

# A Comparison of Two Approaches to Inconsistency Detecting

Bertrand Mazure, Lakhdar Saïs, Eric Grégoire  
CRIL – Université d'Artois  
rue de l'Université SP16, F-62307 Lens Cedex, France  
Phone: +33 – (0)3 21 79 32 74, Fax: +33 – (0)3 21 79 32 79 60  
email: {mazure,sais,gregoire}@cril.univ-artois.fr

**ABSTRACT:** Inconsistency detecting is a vital but difficult issue to be addressed when intelligent systems are to be built. Indeed, a system that is provided with deductive capabilities can prove useless in the presence of the smallest contradiction: any conclusion (and its contrary) can be deduced from it under standard logical rules. On the other hand, detecting logical contradictions in large knowledge bases is often considered intractable, even in the basic propositional case. In this paper, we compare two practical approaches to inconsistency detecting in large real-life propositional knowledge bases. The first one is based on a non-standard use of local search whereas the second one is based on more conventional AI techniques that consist of a restricted use of the resolution rule.

## EXTENDED ABSTRACT

A difficult issue to be handled when knowledge-based reasoning systems are built consists in ensuring the absence of contradictory information (or, at least, the correct handling of such information). When the reasoning system is provided with full logical deductive capabilities, this problem is crucial. Indeed, the reasoning system can deduce any conclusion (i.e. both any sentence and its contrary) from the smallest kernel of contradictory knowledge. Unfortunately, detecting inconsistencies in a knowledge base (KB) is extremely expensive from a computational point of view. Even when the knowledge representation language is expressively equivalent to basic propositional logic, this process remains exponential time with respect to the size of the KB (unless  $P = NP$ ). Accordingly, most AI research activities on this topic have concentrated on formalizing forms of reasoning under contradictory information while few algorithms have been developed in order to detect inconsistencies in the context of large real-life systems.

Recently, two practical approaches have been proposed to detect and locate inconsistencies that can be applied in the context of large propositional KBs [Mazure et al. 1995, 1996] [Boufkhad 1996] and that are often surpassing more conventional techniques. More precisely, they can be applied to the problem of localizing small kernels of contradictory information in large deductive propositional KBs. The first one is based on a restricted use of the standard deductive rule of resolution called *bounded resolution*, whereas the second one relies on a heuristic use of local search methods. Let us describe both of them briefly before we compare their respective strengths and limitations.

## BOUNDED RESOLUTION

From a technical point of view, resolution is a basic rule of inference that is complete for propositional logic in the sense that deduction can be achieved by using this rule. Propositional knowledge can be encoded in clausal normal form (or can be reinterpreted as such), namely under the form of a conjunction of disjunctive formulas (i.e. clauses) like

$$\neg(\text{switch1\_on}) \vee \neg(\text{bulb\_ok}) \vee \text{light-on}$$

where  $\neg$  and  $\vee$  represent the standard connectives of negation and inclusive disjunction, respectively.

Now, resolution operates on two clauses to deliver a third one (called the *resolvent*) in the following way: from the clauses  $a \vee \neg b$  and  $b \vee d$ , the new clause  $a \vee d$  is deduced. When an empty clause is obtained, this proves that the initial set of clauses is contradictory. Resolution is the basic principle underlying most automatic reasoning systems, including Prolog.

However, a systematic use of resolution leads to intractability in the worst cases for the full language of propositional logic. In the context of detecting local logical inconsistencies, it has been proposed to restrict its use in the following way [Boufkhad 1996]. Resolvents are generated under the strict condition that their lengths in the number of propositions is lower than the length of each of the two initial clauses. Clearly, such a process which is a form of bounded resolution can be performed in polynomial time with respect to the number of clauses in the KB. Accordingly, this technique has been included as a preprocessing step in C-SAT [Dubois et al. 1996], which is an efficient satisfiability prover that is widely recognized as such. Other variant uses of bounded resolution have been proposed in [Castell 1996] and [Van Gelder and Tsuji 1996].

## LOCAL SEARCH FOR INCONSISTENCY DETECTING

One of the recent results in the AI domain that appear most promising from a practical point of view lies in the discovery of the surprising power of local search [Mitchell et al. 1992] [Selman et al. 1992] [Selman et al. 1993]. Let us discuss it in the context of propositional consistency checking.

A KB is consistent when there exists at least one truth assignment of the propositional variables such that all clauses in the KB are interpreted true (this truth assignment is then called a *model*). In the above example, assigning the variables *switch\_on* and *bulb\_ok* to false and *light\_on* to true, allows the clause to be satisfied. Assigning all three variables to true allows the clause to be satisfied as well. No such assignment can be discovered if and only if the KB contains contradictory knowledge. Accordingly, traditional satisfiability checkers perform a combinatorial search, which enumerates all possible truth assignments in order to find a model. When the whole search space of truth assignments is explored without finding a model, then it can be concluded that the knowledge base is inconsistent.

Local search techniques have been shown as competitive approaches in the problem of showing the existence of models of propositional KBs. Due to their incomplete exploration of the search of the space of truth assignments, they are often believed to be irrelevant to the issue of showing that a KB is inconsistent. Indeed, local search techniques generate an initial interpretation randomly. Then, performing the smallest changes that is most promising, they attempt at finding a model. More precisely, they perform a greedy local search for a satisfying assignment of a set of propositional clauses. The algorithm starts with a randomly generated truth assignment. It then changes (« flips ») the assignment of the variable that leads to the largest increase in the total number of satisfied clauses. Such flips are repeated until either a model is found or a preset maximum number of flips is reached. This process is repeated as needed up to a preset maximum of times.

