

Checking Several Forms of Consistency in Non-Monotonic Knowledge-Bases^{*}

Bertrand Mazure, Lakhdar Saïs and Éric Grégoire

CRIL – Université d'Artois
rue de l'Université SP 16
F-62307 Lens Cedex, France
{mazure,sais,gregoire}@cril.univ-artois.fr

Abstract. In this paper, a new method is introduced to check several forms of logical consistency of nonmonotonic knowledge-bases (KBs). The knowledge representation language under consideration is full propositional logic, using "Abnormal" propositions to be minimized. Basically, the method is based on the use of local search techniques for SAT. Since these techniques are by nature logically incomplete, it is often believed that they can only show that a formula is consistent. Surprisingly enough, we find that they can allow inconsistency to be proved as well. To that end, some additional heuristic information about the work performed by local search algorithms is shown of prime practical importance. Adapting this heuristic and using some specific minimization policies, we propose some possible strategies to exhibit a "normal-circumstances" model or simply a model of the KB, or to show their non-existence.

1 Introduction

Assume that a knowledge engineer has to assemble several logic-based propositional knowledge modules, each of them describing one subpart or one specific view of a complex device. In order to make fault-diagnosis possible in the future, each knowledge module describes both normal functioning conditions and faulty ones. To this end, the ontology is enriched with McCarthy's additional "Abnormal" propositions (noted Ab_i) [7] allowing default rules to be expressed together with faulty operating conditions. For instance, the rule asserting that, under normal circumstances, when the switch is on then the lights should be on is represented by the formula $switch_on \wedge \neg Ab_1 \Rightarrow lights_on$, and in clausal form by $\neg switch_on \vee Ab_1 \vee lights_on$. In this very standard framework, Ab_i propositions are expected to be **false** under normal operating circumstances of the device. The knowledge based system (KB) is expected to be used in a nonmonotonic way, in the sense that conclusions can be inferred when they are satisfied in some preferred models of the KB where Ab_i are **false**. Also, model-based diagnosis

^{*} This work has been supported by the Ganymède II project of the "Contrat de Plan Etat/Nord-Pas-de-Calais".

[9] can be performed in order to localize faulty components in the presence of additional factual data.

The specific issues that we want to address in this framework is the following one. How can the knowledge engineer check that the global KB is consistent? Also, how can he exhibit (or show the non-existence of) one model of the KB translating normal functioning conditions of the device? We would like these questions to be answered for very large KBs and practical, tractable, methods be proposed.

Actually, consistency checking is a ubiquitous problem in artificial intelligence. First, deduction can be performed by refutation, using inconsistency checking methods. Also, many patterns of nonmonotonic reasoning rely on consistency testing in an implicit manner. Moreover, ensuring the logical consistency of logical KBs is essential. Indeed, inconsistency is a serious problem from a logical point of view since it is global under complete (standard) logical rules of deduction. Even a simple pair of logically conflicting pieces of information gives rise to global inconsistency: every formula (and its contrary) can be deduced from it. This problem is even more serious in the context of combining or interacting several knowledge-based components. Indeed, individually consistent components can exhibit global inconsistency, due to distributed conflicting data.

Unfortunately, even in the propositional framework, consistency checking is untractable in the general case, unless $P = NP$. Indeed, SAT (i.e., the problem of checking the consistency of a set of propositional clauses) is NP-complete. Recently, there has been some practical progress in addressing hard and large SAT instances. Most notably, simple new methods [11] that are based on local search algorithms have proved very efficient in showing that large and hard SAT instances are consistent. However, these methods are logically incomplete in that they cannot prove that a set of clauses is inconsistent since they do not consider the whole set of possible interpretations.

However, we have discovered the following phenomenon very recently [6]. When we trace the work performed by local search algorithms when they fail to prove consistency, we extract a very powerful heuristic allowing us to locate probable inconsistent kernels extremely often. Accordingly, we proposed a new family of powerful logically complete and incomplete methods for SAT.

