eleventh International Joint Conference on Artificial Intelligence (IJCAI'89)*, 1989, pp. 1043–1048.
- [11] R. Reiter, A theory of diagnosis from first principles, *Artificial Intelligence* 32 (1987) 57–95.
- [12] E. Grégoire, B. Mazure, L. Saïs, Logically-complete local search for propositional nonmonotonic knowledge bases, in: I. Niemelä, T. Schaub (Eds.), *Proceedings of the KR'98 Workshop on Computational Aspects of Nonmonotonic Reasoning*, Helsinki University of Technology Research Report, Digital Systems Laboratory, Trento, 1998, pp. 37–45.
- [13] B. Mazure, L. Saïs, E. Grégoire, Checking several forms of consistency in non-monotonic knowledge-bases, in: D. Gabbay, R. Kruse, A. Nonnengart, H. J. Ohlbach (Eds.), *Proceedings of the First International Joint Conference on Qualitative and Quantitative Practical Reasoning (ECSQARU-FAPR'97)*, Vol. 1244 of *Lecture Notes in Artificial Intelligence*, Springer, Bad Honnef (Germany), 1997, pp. 122–130.
- [14] M. Davis, G. Logemann, D. Loveland, A machine program for theorem proving, *Journal of the Association for Computing Machinery* 5 (1962) 394–397.
- [15] B. Mazure, L. Saïs, E. Grégoire, System Description: CRIL Platform for SAT, in: *Proceedings of the Fifteenth International Conference on Automated Deduction (CADE'15)*, Vol. 1421 of *Lecture Notes in Artificial Intelligence*, Lindau (Germany), 1998, pp. 116–119.
- [16] Second Challenge on Satisfiability Testing organized by the Center for Discrete Mathematics and Computer Science of Rutgers University, <http://dimacs.rutgers.edu/Challenges/> (1993).

- [17] S. Benferhat, D. Dubois, H. Prade, How to infer from inconsistent beliefs without revising, in: Proceedings of the Fourteenth International Joint Conference on Artificial Intelligence (IJCAI'95), 1995, pp. 1449–1455.
- [18] E. Grégoire, Overcoming the christmas tree syndrome, in: Proceedings of the Eleventh IEEE International Conference on Tools with Artificial Intelligence (ICTAI'99), IEEE Computer Press, Chicago, 1999, pp. 425–430.
- [19] P. Hansen, B. Jaumard, Algorithms for the maximum satisfiability problem, *Journal of Computing* 22 (1990) 279–303.
- [20] D. A. McAllester, B. Selman, H. A. Kautz, Evidence for invariants in local search, in: Proceedings of the Fourteenth National Conference on Artificial Intelligence (AAAI'97), 1997, pp. 321–326.
- [21] B. Mazure, L. Saïs, E. Grégoire, An efficient technique to ensure the logical consistency of interacting knowledge bases, *International Journal of Cooperative Information Systems* 6 (1) (1997) 27–36.