Logically-complete local search for propositional nonmonotonic knowledge bases

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

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

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

Abstract

In this paper, a new approach to compute nonmonotonic inferences in a simple but useful propositional model-preference formalism is presented. It is original from at least two points of view. First, it makes use of local search techniques while preserving logical completeness. Second, it proves experimentally efficient for an important class of very large nonmonotonic knowledge bases. More precisely, it extends recent SAT-related practical computational results to a nonmonotonic framework. The proof-strategy is based on the use of local search techniques for SAT together with an efficient heuristic when these techniques fail to deliver a model. It is applied to a formalism allowing prioritized rules of default reasoning to be expressed, using McCarthy's Abnormality propositions. A typical application domain concerns the forms of defeasible reasoning that can be held from a deep model of a complex device or system, where Abnormality propositions are used to represent possible (but unexpected) faulty components and where a limited number of failures are expected to occur simultaneously.

1 INTRODUCTION

Nonmonotonic logic has long been a central issue in knowledge representation and reasoning research. Many efforts have been devoted to the definition of formalisms with adequate expressive power and desirable logical properties. Even in the basic propositional setting, the computational counterpart of these logics has often been left apart by most researchers, probably because of the apparently bad inherent computational properties. For instance, the common pattern of default reasoning "If it is consistent to assume that X, then nonmonotonically infer Y" requires a consistency check, which is exponential in the worst case in the full standard propositional logic (unless $P = NP$). Not surprisingly, recent high quality theoretical research has shown the intractability of many nonmonotonic logic formalisms (see e.g. (Cadoli and Schaerf, 1993) for a comprehensive survey).

At the same time, some very impressive computational progress has been obtained with respect to the SAT problem. SAT is related to the above consistency check; it consists in checking whether a CNF propositional formula does exhibit a model: it is a canonical NP-complete problem, meaning that any algorithm to solve it should be exponential in the worst cases (unless $P = NP$). Using surprisingly efficient local search algorithms as first proposed by Selman and co-authors (Selman et al., 1992), we are now able to address large and difficult consistent SAT instances. However, local and stochastic search techniques are logically incomplete by nature. They can be used to show us that a CNF formula does have a model. However, when they fail to deliver a model after a preset amount of computing time, we cannot formally conclude that the formula is inconsistent. On the other hand, logically-complete algorithms exhibit a quite smaller actual scope (Selman et al., 1997). Recently, we have shown that, quite surprisingly, local search techniques can also prove extremely efficient in showing that CNF formulas are inconsistent (Mazure et al., 1996). To this end, the work performed by the local search algorithm is observed when it fails to deliver a model. Most often, a powerful heuristic allows one to detect an inconsistent kernel directly. More generally, this heuristic delivers a branching strategy that can boost logically-complete techniques *à la* Davis and Putnam (Davis and Putnam, 1960). Such a symbiotic combination of local search and complete technique is now widely recognized as one of the hottest topics for further progress in propositional reasoning (Selman et al., 1997). Additionally, it can also be a key issue for further significant progress in computing propositional nonmonotonic inferences, as the present work will illustrate it.

In this paper, a new approach to compute nonmonotonic inferences in a simple but useful propositional model-preference formalism is presented. It is original from at least two points of view. First, it makes use of local search while preserving logical completeness. Second, it proves efficient for an important class of very large nonmonotonic knowledge bases. More precisely, it extends the above SAT-related practical computational results to a nonmonotonic framework. The proof-strategy is based on the use of local search techniques for SAT together with an extension of the above efficient heuristic when these techniques fail to deliver a model. It is applied to a simple but useful nonmonotonic formalism allowing prioritized rules of default reasoning to be expressed, using McCarthys Abnormality propositions (McCarthy, 1986). The simplicity of the representation language together with the way it should be used by the knowledge engineer appears as a good trade-off between representational expressivity and experimental complexity with respect to a specific class of applications. Indeed, although no new result is obtained with respect to worst cases analysis, the approach proves experimentally efficient for an important kind of very large nonmonotonic knowledge bases.