In this paper, we extend this previous work in order to address nonmonotonic propositional KBs, using the above "Abnormal" propositions that are expected to be false under normal circumstances. Using this preference for normal conditions, we guide the local search towards a possible "normal circumstances" model. When such a model is not found, several issues can be addressed. First, using the above heuristic and assuming that normal circumstances are satisfied, we propose a technique that allows us to prove (very often) the absence of such a model and to exhibit an inconsistent kernel. Then, dropping the special status of "Abnormal" propositions to several possible extent, we introduce various strategies for showing the consistency or inconsistency of the KB.

The paper is organized as follows. First, we recall some background about SAT, local search methods and our specific knowledge representation language.

Then, we explain how the trace of local search algorithms can give rise to a powerful heuristic to locate inconsistent kernels. We then propose a family of logically complete and incomplete techniques for SAT, using a specific policy towards "Abnormal" propositions. More precisely, before relating some experimental results about the consistent combination of very large propositional KBs, we discuss some possible strategies to exhibit a normal- circumstances model or simply a model of the KB, or show their non-existence.

2 "Abnormal" propositions, SAT and local search techniques

The knowledge representation language under consideration in this paper is full propositional logic, using a finite set of propositional variables and standard logical connectives. Knowledge is assumed to be represented in clausal form. As explained in the introduction, default knowledge is represented using additional "Abnormal" propositional variables Ab_i , allowing faulty components or behaviours to be represented. Let us stress that we make a very specific use of Ab_i variables. They are intended to represent unexpected faulty conditions of the device. This means that when the KB is assembled, they could be consistently assumed **false**. This is a restricted use in the sense that we do not represent default rules where possible exceptions are expected to exist in the initial KB, like "Typical humans are right-handed".

SAT is the problem of checking whether a set of propositional clauses is consistent, i.e. whether there exists an interpretation assigning values from {**true**, **false**} to the propositional variables so that the clauses are **true** under standard compositional rules of interpretation. Such an interpretation is called a *model* of the SAT instance.

A *normal-circumstances model* of a SAT instance is defined in this paper as a model of the SAT instance where all Ab_i are interpreted false.

Although SAT is NP-complete, there has been significant progress in showing the consistency of very large and hard SAT instances, using very simple local search techniques. Let us simply recall here the most representative one, namely Selman et al.'s GSAT algorithm [11, 12]. This algorithm (*cf.* Fig. 1) 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 following, we shall use a variant of GSAT, namely TSAT [5], a local search technique using a tabu list forbidding recurrent flips. Let us stress that all results presented in the following Sections keep holding for most members of the family of GSAT-like algorithms.

```

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 do
    I := a randomly generated truth assignment
    for j := 1 to MAX-FLIPS do
      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 clauses2 of S that are satisfied by I
      I := I with the truth assignment of x reversed
    done
  done
  return "no satisfying assignment found"
End

```

Fig. 1. GSAT Algorithm: basic version

3 A heuristic to locate inconsistent kernels

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

The following test³ has been repeated very extensively, giving rise to the same result extremely often [6]. TSAT (or any other GSAT-like algorithms) is run on a SAT instance. The following phenomenon is encountered when the algorithm fails to prove that the instance is consistent. TSAT 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 recorded 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 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 most of the time experimentally correct. This phenomenon can be summarized as follows. When GSAT-like algorithms are run on a locally inconsistent SAT problem (i.e. on a SAT problem whose smallest inconsistent kernel is a small subpart of it), then the above counters allow us to split the SAT problem into two parts: a consistent one and an unsatisfiable one.

² this number can be negative

³ all our experimentations have been conducted on a 166 Pentium PC.

A significant gap between the scores of two parts of the clausal representation are 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.

