

This paper appears in the *Proceedings of the International Conference on Information Processing and Management of Uncertainty in Knowledge-Based Systems (IPMU'96)*, Granada, Spain, 1996.

A Powerful Heuristic to Locate Inconsistent Kernels in Knowledge-Based Systems *

Mazure Bertrand

CRIL
Université d'Artois
rue de l'Université SP 16
F-62307 Lens Cedex
France
mazure@lens.lifl.fr

Saïs Lakhdar

CRIL - IUT de Lens
Université d'Artois
rue de l'Université SP 16
F-62307 Lens Cedex
France
sais@lens.lifl.fr

Grégoire Eric

CRIL
Université d'Artois
rue de l'Université SP 16
F-62307 Lens Cedex
France
gregoire@lens.lifl.fr

Abstract

A fundamental problem in logical knowledge-bases lies in inconsistency handling. Indeed, any (even local) logical contradiction makes the knowledge-base wholly inconsistent: any piece of information (and its contrary) can be deduced from it under complete rules of deduction. Unfortunately, there is no universal practical way to detect and/or handle such inconsistencies even in the general propositional case since SAT, i.e. checking the satisfiability of a boolean formula, is NP-complete. Very recently, we have discovered a powerful heuristic allowing one to locate and prove inconsistent kernels in propositional knowledge bases. Based on the use of local search methods, this heuristic appears to be surprisingly efficient and opens new perspectives with respect to many artificial intelligence domains. In this paper, its efficiency is illustrated in the context of large propositional knowledge-bases and specifications of complex VLSI system.

Keywords

inconsistency handling, logical inconsistency, SAT, local search methods

* This work has been supported by the Ganymède II project of the Contrat de Plan Etat/Nord-Pas-de-Calais, by the PRC-GDR Intelligence Artificielle, and by the IUT de Lens.

1 Introduction

Inconsistency handling is a ubiquitous problem in artificial intelligence. In particular, it is a serious issue in logical- oriented knowledge-bases (KBs). Indeed, any piece of contradictory information in a logical KB makes it wholly inconsistency under complete rules of deduction. Unfortunately, checking logical consistency is a time-consuming operation that is often out of reach for real-life applications. Even in the basic propositional case, this is an NP-complete problem, making the existence of a universal polynomial-time solution extremely improbable (unless $P = NP$). Accordingly, standard techniques for checking logical satisfiability are based on exponential search techniques that cannot address real KBs.

Recently, there has been a renewal of interest in the study of the propositional satisfiability issue. Several authors have shown how very basic search mechanisms prove to be surprisingly efficient in dealing with large and hard sets of propositional formulas (see mainly [17, 18]). These search mechanisms are however incomplete in the sense that they do not address the complete space of interpretations. Accordingly, they are logically incomplete in the sense that they can prove that a KB is consistent (by exhibiting a model of it), but they are unable to prove in a definitive manner that a KB is inconsistent.

In [13, 14], it is shown that, however, such logically incomplete techniques can underly a very powerful heuristic allowing inconsistent kernels to be located in KBs. Moreover, this heuristic can be used in guiding the search of standard complete techniques for checking consistency, making them efficient for many classes of large and very hard problems. In this paper, we illustrate these results described in [13, 14] by showing their very positive performance with respect to the problem of localizing inconsistent kernels in large KBs and logical specifications of complex VLSI systems.

The paper is organized as follows. First, some technical background is given about SAT, i.e. the problem of checking the satisfiability of a propositional formula. Local search techniques for SAT are then described briefly. Useful dual concepts of local and global inconsistencies are introduced. A heuristic allowing one to locate inconsistent kernels in KBs is then explained and illustrated in the context of several classes of application domains. In the conclusion, we emphasize the relevance and perspective of these results with respect to various artificial intelligence domains.

2 SAT and Local Search Techniques

SAT consists in checking the satisfiability (i.e. consistency) of a propositional formula in conjunctive normal form (CNF). Let us recall here that any propositional formula can be translated thanks to a linear time algorithm in CNF, equivalent with respect to SAT. A CNF formula is a set (interpreted as a conjunction) of clauses, where a clause is a disjunction of literals. A literal is a positive or negated propositional variable.

An interpretation of a boolean formula is an assignment of truth values to its variables. A model is an interpretation that satisfies the formula. Accordingly, SAT consists in finding a model of a CNF formula when such a model does exist or in proving that such model does not exist.

Recently, there has been a renewal of interest in designing efficient methods for hard SAT problems.

On the one hand, several authors have improved logically complete techniques like Davis and Putnam procedure DP [4] (e.g. CSAT [6]). However, these techniques remain of a limited practical scope since they do not allow one to manage large and hard SAT problems.

