Automatic Symmetry Breaking Method Combined with SAT

Jian Zhang
Laboratory of Computer Science
Inst. of Software, Chinese Acad. Sci.
P.O.Box 8718, Beijing 100080, China
Tel. (86) (10) 6264 4790
zj@ios.ac.cn

ABSTRACT

Finding models of first-order formulas is an important and challenging problem in computational logic. Many satisfiable formulas have finite models. To find a finite model of a first-order formula, one may use a backtracking search procedure directly, or transform the problem into propositional satisfiability. In the former approach, symmetries can be used effectively to prune the search space; while the latter approach benefits from the efficiency of the unit propagation rule. In this paper, a hybrid approach is proposed. The automatic symmetry breaking method is used as a preprocessing step, and then the propositional satisfiability procedure is called to complete the search. Experimental results show that the hybrid approach is better in some cases.

Keywords

Finite models, backtracking search, symmetry.

1. INTRODUCTION

An important problem in computational logic is to find models of formulas automatically. The existence of a model implies the satisfiability of the formula. A model may also serve as a counter-example showing that some conjecture does not hold. In the propositional logic, the satisfiability problem is known as SAT. It is the first NP-complete problem. In the first-order logic, the problem is undecidable in general. However, in many cases, a satisfiable formula has *finite* models.

In the past 10 years, a number of finite model searching programs have been developed around the world. They are becoming more and more efficient, and they have been used to solve many problems in finite mathematics. (See, e.g., [2,4,5,6,8].) Some of the tools transform the original problem into SAT and then use the Davis-Putnam procedure; while the others process first-order formulas directly.

SAC 2001, Las Vegas, NV

For simplicity, we assume that the first-order logic language is unsorted (or untyped). Let us denote by n the size of the model. Thus we search for an n-element model of the input formula. Without loss of generality, we represent the domain by the following set of integers: $D_n = \{ 0, 1, \ldots, n\text{-}1 \}$.

Obviously, we can find an n-element model or conclude that it does not exist, using exhaustive search methods. In fact, the problem can be regarded as a finite constraint satisfaction problem (CSP). The unknowns are such ground terms as $f(0,1)$, $g(3)$, etc. (Here f and g are function symbols in the input formula.) The possible values of the unknowns are elements of the domain D_n. We can decide the values one by one through backtracking search. Typical tools based on this approach include FINDER [6] and SEM [8].

Alternatively, we may transform the finite model finding problem into SAT and then use a SAT tool. The input first-order formula is instantiated over the domain D_n, and then a set of propositional formulas are generated. The satisfiability is decided by a propositional method such as the Davis-Putnam algorithm. When a solution exists, it can be translated into a first-order model. (For example, a binary function is represented by an $n \times n$ table.) For more details, see [3,4]. The program MACE [4] is based on this approach and it is quite popular.

On many problems, tools based on the Davis-Putnam method are more efficient. After all, the satisfiability problem in the propositional logic has been studied for about 40 years. Many techniques have been invented. But on some other problems, the first-order search method is better and can find quite large models. For such a problem, if we translate the first-order formula into the propositional logic, we may get too many propositional formulas. Another benefit of the first- order approach is that, we can exploit the structure of the formula to reduce the search space (e.g., to avoid exploring symmetric subspaces).

In this paper, we describe a way of combining the two approaches. The new method first generates a set of *partial* models using the first-order search procedure. Then for each of them, the Davis-Putnam procedure is invoked to extend it to a full model. The process is terminated as soon as a full model is found. In other words, first-order reasoning is performed at the upper levels of the search tree, while propositional reasoning is done at the lower levels.

In the sequel, we shall describe the new method in detail. Some experimental results will be reported, based on the combination of the programs SEM and MACE.

2. SYMMETRIES

In computational logic, a first-order formula is often transformed into a set of *clauses*. Such a transformation preserves (un)satisfiability. A clause is a disjunction of literals, while a literal is a predicate symbol (or its negation) applied to a set of first-order terms. A unit clause has only one literal. In a clause, all variables are universally quantified. Thus the clause $f(x,y) = f(y,x)$ means that, for any x and any y, $f(x,y)$ equals to $f(y,x)$.

Given a set of first-order clauses, suppose a finite model searcher is asked to find all models of size n. Many symmetric models will be found if we do not take special measures to avoid them. One model is *symmetric* (or *isomorphic*) to another, if they are the same under some permutation of the elements.