Let us stress that the validity of this experimental result depends on the size of the discovered probable kernel with respect to the size of the SAT instance. When the SAT instance tends to be globally inconsistent, i.e. when the size of the smallest inconsistent kernel converges towards the size of the SAT instance, no significant result is obtained. Fortunately, most real-life inconsistent KBs exhibit some small kernels of conflicting information.

We have proposed and experimented several ways to use the above heuristic information to prove inconsistency in a formal way [6]. The most straightforward one consists in using GSAT-like algorithms to detect the probable inconsistent kernel. Then, complete techniques like Davis and Putnam procedure (in short DP) [1] can be run on this kernel to prove its unsatisfiability and thus, consequently, the inconsistency of the global problem. Also, we can sort the clauses according to their decreasing scores and use incremental complete techniques on them until unsatisfiability is proved. In this respect, clauses outside the discovered probable kernel could also be taken into account.

More generally, the scores delivered by the GSAT-like algorithm are used to guide the search performed by the complete technique. For instance, the following (basic) procedure (*cf.* Fig. 2) combining DP with our heuristic proves extremely competitive. TSAT is run to deliver 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 an approach can be seen as using the trace of TSAT as a heuristic in order to select 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 SAT instance when this instance is satisfiable. Each time DP needs to select the next variable to be considered, such a call to TSAT can be performed with respect to the remaining part of the SAT instance.

4 Using "Abnormal" propositions to guide local search

Assume now that several knowledge modules need to be combined. Each module represents the knowledge about e.g. one device component or one specific view of it. "Abnormal" propositions are used to represent normal functioning circumstances and it is expected that they can be consistently interpreted as **false**. Let us make the global consistency problem more precise in this framework.

For the moment, let us assume that each module is consistent. The technique we shall develop can be used to check this basic assumption, too.

Question 1. Can we prove that the global system combining all the modules is consistent? In the negative case, can we locate the conflicting distributed information?

```

Procedure DP+TSAT(S)
Input: a set S of clauses.
Output: a satisfying truth assignment of S if found, or a definitive
statement that S is inconsistent.
Begin
  S := Unit_propagate(S)
  if the empty clause is generated then return (false)
  else if all variables are assigned then return (true)
  else begin
    if TSAT(S) succeeds then return (true)
    else begin
      p := the most often falsified literal during TSAT search
      return ( DP+TSAT(S $\wedge$ p)  $\vee$  DP+TSAT(S $\wedge$  $\neg$ p))
    end
  end
End

```

Fig. 2. DP+TSAT Algorithm

The first question is not very satisfactory in our specific framework. Indeed, if we extract one model of the modules where some Ab_i are **true**, this will not inform us about the possible existence (or non-existence) of good (i.e. not abnormal) working conditions for the described device. However, this would be a relevant issue with respect to the consistent combination of several knowledge modules, where Ab_i are used to represent default rules where exceptions are expected to exist in the KB, like "Typical human are right-handed". Focusing on our diagnosis-oriented framework, we shall turn our attention to the following adapted consistency issues.

Question 2. Can we prove that the global system combining all the modules is normal-circumstances consistent in the sense that it exhibits at least one model where all Ab_i are **false**?

In the negative case, we can try to solve the following subsequent issues.

Question 3. Can we formally prove that no normal-circumstances model does exist, and can we locate the information that is conflicting when all Ab_i are assigned **false**?

Question 4. Can we exhibit a model where some Ab_i are **true**?

Question 5. When the answer to *Question 4.* is negative, can we locate the conflicting distributed information?

Let us adapt local search methods to the above problems. Most local search methods generate one initial interpretation randomly. However, addressing *Question 2*, the initial interpretation must be generated so that every Ab_i variable is

assigned false while the truth value of the other variables is still selected randomly. Now, different policies should be observed with respect to the selection of the next variable to be flipped, depending on the Question in hand.