On the other hand, logically incomplete techniques based on local search have been shown particularly efficient in proving large and hard consistent problems. Let us now briefly recall one of these methods, namely Selman et al.'s GSAT algorithm [17, 18]. This algorithm performs 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 (MAX-FLIPS) is reached. This process is repeated as needed up to a maximum of MAX-TRIES times.

In the sequel, TWSAT, i.e. a more recent and efficient variant of GSAT, will be considered. TWSAT (as Taboo Walk Strategy for SAT) [12], departs from basic GSAT by making a systematic use of a taboo list of variables in order to avoid recurrent flips and thus escape local minima. More precisely, TWSAT keeps a fixed length -chronologically-ordered FIFO- list of flipped variables and prevents any of the variables in the list from being flipped again during a given amount of time.

These very simple logically incomplete algorithms, which belong to the

```

Procedure GSAT
Input : a set of clauses S, MAX-FLIPS, and MAX-TRIES
Output : a satisfying truth assignment of S, if found
Begin
  for i := 1 to MAX-TRIES
    I := a randomly generated truth assignment
    for j := 1 to MAX-FLIPS
      if I satisfies S then return I
      x := a propositional variable such that a change
        in its truth assignment gives the largest
        increase in the number of clauses1
        of S that are satisfied by I
      I := I with the truth assignment of x reversed
    end for
  end for
  return "no satisfying assignment found"
End

```

Figure 1: GSAT Algorithm: basic version

local search procedures family, are surprisingly efficient in demonstrating that CNF formulas are consistent.

3 Local vs. Global Inconsistencies

First, let us define some dual concepts of local and global inconsistency for propositional KBs [14].

Definition 1

A SAT instance *S* is *globally inconsistent*
iff
S is inconsistent and $\forall S' \subset S : S'$ is consistent.

¹this number can be negative

When an inconsistent SAT instance S is not globally inconsistent, it is said to be locally inconsistent.

Definition 2

A SAT instance S is *locally inconsistent*
iff
 S is inconsistent and $\exists S' \subset S : S'$ is inconsistent

Clearly, several measures of locality for inconsistency can be defined, making use of e.g. the (size of the inconsistent kernel)/(size of theKB) ratio. In this paper, we focus on locally inconsistent KBs. Indeed, such a form of inconsistency is of prime importance in actual applications. Very often, inconsistency is due to the accidental presence of a few pieces of contradictory information about a given subject. Obviously, globally inconsistent problems are the most difficult to handle; they are often generated in an artificial manner since they are scarce in real life applications.

4 An efficient Heuristic to Locate Inconsistencies

In this section, a somewhat surprising finding is presented: GSAT-like techniques can be used to localize inconsistent kernels of propositional KBs, although the scope of such logically incomplete algorithms was normally expected to concern consistent problems, only.

The following test² has been repeated very extensively, giving rise extremely often to the same result. TWSAT (or other GSAT-like algorithms) is run on a SAT problem. The following phenomenon is encountered when the algorithm fails to prove that the problem is consistent. TWSAT is traced and, 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 also done for each literal occurring in the SAT problem, counting the number of times it has appeared in the falsified clauses. Intuitively, it seemed to us that the most often falsified clauses should normally belong to an inconsistent kernel of the SAT problem if

²all our experimentations have been conducted on i486 DX2 PCs.

this problem is actually inconsistent. Likewise, it seemed to us that the literals that exhibit the highest scores should also take part in this kernel.

Actually, this hypothesis proved to be most of the time experimentally correct (except for globally inconsistent problems). This phenomenon can be summarized as follows. When GSAT-like algorithms are run on a locally inconsistent SAT problem, then the above counters allow us to split the SAT problem into two parts: a consistent one and an unsatisfiable one. A significant gap between the scores of two parts of the clausal representation is obtained, differentiating a probable inconsistent kernel from the remaining part of the problem. Strangely enough, it appears thus that the trace of GSAT-like algorithms delivers the probable inconsistent kernel of locally inconsistent propositional KBs. We are currently analyzing how the form and properties of this trace can be experimentally related to graded notions of locality for inconsistency.

In Figure 2, we show the scores of the literals and of the clauses when applying TWSAT to a large complex VLSI problem taken from a standard benchmark (i.e. the so-called bf1355-638 problem consisting of 6768 clauses and 2177 propositional variables) [5]. TWSAT failed to prove that this KB is consistent; this problem is also clearly out of reach of usual techniques (applying improved DPs techniques, no answer was obtained within a preset 30 hours CPU time). However, the trace of TWSAT shows us the probable existence of an inconsistent kernel. Actually, using this information, we have been able to prove the inconsistency of this KB in 584 seconds and point out the inconsistent kernel.