Example 1. Quasigroup identity problems [2, 6]. A quasigroup has only one binary function symbol f. Its multiplication table is a Latin square. A quasigroup of order n is an $n \times n$ matrix, each row (column) of which is a permutation of n numbers. For example, the following table represents a quasigroup of order 3:

f	0	1	2
0	0	2	1
1	2	1	0
2	1	0	2

A quasigroup may satisfy some identities (or equations). For example, the above quasigroup satisfies the equation: $f(x,x) = x$, because the number at the i'th row and the i'th column is equal to i. Such a quasigroup is called an *idempotent* quasigroup. Mathematicians are interested in the existence of quasigroups satisfying various other identities, and model finding programs can be quite helpful when the size is not too large. For more details, see [2, 6].

When searching for quasigroups, we may add some special constraints such as $f(x,n-1) \geq x-1$ (for all x), so as to avoid exploring symmetric subspaces [2, 6]. To see the usefulness of this technique, let us look at the QG5 problem. It asks for idempotent quasigroups satisfying the identity $f(f(f(y,x),y),y) = x$. Suppose the size is 5. If we use the additional constraint, we get only one model, as shown below.

f	0	1	2	3	4
0	0	2	1	4	3
1	4	1	3	2	0
2	3	4	2	0	1
3	1	0	4	3	2
4	2	3	0	1	4

Otherwise, more than one models are found, which are isomorphic to each other. The following is another model. It is isomorphic to the previous one. If we exchange the two elements 1 and 2, the second model will become the same as the first one.

f	0	1	2	3	4
0	0	2	1	4	3
1	3	1	4	0	2
2	4	3	2	1	0
3	2	4	0	3	1
4	1	0	3	2	4

The above technique works well on quasigroup problems. But it certainly does not work on any problem.

For many problems that contain constant symbols, models exist in which the constants are assigned distinct values. MACE (version 1.3.4) has a command-line option "-c". When it is used, the program assumes that the constants' values are different from each other. Then some search efforts can be saved.

Example 2. Noncommutative groups. The axioms are as follows.

$f(e,x) = x.$ $\qquad$ $f(g(x),x) = e.$

$f(f(x,y),z) = f(x,f(y,z)).$ $\qquad$ $f(a,b) \neq f(b,a).$

Here a and b are two constants, e denotes the identity element, f and g denote the multiplication and inverse operations, respectively.

On a SUN Sparcstation 10, MACE needs about 55 seconds to find the first model. However, if the "-c" option is used, the running time of MACE is reduced to about 3 seconds. An alternative is for the user to specify part of the model. For example, one can assign 0 to e, 1 to a, and 2 to b. But this needs some insight and careful reasoning about the problem.

This technique does not always work, either. For some problems, there is a solution in which two different constants take the same value. One example is the following problem from combinatory logic [7]: Does the fragment with the basis consisting of B and N_1 satisfy the strong fixed point property? The first-order clauses are as follows:

$a(a(a(B,x),y),z) = a(x,a(y,z))$
$a(a(a(N1,x),y),z) = a(a(a(x,y),y),z)$
$a(y,f(y)) \neq a(f(y),a(y,f(y)))$

Here a is a binary function symbol. When $B = N_1 = 0$, there is a model of size four.

Benhamou and Sais [1] introduced a method for using symmetries in propositional reasoning. But their method requires the dynamic search for a symmetry. This is very expensive when there are a large number of clauses.

It appears to be easier to use symmetries in the first-order search method, which processes the first-order clauses directly. SEM [8] adopts the so-called *least number heuristic* (LNH) to eliminate symmetries. The basic idea of this technique is that, in the early stages of the search, only a few elements of the domain are used, which have some distinguished properties. The other elements are equivalent. In the search process, we need a special parameter, *mdn* (for "maximal designated number"), which is an integer. It is initialized to (-1).

The intuitive meaning of *mdn* is as follows. The set D_n is divided into two disjoint subsets, [0..*mdn*] and [(*mdn*+1)..(*n*-1)]. The integers in the first subset have already been used in the search, while the integers in the second subset have not yet been used.

When choosing values for an unknown, we consider every integer in the first subset; but in the second subset, we consider only one integer --- the smallest one. The value of *mdn* changes during the execution.

The technique can be shown to be correct. When models exist, at least one will be found. The heuristic has several appealing features. It is automatic and effective (especially at the upper levels of the search tree). It is also quite general, and can be applied to a variety of problems.

Suppose we want to find a 6-element quasigroup satisfying the QG5 identities using SEM (version 1.7). Fig.1 shows the first 3 levels of the search tree. (X means a dead end or bad choice.) When choosing a value for the unknown *f*(0,1), *mdn* is equal to 1. Thus we need only consider the elements 0, 1, 2 (but not 3, 4, 5) as possible values of *f*(0,1).