When we address *Question 2*, assigning **false** to Ab_i variables will give rise to a simplified KB. No Ab_i can be allowed to be flipped later. Running standard local search technique can deliver one model, that will obviously be a normal-circumstances model of the KB. When it fails to do so, using our heuristic, we order literals and clauses according to their scores. Clauses with highest scores form a probable inconsistent kernel (assuming all Ab_i are **false**). Running one complete method described in section 3 can solve *Question 3*.

Assume now that we want to address *Question 4*. In this case, we run a local search method, using an initial interpretation where all Ab_i variables are assigned **false** while the truth value of the other variables is still generated randomly. Indeed, we still want to guide the search towards a normal- circumstances model, preferring to exhibit a model where as many Ab_i as possible are **false**. Accordingly, we should reconsider our flipping policy. An extreme solution consists in making no distinction anymore between Ab_i variables and the other ones. We could also pounder Ab_i variables so that they are less easily flipped to **true** than the other ones. Also, we can define several policies about the initial values of Ab_i in the subsequent "tries", depending on e.g. their previous behaviour. Obviously enough, a model that is delivered by the performed local search is not necessarily a normal- circumstances one. If no model is found after a certain amount of time, we use the trace of the search to locate the most often falsified clauses and flipped variables to locate a probable inconsistent kernel. A complete method described in section 3 can be run to show this inconsistency formally and thus answer *Question 5*.

5 Experimental results

Let us now illustrate the power of our heuristic to locate inconsistency in the context of combining large KBs. We have made extensive experimentations on real-life and benchmarks problems and the same phenomenon occurred extremely often.

For example, we took several benchmarks for consistency checking [2] describing electronic devices, namely ssa7552-038, ssa7552-158, ssa7552-159 and ssa7552-160. These KBs involve 3575, 3034, 3032, 3126 (1501, 1363, 1363 and 1391) clauses (variables), respectively. Each of these KBs can be shown consistent using TSAT within 1 second CPU time. We also took bf1355-638 (4768 clauses and 2177 variables), whose consistency status cannot be settled neither by local search techniques neither using the best complete Davis and Putnam's techniques, like C-SAT [3] or DP with Jeroslow and Wang heuristic [4] (more precisely, we gave up after spending more than 24 H CPU time for each of these techniques).

We assembled these five KBs together to yield a global one. Neither local search techniques nor the above complete techniques allowed to settle the con-