The experimental good performance that we obtain is due mainly to the following correlated features:

1. the intrinsic properties of the concerned applications,
2. the correct use of the representation language, whose expressivity is carefully restricted to match our minimal needs,
3. the carefully defined structure of the decision procedure,
4. the low probability that worst cases do actually occur,
5. additional new powerful heuristic findings,
6. the focus that is laid successively on the different possible situations, according to their decreasing probability
7. the most efficient techniques used for each different type of situations.

This paper is organized as follows. First, the nonmonotonic propositional formalism under consideration is described. Then, local search techniques for SAT are briefly reviewed, including the powerful heuristic allowing inconsistency to be proved and that will be extended in this paper. Then, an original proof-procedure for this nonmonotonic formalism is presented, extending these new SAT-related findings. The focus is laid on the reasons explaining its experimentally good computational efficiency. The proof procedure itself, which is highly technical, is given in an Appendix. A sample of experimental results for very large KBs are then described. Finally, the limits of the approach are discussed, together with possible extensions.

2 A PROPOSITIONAL NONMONOTONIC FRAMEWORK AND ITS TARGETED APPLICATION DOMAIN

Unless some new highly improbable complexity results, we cannot hope for tractable computational approaches for most nonmonotonic propositional logics without some forms of approximations and restrictions on the expressive power of the knowledge representation formalism or reasoning mechanism, or without constraints on the concerned applications. In this paper, we keep the full language of propositional logic together with a simple model-preference reasoning mechanism. Accordingly, as we do not restrict the expressive power of the logic, we have to restrict its use to a certain class of applications, which however matches real-world problems and proves computationally tractable (very often).

The application domain that we address is the following one. We want to model defeasible forms of inference that can be drawn on *very large* propositional knowledge bases that represent complex devices or systems. We want to be allowed to assert rules of default reasoning that allow the knowledge engineer to represent the normal functioning conditions of the device, together with the possibility of failures (which would lead to inconsistency if the representation were simply using standard logic). The knowledge engineer is expected to be able to order the probability and importance of these possible failures, at least to some extent. We also assume that we shall not be faced with Murphy's law, few failures do occur at the same time. When several faulty components do occur, we are most interested with the most probable and important ones, as specified by the knowledge engineer. Indeed, in real systems, when several failures do occur, many symptoms of bad behavior are often only consequences of a single source problem. In a first analysis, it is better to concentrate on the most probable failure and avoid being confused by these other symptoms. We believe that this represents a really important class of applications for nonmonotonic reasoning. In this respect, our approach extends the diagnosis task, since we want to be able to detect the failures and to infer in a defeasible way in their presence.

The representation language under consideration is full propositional logic, where knowledge is expected to be encoded under conjunctive normal form. The language is enriched with a finite set of McCarthy's prioritized Abnormality propositions, noted Ab_i (McCarthy, 1986), allowing rules of default reasoning to be expressed together with exceptions. Abnormality propositions are used to describe possible (but unexpected) faulty components. According to the targeted application domain, only a limited number of Abnormality propositions are expected to be required to be *true* when the KB is queried. 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 $switch\_on \wedge \neg Ab_1 \Longrightarrow lights\_on$, and in clausal form by $\neg switch\_on \vee Ab_1 \vee lights\_on$. Let us stress again that in this very standard framework Ab_i propositions are expected to be $false$ under normal operating circumstances of the device since they are intended to represent unexpected faulty conditions of the device. This is thus a very specific use of Ab_i variables; we do not consider knowledge bases including many default rules where possible exceptions are expected to exist under normal circumstances, like "Typical humans are right-handed".

Let us interpret this knowledge representation formalism under a very general model-preference framework. First, let us recall that an *interpretation* assigns values from $\{true, false\}$ to the propositional variables of the (enriched) language. A *model* of a formula f is an interpretation under which the formula f gets the truth value *true*. A model will be represented by the set of propositional variables that it satisfies. The Abnormality propositions Ab_i are prioritized in the following sense. We say that Ab_i is *lower* than Ab_j (noted $Ab_i < Ab_j$) when $i < j$. As explained above, the knowledge engineer is expected to encode the default rules in such a way that more probable and important device failures relate to lower Ab_i. Accordingly, we shall prefer models that are such that the lowest Ab_i that they contain is as low as possible. Let us stress that this framework is easily extended to a pre-order on Ab_i propositions. More formally, let M_1, M_2, M_3 be models of KB. We say that

Definition 2.1
M_1 is a preferred model of KB iff:

- *M_1 does not contain any Ab_i*
- *or M_1 contains at least one Ab_i and :*
  1. *there does not exist a model M_2 of KB that does not contain any Ab_j ;*
  2. *there does not exist a model M_3 of KB containing an Ab_j that is such that $Ab_j < Ab_i$ for all Ab_i in M_1.*

Accordingly, nonmonotonic inference from KB is defined as follows. Let f be a literal, i.e. a propositional variable or its negation (the following is easily extended to clauses or formulas).

