

# Two techniques to improve Finite Model Search

Gilles Audemard, Belaid Benhamou, and Laurent Henocque

Laboratoire d'Informatique de Marseille  
Centre de Mathématiques et d'Informatique  
39, Rue Joliot Curie - 13453 Marseille cedex 13 - France  
Tel : 04 91 11 36 25 - Fax : 04 91 11 36 02  
{audemard, benhamou, henocque}@lim.univ-mrs.fr

**Abstract.** This article introduces two techniques to improve the propagation efficiency of CSP based finite model generation methods. One approach consists in statically rewriting some selected clauses so as to trigger added constraint propagations. The other approach uses a dynamic lookahead strategy to both filter out inconsistent domain values and select the most appropriate branching variable according to a first fail heuristic.

## 1 Introduction

Many methods have been implemented to deal with many-sorted or uni-sorted theories: FINDER [7], FMSET [3], SATO [8], SEM [11], FMC [5] are known systems which solved some open problems.

The method SEM (System for Enumerating Models) introduced by J. Zhang and H. Zhang in [11] is one of the most powerful known methods for solving problems expressed as many-sorted theories.

The goal of this article is to explore ways to improve SEM by increasing the propagations it performs (i.e. the number of inferred negative ground literals) so as to reduce the search space and overall computation time. A first possible improvement is a static preprocessing which automatically rewrites clauses having a specific structure.

A second improvement consists in a dynamic domain filtering achieved by using a lookahead at some nodes of the search tree. This lookahead procedure uses unit propagation and detects incompatible assignments (e.g. trying  $f(0) = 0$ , then  $f(0) = 1$  ...). This filtering is augmented by the introduction of a new heuristic, in the spirit of the SATZ propositional solver (see [4]).

This article is organized as follows : Section 2 defines the first-order logic theories accepted as input language and the background of the SEM algorithm. In section 3 we study two techniques which improve SEM efficiency. In section 4, we compare our work with other methods on mathematical problems. Section 5 concludes.

## 2 Background and SEM description

The theories accepted as input by the model generator SEM are many sorted first order theories, with equality, without existential quantifiers, in clause normal form (CNF). Since we are interested in finite models only, all sorts are finite. Because all the variables are universally quantified, the quantifiers are usually omitted.

We call the *degree* of a literal the number of its functional symbol occurrences. We call a *cell* the *ground term*  $f(e_1, \dots, e_k)$  where all  $e_i$  are sort elements. An interpretation of a theory maps each cell to a value from the appropriate sort. A model of a theory is an interpretation which satisfies all its clauses.

As an initial preprocessing stage, SEM expands the original theory axioms to the set of their terminal instances (i.e. ground clauses), by substituting for each logical variable all the members of the appropriate sort. SEM's finite model search algorithm is described in figure 1. It uses the following parameters :  $A$  the set of assignments,  $B$  the set of unassigned cells and their possible values and  $C$  the set of ground clauses. The function *Propa* of the search algorithm propagates the assignment from  $A$  to  $C$ . This simplifies  $C$  and may force some cells in  $B$  to become assigned. It modifies  $(A, B, C)$  until a fixed point is reached or an inconsistency is detected, and returns the modified triple  $(A, B, C)$  upon success. For a full description of SEM and the propagation algorithm one can refer to [9] and [11].

```
Function Search( $A, B, C$ ) : Return Boolean
If  $B = \emptyset$  Then Return TRUE
Choose and delete  $(ce_i, D_i)$  from  $B$ 
If  $D_i = \emptyset$  Then Return FALSE
for All  $e \in D_i$  Do
   $(A', B', C') = Propa(A \cup (ce_i, e), B, C)$ 
  If  $C' \neq False$  Then Search( $A', B', C'$ )
```

Algorithm 1: SEM Search Algorithm

## 3 Two Domain Filtering Techniques

SEM's propagation algorithm allows propagation of negative assignments only when literals with the form  $ce! = e$  exist in the set of clauses ( $ce$  is a cell and  $e$  an element). Otherwise, only positive facts (value assignments) are propagated. This leads to an increase in the number of decision points necessary for the search, and potentially increases run times. Because SEM performs some amount of value elimination using negative facts, one approach consists in favoring it by rewriting some clauses to give them the appropriate structure. This static technique is performed in a preprocessing phase.

The second approach is dynamic, involving computations performed at each decision node. It consists in using a lookahead technique, called unit propagation, to eliminate selected values from the domains of selected cells.