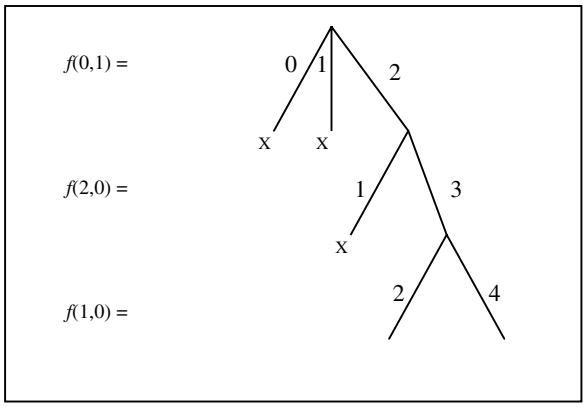

Fig. 1. Part of the Search Tree.

3. COMBINATION OF AUTOMATIC SYMMETRY BREAKING WITH SAT

As we mentioned earlier, both the first-order search approach and the Davis-Putnam method have some merits and weaknesses. Naturally we may consider combining them, so that we can have the efficiency of propositional reasoning and the low cost of symmetry elimination.

We slightly modified SEM and combined it with MACE. Let us simply call the new system MS. (It is not really a new tool, but rather a combination of two existing programs.) A new option, "-d*n*" is introduced. Here *n* is a positive integer, corresponding to the program variable *MS_Depth*.

MS works as follows. At first, SEM is executed. After the search depth reaches *MS_Depth*, MACE is executed with the partial models generated by SEM.

Fig. 2 shows the hybrid search process when *MS_Depth* = 2. (The arrow denotes the boundary between the two programs.)

Example 2. (continued) Suppose we want to find a noncommutative group of size 6, using the four formulas given previously. As we mentioned earlier, if we do not add any

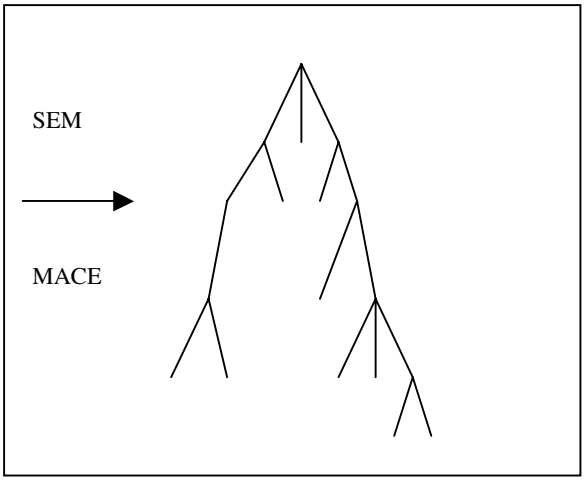

Fig.2. Hybrid Search

special constraints or use the command-line option "-c", the running time of MACE will be close to 1 minute.

Now we use MS with the option "-d3". The first 3 choices of SEM are:

f(0,0) = 0, e = 0, g(0) = 0.

The partial model is defined as follows.

e = 0, g(0) = 0, f(0,0) = 0, f(0,1) = 1,

f(0,2) = 2, f(0,3) = 3, f(0,4) = 4, f(0,5) = 5.

When this is given to MACE, a complete model is found in about 6 seconds on the same machine (SPARCstation 10). More than half of the time is spent on the instantiation of clauses. The actual time spent on backtracking search is about 1 second.

This example is very simple, and SEM alone performs very well on it. But it shows the effectiveness and generality of the hybrid approach. In addition to the constant *e*, other ground terms are defined in the partial model. The values of some terms can be equal.

There are several different ways of implementing MS. In the first way, each time SEM generates a partial model, MACE is called to complete the search. If no model is found, SEM will try to find another partial model. The process is repeated until a complete model is found or all possibilities have been checked. This method is easier to implement, and MACE does not have to be modified. We need only provide different input formulas to MACE. However, for each input specification, MACE has to translate it into propositional formulas. This may take a lot of time.

Another way of implementing MS is as follows. We first let SEM generate all partial models up to som depth (e.g., those corresponding to all paths from the root, which have length 3). Then they are added to the original input formula, and given to MACE.

As an example, let us see the partial search tree in Fig.1. From the two partial models, we get the following additional constraint:

(*f(0,1) = 2* and *f(2,0) = 3* and *f(1,0) = 2*) or

(*f(0,1) = 2* and *f(2,0) = 3* and *f(1,0) = 4*).

The benefit of this approach is that, the first-order formulas are instantiated only once.

Of course, after SEM generates all the partial models, we may also add each partial model to the original input, and let MACE run several times (in batch mode).

A disadvantage of the above approach is that, if the value of *MS_Depth* is not too small, there may be many partial models, most of which are useless. Usually there is no solution in the left part of the search tree. To reduce the number of partial models, we add another option "-R*m*", where *m* is a positive integer. Its meaning is that, the partial models are saved only after SEM has been running for *m* steps (called "rounds").