Let us assume that KB is consistent.

Definition 2.2
$KB \mid\sim f$ iff f is true in all preferred models of KB.

Let us introduce the following convenient definition.

Definition 2.3
M_1 is at least as preferable than M_2 iff:

- *M_1 does not contain any Ab_i,*
- *or M_2 contains no Ab_i ; in this case, M_1 does not contain any Ab_j,*
- *or both M_1 and M_2 contain at least one Abnormality proposition; in this case, $\forall Ab_i \in M_2, \exists Ab_j \in M_1$ such that $Ab_j \leqslant Ab_i$.*

Let us stress that this differs with most semantical accounts of prioritized circumscription[1] in the sense that the set of models is here partitioned into $n+1$ equivalence classes, where n is the total number of Abnormality propositions (roughly, only the lowest Ab_j proposition that is *true* in a model does influence the ordering relation; no refinement is made depending on the truth value of the not so low Ab_i propositions). This requirement is motivated by the nature of the targeted applications: most important and probable failures should be exhibited in order for a first analysis to be conducted, other ones are often simple consequence ones or are considered secondary and treated

[1] Other approaches to compute circumscritption and minimal models can be found in *e.g.* (Niemelä, 1996a; Niemelä, 1996b) and (Castell et al., 1996).

afterwards (they should not hide the *real* problem). Coincidently, this will also allow better experimental efficiency results to be obtained.

Accordingly, we can rewrite the definition of nonmonotonic inference in the following way, which will prove more convenient for our purpose.

Let us assume that KB is consistent.

Definition 2.4
$KB \mid\sim f$ iff there exists a model M_1 of KB satisfying f and there does not exist a model M_2 of KB that is at least as preferable than M_1 and that does not satisfy f.

3 LOCAL SEARCH FOR SAT

Let us now briefly review the local search techniques that allowed for significant progress in showing the consistency of very large and hard SAT instances. Let us simply recall here the most representative one, namely Selman et al.'s GSAT algorithm (Selman et al., 1992; Selman et al., 1993). Let us mention that even more efficient variants are now available. 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 (cf. Figure 1).

```
Procedure GSAT
Input: a set of clauses S, MAX-FLIPS, 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 (possibly negative)
                 in the number of clauses of S that
                 are satisfied by I
          I := I with the assignment of x reversed
       end-for
    end-for
    return ''no satisfying assignment found''
End
```

Figure 1: GSAT algorithm: basic version

Let us stress that additional moves (e.g. random ones) are provided in order to escape from local extrema. In the following, a variant procedure called TSAT is used (Mazure et al., 1997b). It makes use of a tabu list forbidding recurrent flips and appears extremely competitive. Moreover, it drops stochastic features of GSAT and can exhibit a purely deterministic behavior as far as the initial interpretations are not selected randomly. Let us stress that all results presented in the following Sections keep holding for most GSAT-like algorithms.

4 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 test has been repeated very extensively, giving rise to the same result extremely often (Mazure et al., 1996). TSAT (or any other GSAT-like algorithm) 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. Informally, a locally inconsistent SAT problem contains a *small* inconsistent kernel which would render the problem satisfiable if it were extracted from it. 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.

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 kernels converges towards the size of the SAT instance, no significant result is obtained. Fortunately, many real-life inconsistent KBs exhibit some small kernels of conflicting information. In particular, the KBs that we consider here and that translate the deep model of a complex device obey this property most of the time.

We have proposed and experimented several ways to use the above heuristic information to prove inconsistency in a formal way (Mazure et al., 1996). The most straightforward one consists in using GSAT-like algorithms to detect the probable inconsistent kernel. Then, complete techniques *à la* Davis and Putnam procedure (DP) (Davis and Putnam, 1960) 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 can be used to guide the search performed by the complete technique. In (Mazure et al., 1996), a basic procedure that combines Davis and Putnam's algorithm with this heuristic and that proves extremely competitive is proposed.