Recently, we have shown that, contrary to the common and expected belief, local search techniques can also be used to locate inconsistent kernels in KBs [Mazure et al. 1995, 1996].

The following phenomenon can be observed when the algorithm fails to prove that the KB is consistent. The algorithm is traced: for each clause, taking each flip as a step of time, the number of times during which this clause is falsified is updated. A similar trace is recorded for each propositional variable occurring in the knowledge base, counting the number of times it appears in the falsified clauses. Intuitively, it seemed to us that the most often falsified clauses should normally belong to an inconsistent kernel of the knowledge base if this knowledge base is actually inconsistent. Likewise, it seemed to us that the literals that exhibit the highest scores should also take part in this kernel. Actually, these hypotheses prove experimentally correct extremely often.

Strangely enough, it appears thus that the trace of the local search algorithms delivers the probable inconsistent kernel of *locally* inconsistent problems. The most straightforward use of the above discovered feature to solve locally inconsistent problems is the following one. Use local search algorithms to circumscribe the probable inconsistent kernel. Then, apply logical complete techniques to this kernel to prove its unsatisfiability and thus, consequently, the inconsistency of the global problem. The scores delivered by the local search algorithm can be used to guide the branching strategy performed by the complete technique, by selecting the variables with the highest scores first [Mazure et al. 1996].

Obviously, this simple schema can only be used when the discovered probable inconsistent kernel is of manageable size and when the gap between the score of the clauses of the kernel and the score of the remaining clauses is large enough. Also, the clauses can be sorted according to their decreasing scores; incremental complete techniques can be applied on them until unsatisfiability is proved. In this respect, clauses outside the discovered kernel could also be taken into account.

This technique has been experimented extensively for very large propositional KBs and appears extremely competitive. Most often, small inconsistencies within KBs containing more than 10 000 clauses, which do not need to be Horn ones, can be detected within a few seconds CPU time on conventional personal computers.

## COMPARISON OF BOTH TECHNIQUES

In the full paper, we compare both approaches with respect to several types of KBs. First, we show that none of both techniques can detect all inconsistencies within a reasonable amount of time (otherwise, we would have that  $P = NP$ ). We show that the main limitation of the local search approach lies in the ratio of the size of the inconsistent kernel with the size of the KB. We claim that in many real-life domains, inconsistency is most often due to small dispersed sets of pieces of contradictory information. Another advantage of this approach to inconsistency detecting lies in the fact that, at the same time, it keeps the power of local search to show the consistency of a KB when this one is consistent. On the other hand, we show that bounded resolution is by definition limited to local inconsistencies of a specific form that do not cover all usual types of real-life inconsistencies. Mainly, when a contradiction requires longer resolvents to be generated, then it is useless. Unfortunately, such inconsistencies do occur. Finally, we compare the respective performances of both approaches with respect to problems that can be addressed by both techniques.

## (Some) REFERENCES

- Boufkhad, Y. 1996. Aspects probabilistes et algorithmiques du problèmes de satisfiabilité. PhD thesis, Univ. Paris VI.
- Castell, T. 1996. Résolution arrière dans la procédure de Davis and Putnam. R.F.I.A.'96, Rennes, France, pp. 109 – 117.
- Davis, M.; Putnam, H. 1960. A Computing Procedure for Quantification Theory. J.A.C.M. vol. 7, pp. 201 – 215.
- Dubois, O.; André P.; Boufkhad, Y.; Carlier, J. 1996. SAT vs. UNSAT. Second DIMACS Implementation Challenge, DIMACS Series in Discrete Mathematics and Theoretical Computer Science, American Mathematical Society, pp. 415 – 436.
- Mazure, B.; Saïs, L.; Grégoire, E. 1995. TWSAT: a New Local Search Algorithm for SAT. Performance and Analysis. C.P.'95 Workshop on Studying and Solving Really Hard Problems, Cassis, France, pp. 127 – 130.
- Mazure, B.; Saïs, L.; Grégoire, E. 1996. Detecting logical inconsistencies. AI and Maths Symposium, Fort Lauderdale/FL, USA, pp. 116–121.
- Mazure, B.; Saïs, L.; Grégoire, E. 1996. SUN: a Multistrategy Platform for SAT. First Int. Competition and Symposium on Satisfiability Testing, Beijing, China.
- Mitchell, D.; Selman, B.; Levesque, H. 1992. Hard and Easy Distributions of SAT Problems. A.A.A.I.'92, San Jose/CA, USA, pp. 459–465.
- Selman, B.; Levesque, H.; Mitchell, D. 1992. A New Method for Solving Hard Satisfiability Problems. A.A.A.I.'92, San Jose/CA, USA, pp. 440–446.
- Selman, B; Kautz, H.A.; Cohen, B. 1993. Local Search Strategies for Satisfiability Testing . DIMACS Workshop on Maximum Clique, Graph Coloring, and Satisfiability.
- Van Gelder, A.; Tsuji, Y.K. 1996. Satisfiability Testing with More Reasoning and Less Guessing. Second DIMACS Implementation Challenge, *DIMACS Series in Discrete Mathematics and Theoretical Computer Science*, American Mathematical Society, pp. 559–586.