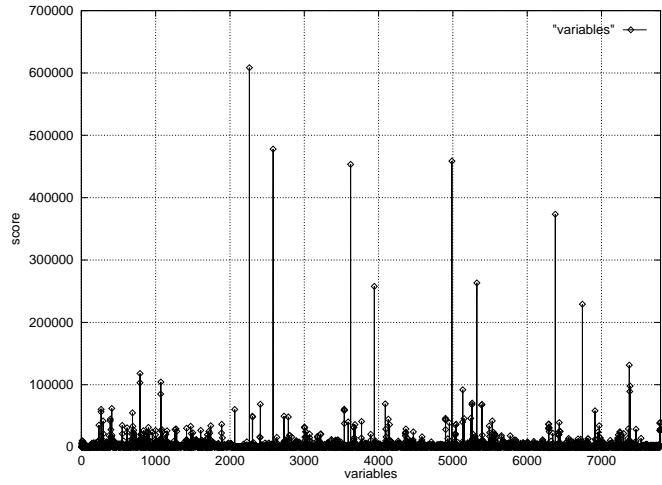


Fig. 3. scores of variables

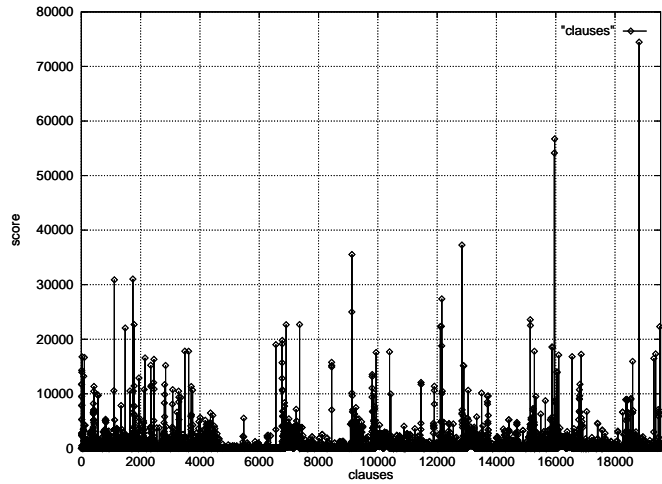


Fig. 4. scores of clauses

sistency status of this KB. Figure 3 gives the scores of the variables (number of times that they have been flipped) and the clauses (Fig. 4) (number of times that they are satisfied) obtained after running TSAT during 8 minutes. We selected the 210 clauses with the highest scores. Running a standard complete Davis and Putnam method on them, we proved the inconsistency of these clauses and thus of the whole KB in 0.03 second. Moreover, using DP+TSAT we proved in 40 seconds that the global KB without this kernel is consistent. Actually, it is possible to prove within 32 seconds CPU time, using DP+TSAT, that, the bf problem is inconsistent.

Let us stress also that this phenomenon is also encountered when variables are shared by the combined KBs and when the inconsistent kernel is really distributed among the components. Introducing Ab_i variables and the above strategies do not alter the above results, neither.

6 Conclusions

The significance of this paper is twofold. First, we have illustrated how local search techniques can be used to prove inconsistency, which was quite unexpected. Moreover, we have illustrated the actual tractability of this approach for very large propositional KBs. Second, this work is a first step towards implementing fast nonmonotonic inference using local search techniques. Indeed, providing a new efficient complete approach for checking consistency can prove useful with respect to the direct implementation of many nonmonotonic system, like e.g. default logic [10]. By generalizing the treatment performed about Ab_i propositions in this paper, local search can be easily fine-tuned with respect to the issue of searching for specific models where some propositions are minimized.

References

1. Davis, M., Putnam, H.: A Computing Procedure for Quantification Theory. *Journ. of the ACM* **7** (1960) 201–215
2. Proc. of the Second DIMACS Challenge on Satisfiability Testing, Rutgers (1993)
3. Dubois, O., André, P., Boufkhad, Y., Carlier, J.: SAT vs. UNSAT, in [2].
4. Jeroslow, R.E., Wang, J.: Solving Propositional Satisfiability Problems. *Ann. Maths and AI* **1** (1990) 167–187
5. Mazure, B., Saïs, L., Grégoire, E.: TWSAT: a New Local Search Algorithm for SAT. Performance and Analysis. CP'95 Workshop on Studying and Solving Really Hard Problems, Cassis, France (1995) 127–130
6. Mazure, B., Saïs, L., Grégoire, E.: Detecting logical inconsistencies. Proc. AI and Maths Symposium, Fort Lauderdale (FL) (1996) 116–121
7. McCarthy, J.: Applications of circumscription for formalizing common-sense knowledge. *Artificial Intelligence* **28** (1986) 89–116
8. Mitchell, D., Selman, B., Levesque, H.: Hard and Easy Distributions of SAT Problems. Proc. AAAI-92 (1992) 459–465
9. Reiter, R.: A theory of diagnosis from first principles. *Artificial Intelligence* **32** (1987) 57–95
10. Reiter, R.: A logic for default reasoning. *Artificial Intelligence* **13** (1980) 81–131
11. Selman, B., Levesque, H., Mitchell, D.: A New Method for Solving Hard Satisfiability Problems. Proc. AAAI-92 (1992) 440–446
12. Selman, B., Kautz, H.A., Cohen, B.: Local Search Strategies for Satisfiability Testing. Proc. DIMACS Workshop on Maximum Clique, Graph Coloring, and Satisfiability (1993)