5 HOW INFERENCE CAN BE PERFORMED EFFICIENTLY (VERY OFTEN)

Let us now come back to our initial problem. How can we use the above techniques to compute the inferences that can be drawn in our nonmonotonic framework in an effective way?

In order to infer f from KB, we shall try to find a model M_1 of KB satisfying f in such a way that there is no model of KB as preferrable as M_1 and that does not satisfy f.

Clearly, from a *worst-case* analysis, the model-preference schema has been carefully restricted as to require at most $\mathcal{O}(n)$ calls to a satisfiability check, i.e. is $P^{NP[\mathcal{O}(n)]}$, where n is the total number of Abnormality propositions and not the size of the knowledge base. This form of nonmonotonic inference remains thus at the first level of the polynomial hierarchy, which is not usual for nonmonotonic logics. Roughly, this is due to the fact that only the lowest Ab_j proposition that is *true* in a model does matter with respect to the ordering relation between models; no refinement is made depending on the truth value of the not so low Ab_i propositions. This is one of the keys to good computational properties. But, in most *real* situations, although n can be very large, we shall see that a few calls (i.e. about 5) to a satisfiability testing procedure (that will themselves prove efficient extremely often) is actually enough to decide whether f can be inferred or not (most often).

Let us stress again that this restriction on the ordering between models is not an artificial feature but that this coincides with the nature of the specific but important targeted applications. In complex devices and systems, only most important and probable sources of bad behavior do actually matter for a first analysis. Knowledge is expressed in such a way that failures are translated using lower abnormality propositions that are required to be *true*, unless inconsistency occurs. Other symptoms of bad behavior using other abnormality propositions are often derivative ones and should not be taken into account directly, this in order to prevent them from hiding the *real* problem that is occurring.

Second, in real-world knowledge bases applications, when inconsistency does occur, it can be related extremely often to a small subpart of the KB. Really hard problems with respect to propositional unsatisfiability checking seem often related to globally inconsistent ones. In turn, this entails good (experimental) computational properties for (in)consistency checking, at least with respect to our specific range of considered applications. Namely, when the KB is consistent, local search proves efficient very often. Second, when the KB is inconsistent, complete techniques *à la* Davis and Putnam (DP), when focused on the trace of local search, proves efficient very often. Accordingly, these computational techniques often prove most of the times (experimentally) polynomial with respect to the size of the KB. Actually, they are exponential but with respect to the size of the smallest inconsistent kernel, which is most of the times very small with respect to the actual size of the KB. Obviously enough, unless $P = NP$, worst cases can be imagined leading to really out of reach required computing times. But we claim that the possibility that such situations do occur in our application domain is extremely low.

A third key to good actual computational efficiency is the way the proof procedure is devised. First, whenever a proof of consistency or inconsistency is required, we use a local search technique and, if necessary, a complete technique *à la* Davis and Putnam that focuses first on the probable inconsistent kernel put in light by the search. Accordingly, local search techniques always precede a call to a complete technique, which then benefits from the work performed before. We treat the most probable situations first, i.e. situations where *normal-circumstances* models (Mazure et al., 1997a) (i.e. models not containing any Ab_i) do exist. This corresponds with normal functioning conditions of the device. Two calls to a combination of a local search procedure with a complete technique for satisfiability testing can achieve the test of existence of such models for f and $\neg f$, assuming thus that any Ab_i is $false$. Cases of device failures are addressed in a second step (normally, the device is functioning in a correct way). In these cases, at least one Ab_i must be *true* for the KB to be consistent. Also, as far as a limited number of Abnormality propositions are required to be *true* at the same time to restore consistency, the trace of local search that was obtained at the first step delivers us very often the right set of Abnormality propositions that "restore" consistency when they are *true* (these are the abnormality propositions that belong to the most often falsified clauses, assuming all Ab_i being $false$). This also corresponds with the nature of the targeted applications: unless Murphy's law, few unexpected faulty components appear at the same time. This inconsistent kernel provides us with a list of Abnormality propositions Ab_j that will be the first ones to be checked to play the role of the lowest Ab_j in the tentative preferred models. It appears experimentally that they are often required to be *true* in any model of the KB. The role of Step 2 is thus to focus on this list to find out a model of f containing at least one of the members of this list, that is such that no preferrable model that does not satisfy f does exist.