Let us now illustrate the performance of this heuristic in the problem of detecting inconsistencies when merging individually consistent propositional KBs. In Figure 3, the scores of the above counters are given when TWSAT is applied to a KB resulting from the union of two large initial ones (each initial KB containing 6500 clauses (with a number of literals per clause ranging from 2 to 4) referring to 5000 different variables occurring in both KBs). Each initial KB was shown consistent using TWSAT (in less than 2 seconds). TWSAT failed to prove that the merged KB is consistent. However, the trace of TWSAT showed us that a few (5) clauses exhibit a very high score. Some of them belonging to the first initial KB while the others belonging to the second one. We then applied to the subKB formed with these clauses a standard complete method based on Davis and Putnam procedure to show that it was actually

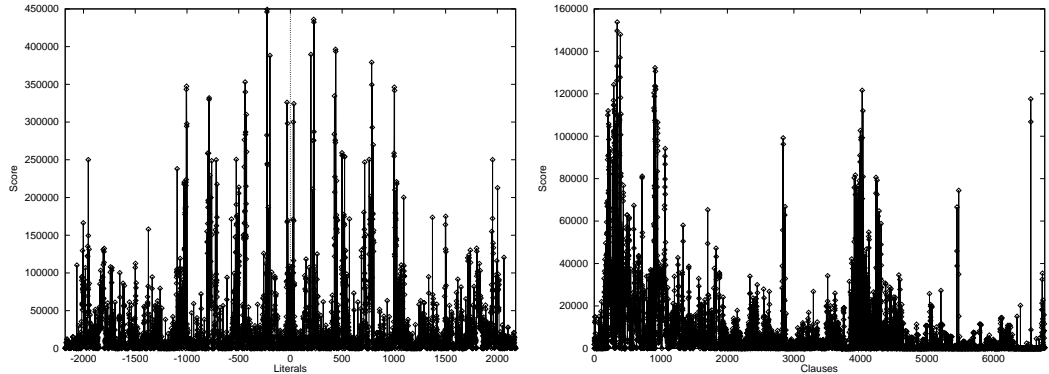


Figure 2: An example in the VLSI domains

inconsistent (this took 0.01 second). This subKB is thus an inconsistent kernel causing the inconsistency of the whole merged KB. Retracting this kernel from the merged KB, we managed to prove (in less than 11 seconds) using TWSAT that the filtered KB is consistent. On the other hand, we have run several of the best optimized versions of standard complete techniques (e.g. CSAT [6]) on the inconsistent merged KB but obtained no result within a 24 H preset CPU time.

This experimentation has been repeated very extensively and has given rise to this phenomenon extremely often. By itself the discovery of a practical way to locate probable inconsistent kernels is a very positive result. Indeed, we can now implement techniques addressing these kernels in several ways. A first-one could simply consist in rejecting the probable inconsistent kernel from the KBs. It could also consist in requesting a human agent to check it and, when needed, correct the pieces of interacting knowledge that seemed to cause inconsistency.

More ambitiously, we might want the system to prove formally that this probable inconsistent kernel is actually inconsistent. As for the above example, we might apply standard techniques to the discovered kernel in order to prove its inconsistency. 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.

However, exploiting further the above heuristic, we have managed to de-

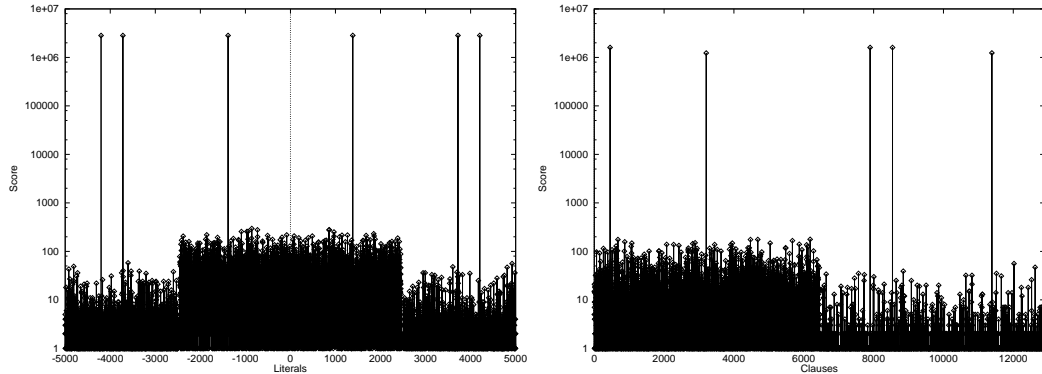


Figure 3: Scores of large inconsistent KBs.

velop a new complete method to prove inconsistency that outperforms the best standard ones very often [13, 14]. This method can be described as follows. We apply TWSAT to select the next literal to be assigned the truth-value true by DP. This literal is selected as the one with the highest score as explained above. Such a technique can be interpreted as using the trace of TWSAT as an heuristic for selecting the next literal to be assigned true by DP, and a way to extend the partial assignment made by DP towards a model of the KB when this KB is satisfiable.