4. EXPERIMENTAL RESULTS

In this section, we report some experimental results. The 3 programs MACE, MS and SEM were run on a SUN server with two 400 MHz CPUs and 1024 MB memory. They are tested on well-known benchmarks, some of which are the following:

- NCG.*n* : Find an *n*-element non-commutative group. When running MACE to solve this problem, the option "-c" is used. When running MS, the options "-R20 -d3" are used.

- QG5.*n* : Find an *n*-element idempotent quasigroup satisfying the QG5 identity. When running MACE to solve this problem, the option "-x1" is used. The purpose is to eliminate some isomorphism by a special constraint, as described previously. When running MS, the options "-R20 –d5" are used.

- OLE1.*n*: Find an *n*-element ortholattice violating some equation (E1). For more details about this problem, see [5]. When running MS on this problem, the options "-R20 -d3" are used.

- GC.*n.m*: the graph coloring problem with *n* vertices and *m* colors. The experiments were done on propositional clauses, since the constraints can be naturally represented by many-sorted formulas, but the sorted language is not well-supported by the current version of MACE. The partial models are added manually to the original input. On this problem, the option "-d3" or "-d4" is used when running MS.

For each problem instance, we try to find only 1 model, not all models. The results are summarized in Table 1. For MACE and MS, we give the running times as well as the branches (splits) of the search tree.

The MS time does not include the time used by SEM to generate partial models, because the latter is trivial in most cases. For the graph coloring problems, the running time of MS is simply the sum over several MACE runs.

For the first two problems (NCG and QG), if we do not use any special constraints (as specified by the options "-c" and "-x1"), the running times of MACE will be much longer.

Table 1. Comparison of Performances

Problem	MACE	MS	SEM
NCG.7	471 splits, 7.05 secs.	112 splits, 3.36 secs.	0.13 secs.
NCG.8	90 splits, 4.60 secs.	124 splits, 6.47 secs.	0.02 secs.
QG5.9	18 splits, 0.09 secs.	27 splits, 0.10 secs.	0.02 secs.
QG5.10	199 splits, 0.24 secs.	225 splits, 0.27secs.	0.09 secs.
OLE1.8	66329 splits, 127 secs.	444 splits, 16 secs.	1.85 secs.
OLE1.9	47 splits, 17 secs.	40 splits, 17 secs.	32.32 secs.
OLE1.10	30455 splits, 242 secs.	233 splits, 53 secs.	674.28 secs.
GC.40.10	244 seconds	4 secs.	5076 secs.
GC.50.10	more than 3 hours	1305 secs.	more than 3 hours

5. CONCLUDING REMARKS

To find a finite model of first-order formulas, we may use a backtracking search procedure that processes the formulas directly, or transform them into propositional clauses and then use a SAT program. In the former approach, it is easier to make effective use of symmetries to prune the search space. But in general, current SAT tools are more efficient when deducing contradiction or new clauses from a given set of clauses.

In this paper, a scheme for combining the two approaches is presented, and different ways of implementation are described. In addition, some experimental results are reported, based on the public-domain programs SEM and MACE. They show that the hybrid approach is beneficial when solving certain problems. However, for a particular problem instance, there is no general rule stating when the search process has to be transferred from SEM to MACE. The values of the special parameters (e.g., *MS_Depth*) can be decided by experiments.

6. ACKNOWLEDGMENTS

This work is supported by the Natural Science Foundation of China (NSFC) under grant No. 69703014 and the National Key Basic Research Plan (NKBRSF) under grant No. G1998030600.

7. REFERENCES

[1] Benhamou B., and Sais, L. Tractability through symmetries in propositional calculus. Journal of Automated Reasoning, 12 (1994), 89-102.

[2] Fujita, M., Slaney, J., and Bennett, F. Automatic generation of some results in finite algebra. in Proc. of IJCAI-93, 52-57.

[3] Kim, S., and Zhang, H. ModGen: Theorem proving by model generation. In Proc. of AAAI-94, 162-167.

[4] McCune, W. A Davis-Putnam program and its application to finite first-order model search: Quasigroup existence problems. Tech. Memo ANL/MCS-TM-194, Argonne National Laboratory, Argonne, IL., USA, 1994. http://www-unix.mcs.anl.gov/AR/mace

[5] McCune, W. Automatic proofs and counterexamples for some ortholattice identities. Information Processing Letters 65 (1998), 285-291.

[6] Slaney, J., Fujita, M., and Stickel, M. Automated reasoning and exhaustive search: Quasigroup existence problems. Computers & Mathematics with Applications 29 (1995), 115-132.

[7] Wos, L. The kernel strategy and its use for the study of combinatory logic. Journal of Automated Reasoning, 10 (1993), 287-343.

[8] Zhang, J., and Zhang, H. SEM: a system for enumerating models. In Proceedings of IJCAI-95, 298-303.