Now, a very important and new heuristic finding is the following one. Assume we do not find a model of $KB \wedge f$ (resp. of $KB \wedge \neg f$) with some of the members of the list being *true*. In this case, extremely often, we shall not find such a model with other Ab_j being required to be *true*. This is linked to the accuracy of the first heuristic delivering the list; if the trace of Step 1 delivers us the right minimal inconsistent subsets of the KB, then we must necessary select as true at least one of the abnormality propositions in the list. Thus, when a model of $KB \wedge f$ (resp. $\neg f$) $\wedge$ Ab_i is found, instead of i successive calls to TSAT and DP to prove the inconsistency of $KB \wedge f$ (resp. $\neg f$) $\wedge$ Ab_j ($j \leqslant i$), we make just one call to TSAT and DP on $KB \wedge f$ (resp. $\neg f$) $\wedge$ $(Ab_1 \vee Ab_2 \vee \ldots \vee Ab_i)$. Most generally, no such model is found and the procedure is over. Moreover, selecting the lowest Ab_j in this list is often enough, as it represents the most probable failure. Accordingly, in most actual cases, the procedure will require a very small constant number of calls to a (in)satisfiability checking procedure. However, the decision procedure is logically complete and investigates the other possibilies when necessary. In this respect, one should note that whenever a model of $\neg f$ is found with its lower *true* Abnormality proposition being Ab_j, we know that we must try to find models of f that are such that their lowest *true* Ab_i is such that $i < j$.

Let us stress that this procedure can be optimized in several ways. For instance, successive calls to DP could avoid computing subtrees that have been considered in previous calls. More importantly, an obvious improvement consists in delimiting the connected component in the graph of the propositions occurring in $KB \cup \{f\}$. Accordingly, only the corresponding subpart of KB should be considered in the proof procedure. In the full paper, several variants of this proof-procedure are discussed and compared.

Finally, let us stress that this way of computing inferences is not sensitive to the fact that it is defined for literals. It is a straightforward task to extend it to clauses.

The complete proof-procedure, which is highly technical, is given in Annex.

Theorem 5.1
The Non-Monotonic Inference procedure is sound and complete with respect to the inference relationship of Definition 2.4.

6 EXPERIMENTAL RESULTS

Let us now briefly illustrate the experimental efficiency of the procedure. First, let us stress that experimenting nonmonotonic logic proof-procedures for very large KBs is quite unusual. Although there exist real-world benchmarks for the SAT problem, there are not many large real-world benchmarks for nonmonotonic logics that are available (we are talking about really larger benchmarks than most existing ones, which were created to check whether the logics under construction did exhibit adequate expressive power).

Accordingly, we collected and generated benchmarks by ourselves. They are described in the full paper. In particular, we have generated a class of benchmarks in such a way that they stick as much as possible to the targeted real-world problems. For instance, we transformed real-world SAT benchmarks to include Abnormality propositions, allowing previously encoded rules to suffer from exceptions.

In this paper, let us just relate a test (obtained on a 166 Pentium) that is a good representative of the size of the tested benchmarks and of the obtained computational results.

For instance, we merged 5 large consistent KBs that are related to VLSI circuits (namely, ssa-7522-{156,157,158,159,160}) with a small inconsistent Dubois SAT instance, all from (DIMACS, 1993). The resulting inconsistent KB contains 15929 clauses that are not necessarily Horn, using 7041 variables and is intractable for standard improved DP techniques. We restored the consistency of the KB by introducing 1593 different Ab_i propositions in a random way inside the KB (however, making sure that KB becomes consistent and using at most one Abnormality proposition by clause). This KB remains intractable using standard DP procedures; using TSAT we show within 4.6 seconds CPU time that it is consistent.

We selected f in a random way, from the literals occurring in KB and used our algorithm in order to show whether $KB \mid\sim f$ or not.