5 Conclusions and Perspectives

Clearly, the heuristic presented in this paper could open new perspectives with respect to several artificial intelligence domains.

- As we have illustrated it in this paper, it provides us with a practical way to detect inconsistent kernels in large propositional KBs.
- As described in [13, 14], this can lead to new performant theorem proving techniques. Indeed, showing that a conclusion can be deduced from some premises is equivalent to showing that the negated conclusion together with the premises is inconsistent. Moreover, we have shown that the heuristic described in this paper can boost standard semantic-based theorem proving techniques by guiding their search.

- Inconsistency handling is ubiquitous and implicit in many forms of non-monotonic reasoning. Indeed, many patterns of non-monotonic reasoning are formed or can be implemented with the schema “Since it is consistent to assume that...”.
- Most theories about (logical) belief revision and reasoning under inconsistent logical knowledge should benefit from practical ways to locate and prove inconsistency.

References

- [1] P. Cheeseman, B. Kanefsky, W.M. Taylor (1991). Where the Really Hard Problems are. *Proc. IJCAI-91*, pp. 163-169.
- [2] V. Chvátal, E. Szemerédi (1988). Many Hard Examples for Resolution. *Journ. of the ACM*, vol. 33, no. 4, pp. 759-768.
- [3] S. Cook (1976). A Short Proof of the Pigeon Hole Principle Using Extended Resolution. *SIGACT News*, vol. 8, pp. 28-32.
- [4] M. Davis, H. Putnam (1960). A Computing Procedure for Quantification Theory. *Journ. of the ACM*, vol. 7, pp. 201-215.
- [5] DIMACS (1993). Second challenge organized by the Center for Discrete Mathematics and Computer Science of Rutgers University.
- [6] O. Dubois, P. André, Y. Boufkhad, J. Carlier (1993). SAT versus UNSAT. *Journ. of the Am. Mathematical Society* (submitted). also in *Proc. of the Second DIMACS Challenge*.
- [7] O. Dubois, J. Carlier (1991). Probabilistic Approach to the Satisfiability Problem. *Theoretical Computer Science*, vol. 81, pp. 65-75.
- [8] J. Franco, M. Paull (1983). Probabilistic Analysis of the Davis and Putnam Procedure for Solving the Satisfiability Problem. *Discrete Applied Math.*, vol. 5, pp. 77-87.
- [9] I.P. Gent, T. Walsh (1993). Towards an Understanding of Hill-climbing Procedures for SAT. *Proc. AAAI-93*, pp.28-33.

- [10] I.P. Gent, T. Walsh (1994). The SAT Phase Transition. *Proc. ECAI-94*, pp. 105-109.
- [11] K. Iwama, E. Miyano (1993). Test-Case Generation with Proved Securities. *Proc. 1993 DIMACS Workshop on Maximum Clique, Graph Coloring, and Satisfiability*.
- [12] B. Mazure B., L. Saïs, E. Grégoire (1995). TWSAT: a New Local Search Algorithm for SAT. Performance and Analysis. *Proc. CP-95 Workshop on Studying and Solving Really Hard Problems*, pp. 127-130.
- [13] B. Mazure, L. Saïs, E. Grégoire (1995). Boosting Complete Techniques thanks to Local search Techniques. (submitted)
- [14] B. Mazure, L. Saïs, E. Grégoire E. (1996). Detecting Logical Inconsistencies. *Proc. AI/MATH-96*, pp. 116-121.
- [15] D. Mitchell, B. Selman, H. Levesque (1992). Hard and Easy Distributions of SAT Problems. *Proc. AAAI-92*, pp. 459-465.
- [16] A. Rauzy (1994). On the Complexity of the Davis and Putnam's Procedure on Some Polynomial Sub-Classes of SAT. *LaBRI Technical Report 806-94*, Université de Bordeaux 1.
- [17] B. Selman, H. Levesque, D. Mitchell (1992). A New Method for Solving Hard Satisfiability Problems. *Proc. AAAI-92*, pp. 440-446.
- [18] B. Selman, H.A. Kautz, B. Cohen (1993). Local Search Strategies for Satisfiability Testing. *Proc. 1993 DIMACS Workshop on Maximum Clique, Graph Coloring, and Satisfiability*.
- [19] G.S. Tseitin (1968). On the Complexity of Derivations in Propositional Calculus. in: Slisenko A.O. (ed.), *Structures in Constructive Mathematics*, Part II, pp. 115-125.