The first step consists in checking whether there exists a normal circumstances model for KB that satisfies f or falsifies it. TSAT (with its weight option, a tabu length set to 10, 10 tries and 10000 flips) is run on the KB, with all Ab_i set to false and f set to true, during 1min01s30. It fails to deliver a model (indeed, no such model does exist). The trace given by this step is given in Figure 2. Running DP on this trace allows us to prove that no such normal circumstances model does exist for f, using 0s94 CPU time.

A similar work is performed with f set to $false$, spending 59s26 for TSAT and 0s93 on its trace by DP to show the absence of normal circumstances model for $\neg f$.

The trace of Figure 2 allows us to state that, *most probably*, Ab_4 is to be *true* in order for the KB to become consistent. Indeed, Ab_4 appears in the most often falsified clauses (less us stress that the fact that this is a very low Ab_i is purely accidental, and as discussed above, will not influence the efficiency results, most often. It simply corresponds to a failure that exhibits one of the highest probability to occur in the modeled device).

Within 39s35, TSAT allows us to find out a model of $KB \wedge f \wedge Ab_4$. Then, we have to check that no model does exist for $KB \wedge \neg f \wedge (Ab_1 \vee Ab_2 \vee Ab_3 \vee Ab_4)$. This is done in one step, introducing the additional clause $Ab_1 \vee Ab_2 \vee Ab_3 \vee Ab_4$, using first TSAT and then DP

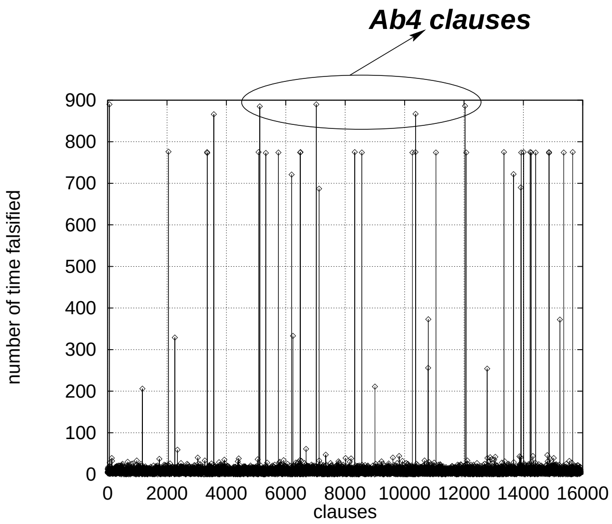

Figure 2: Trace of TSAT for $KB \wedge f \wedge (\neg Ab_1 \wedge \neg Ab_2 \wedge \ldots \wedge \neg Ab_{1593})$

to prove that no such model does exist. TSAT was run during 48s and the unique call to DP on the delivered trace was performed in less than 1.8 seconds.

7 CONCLUSIONS

Worst-case analysis is an important tool to exhibit the computational limits of a formalism. However, the extent to which actual situations do match worst cases should always be investigated. Accordingly, bad worst-case computational results about nonmonotonic formalisms should not be misinterpreted and should not *always* make us too much pessimistic about the actual possibility of implementing these formalisms, even for some large-scale applications.

In this context, the contribution of this paper is twofold. First, it describes a proof-procedure that proves efficient for very large nonmonotonic knowledge bases using the language of full propositional logic. To our best knowledge, this is very unusual. Second, the ingredients of this proof-procedure are themselves unusual, since they consist of local search mechanisms used in a way to preserve logical completeness.

But no miracle, as we do not restrict the expressive power of the logic, we restrict its use to a specific class of applications, which however matches real-world problems and proves computationally tractable (very often). The proof-procedure is carefully designed to fit those applications and to solve usual situations first. The considered applications require a specific use of the Abnormality propositions. However, we claim that if nonmonotonic logics are to be introduced in large-scale applications, it is acceptable to require the knowledge engineer a disciplined use of expressively restricted nonmonotonic ingredients when building the knowledge base. Also, as the paper illustrates it, we think that defining application-oriented proof-procedures for nonmonotonic logics can be a viable alternative to general proof-procedures.

In the full paper, we describe from a technical point of view the main other two requirements for our technique to remain tractable. Mainly, whenever inconsistency does occur in a knowledge base, it should be due to a small subset of it. We claim that this feature is a common one in the actual knowledge bases that we consider, which describe the deep model of physical devices. This is also the case for our last main requirement: unless Murphy's law, few unexpected faulty components appear at the same time. In the full paper, we describe how this may require some additional encoding policy to be respected by the knowledge engineer, without serious loss of expressive power.

Finally, this paper relates a particular way to import the recent impressive computational progress in solving SAT into the nonmonotonic research area. Many patterns of non-monotonic reasoning rely on some forms of consistency check, either in an explicit or in an implicit way. Accordingly, we are convinced that many other nonmonotonic formalisms could benefit from the recent progress about SAT. This paper is just a first step in this new promising direction for using nonmonotonic logics in large applications.

We also believe that after almost two decades of theoretical research about nonmonotonicity, algorithms that prove efficient with respect to very large applications can help in showing the practical interest of this important research domain.

Acknowledgements

This work has been supported in part by the Ganymède II project of the "Contrat de plan Etat/Nord–Pas-de-Calais". We thank P. Marquis for many stimulating discussions about the contents of this paper.

References

Cadoli, M. and Schaerf, M. (1993). A survey on complexity results for non-monotonic logics. *Journal of Logic Programming*, 17:127–160.

Castell, T., Cayrol, C., Cayrol, M., and Le Berre, D. (1996). Using the Davis and Putnam procedure for an efficient computation of preferred models. In *Proceedings of the Twelfth European Conference on Artificial Intelligence(ECAI'96)*, pages 350–354.

Davis, M. and Putnam, H. (1960). A computing procedure for quantification theory. *Journal of the Association for Computing Machinery*, 7:201–215.

DIMACS (1993). Second Challenge on Satisfiability Testing organized by the Center for Discrete Mathematics and Computer Science of Rutgers University. http://dimacs.rutgers.edu/Challenges/.

Mazure, B., Saïs, L., and Grégoire, É. (1996). Detecting logical inconsistencies (extended abstract). In *Proceedings of Math. & AI Symposium*, pages 116–121, Fort Lauderdale (FL USA). (extended version in "Annals of Mathematics and Artificial Intelligence").

Mazure, B., Saïs, L., and Grégoire, É. (1997a). Checking several forms of consistency in non-monotonic knowledge-bases. In *Proceedings ECSQARU-FAPR'97*, volume 1244 of *Lecture Notes in Artificial Intelligence*, pages 122–130, Bad Honnef (Germany). Springer.

Mazure, B., Saïs, L., and Grégoire, É. (1997b). Tabu search for SAT. In *Proceedings of the Fourteenth National Conference on Artificial Intelligence (AAAI'97)*, Providence (Rhode Island, USA). 281–285.

McCarthy, J. (1986). Applications of circumscription to formalizing common-sense knowledge. *Artificial Intelligence*, 28:89–116.

Niemelä, I. (1996a). Implementing circumscription using a tableaux method. In *Proceedings of the Twelfth European Conference on Artificial Intelligence(ECAI'96)*, pages 80–84.

Niemelä, I. (1996b). A tableau calculus for minimal model reasoning. In Miglioni, P., Moscato, U., Mundici, D., and Ornaghi, editors, *Theorem Proving with Analytic Tableaux and Related Methods, 5th International Workshop*, pages 278–294, Terrasini, Palermo, ITALY. Springer Verlag, LNCS 1071.

Selman, B., Kautz, H. A., and Cohen, B. (1993). Local search strategies for satisfiability testing. In *Proceedings of the DIMACS Workskop on Maximum Clique, Graph Coloring and Satisfiability.*

Selman, B., Kautz, H. A., and McAllester, D. (1997). Computational challenges in propositional reasoning and search. In *Proceedings of the Fifteenth International Joint Conference on Artificial Intelligence (IJCAI'97)*, volume 1, pages 50–54, Nagoya (Japan).

Selman, B., Levesque, H., and Mitchell, D. (1992). A new method for solving hard satisfiability problems. In *Proceedings of the Tenth National Conference on Artificial Intelligence (AAAI'92)*, pages 440–446.

Appendix: Proof Procedure

The TSAT procedure is not described in this appendix, however a detailed description can be found in (Mazure et al., 1997b).

```
Procedure Unit_Propagation;
Input: A knowledge base KB in CNF ;
Output: A knowledge base without unit clause or
        a knowledge base with an empty clause ;
Begin
   while ((∃ unit clause ∈ KB)
          and (∄ empty clause ∈ KB))
    do
      l := a literal s.t. (l ∈ c and c is a unit clause) ;
      KB := KB ∧ l ;
    done
    return KB ;
End ;
```

```
Procedure DP_focused_on_trace;
Input: A knowledge base KB in CNF, a list T
       of pairss < literal, score^2 >, built
       from the trace of TSAT computation ;
Output: true if KB est consistent, false otherwise ;
Begin
  KB := Unit_Propagation(KB) ;
  if (∃ empty clause ∈ KB return false ;
  elif (KB = ∅) return true ;
  else
     l := the literal with the greatest score in T,
          s.t. l ∈ KB ;
     return ( DP_focused_on_trace(BC ∧ l,T) ∨
              DP_focused_on_trace(BC ∧ ¬ l,T) ) ;
  endif
End ;
```

```
Procedure Satisfiability_Test ;
Input: A knowledge base KB in CNF ;
Output: true if KB is consistent, false otherwise ;
Begin
  if (TSAT(KB)) return true ;
  else
     trace := the trace of TSAT ;
     return (DP_focused_on_trace(KB, trace)) ;
  end if ;
End ;
```

```
Procedure NonMonotonic_Inference ;
Input: A KB (assumed consistent) and f a littral ;
Output: true if KB |~ f, false otherwise ;
Begin

   { Step 1 : Search for normal-circumstances models }

  KB' := KB ∧ {¬ Ab_i}(∀ Ab_i ∈ KB) ∧ f ;
  if (Satisfiability_Test(KB'))
     {A normal-circumstances model exists,
      this model is necessary a preferred model}
    KB' := KB ∧ {¬ Ab_i}(∀ Ab_i ∈ KB) ∧ ¬ f ;
    if (Satisfiability_Test(KB')) return false ;
      {Both Normal-circumstances models exist for
       KB ∧ f and for KB ∧ ¬ f,thus KB |≁ f }
    else return true ;
    end if ;
  else
    KB' := KB ∧ {¬ Ab_i}(∀ Ab_i ∈ KB) ∧ ¬ f ;
    if (Satisfiability_Test(KB')) return false ;
      {Normal-circumstances model exists for
       KB ∧ ¬ f but not for KB ∧ f,thus KB |≁ f }
    else

      { Step 2 : Search of preferred models
       that are not normal-circumstances ones}^3

      KB' := KB ∧ {¬ Ab_i}(∀ Ab_i ∈ KB) ∧ f ;
      TSAT(KB')^4 ;
      L := {Ab_s, ..., Ab_t} ; { L is the set of
    Abnormality propositions sorted in decreasing order
   w.r.t. their score obtained using the trace of TSAT};
      proof := false ; {true, when KB |~ f}

      while (not proof) and (L ≠ ∅)
      do
        lowest_Ab_j := the first element of L ;
        L := L - {lowest_Ab_j}
        KB' := KB ∧ {¬ Ab_i}(∀ Ab_i<lowest_Ab_j) ;
        KB' := KB' ∧ lowest_Ab_j ∧ f ;
        if (Satisfiability_Test(KB'))
          c := Ab_1 ∨ Ab_2 ∨ ... ∨ lowest_Ab_j
          KB' := KB ∧ c ∧ ¬ f ;
          if (Satisfiability_Test(KB'))
          then
             {there are preferred models for f
              and for ¬ f, including lowest_Ab_i
            Ab_r := the lowest Ab_i in the
                    discovered model of ¬ f ;
            L := {Ab_j s.t. Ab_j < Ab_r} ;
          else
             {the exists a model of KB ∧ f
              taht has no preferable model that
              satisfy ¬ f, thus KB |~ f}
            proof := true ;
          endif
        endif
      done
      return proof ;
    end if
  end if
End ;
```

[2] The score of a litteral is the number of times that this litteral has been falsified during the TSAT computation.

[3] If we reach this step, then there does not exist any normal-circumstances model of KB.

[4] We know that this call will not succeed since it has already failed in step 1! But we need the trace of this call. Le us stress that it is not actually computed again, because the scores of Abnormality propositions are saved during the first call to TSAT.