### 3.1 Clauses Transformation

SEM performs value elimination when clauses contain negative literals of degree two. We can thus rewrite some selected clauses, using a flattening technique as in FMSET [3] to rewrite clauses to logically equivalent clauses containing negative literals of degree two. Because such a transformation introduces auxiliary variables, and thus increases the number of ground clause instances, such a rewriting is a tradeoff. Candidate clauses are thus carefully selected : we restrict the rewriting process to clauses of degree 3. This transformation allows to drastically reduce on some problems the number of decision points and the execution times as shown in section 4, the results obtained with this rewriting technique are listed under the name CTSEM (Clause Transformation in SEM).

**Definition 1.** *A reducible clause is a clause which contains a literal with the following pattern:  $f(x_1, \dots, x_m, g(x_{k+1}, \dots, x_l), x_{m+1} \dots x_k) = x_0$  where  $f, g$  are two functional symbols and  $x_{i \in \{1..l\}}$  are variables. Such a literal is called reducible.*

By using the clause transformation algorithm described in [3], we can rewrite each reducible literal to the form :  $f(x_1, \dots, x_m, v, x_{m+1} \dots x_k) = x_0 \vee v \neq g(x_{k+1}, \dots, x_l)$ . This preserves the semantics of the clause, and introduces the negative literal  $v \neq g(x_{k+1}, \dots, x_l)$ . It requires the introduction of an auxiliary variable  $v$ .

*Example 1.* The literal  $h(h(x, y), x) = y$  is reducible and can be transformed to its logical equivalent  $h(v, x) = y \vee v \neq h(x, y)$ . Now, the ground clause  $h(0, 1) \neq 0 \vee h(0, 0) = 1$  exists in the set of ground clauses. When we assign  $h(0, 0)$  to 0 the second literal of the previous clause becomes false. So SEM can propagate the fact  $h(0, 1) \neq 0$ . This eliminates 0 from the domain of the cell  $h(0, 1)$ .

### 3.2 Value Elimination

At a given node of the search tree, let  $B$  equal the set of yet unassigned cells and  $C_A$  the set of axioms simplified by assignments of cells in  $A$ . In other words  $C_A = C'$  such that  $(A', B', C') = Propa(A, B, C)$ . Let  $(ce, D_{ce}) \in B$  and let  $e \in D_{ce}$ . If  $Propa(A \cup \{(ce, e)\}, B, C)$  leads to inconsistency, then we can remove the value  $e$  from  $D_{ce}$ .

However, such unit propagations are time consuming. We must restrict the number of calls to  $Propa$  in order to obtain an efficient method. We use here a property similar to the one introduced for SAT problems in [2]. After the call to  $Propa(A \cup \{(ce, e)\}, B, C)$ , there are two possibilities :

- $C_{A \cup \{(ce, e)\}} = False$  and then  $D_{ce} = D_{ce} - \{e\}$  : Value elimination.

- $C_{AU\{(ce,e)\}} \neq False$  : the value assignments  $(ci, e_i)$  propagated during the process are compatible with the current assignment and would not lead to value elimination if tried later.

This drastically reduces the number of possible candidates for propagation, and minimizes the number of calls to *Propa*. Formally, we have the following propositions :

**Proposition 1.** *Let  $(ce, e) \in B$ , if  $C_{AU\{(ce,e)\}} \models \perp$  then  $C_A$  is equivalent to  $C_A \wedge (ce \neq e)$ .*

This property is used to eliminate values from the cell domains.

**Proposition 2.** *Let  $(ce, e) \in B$ , if  $C_{AU\{(ce,e)\}} \models (ce_1, e_1), \dots, (ce_n, e_n)$  and if  $C_{AU\{(ce,e)\}} \not\models \perp$  then  $\forall i \in \{1 \dots n\} | ce_i \in B, C_{AU\{(ce_i, e_i)\}} \not\models \perp$*

This property avoids to propagate useless facts. This allows to perform fewer calls to the *Propa* procedure.

An additional possibility to reduce the number of unit propagations is to select which cells must be tried. We note  $T \subseteq B$  the set of cells which are candidates for unit propagation. Because of symmetries (LNH), only the cells with indices less or equal than *mdn* need to be considered. We call those cells *mdn cells*. The results obtained using this dynamic filtering technique are listed in section 4 under the name VESEM.

### 3.3 A First Fail Heuristic

In its original version, SEM chooses as the next cell the one with the smallest domain, and tries to not increment *mdn*. We note  $H$  these previous conditions. Then the heuristic chooses as the next variable to instantiate, the one that both satisfies conditions  $H$  and that maximizes the count of the number of propagations done on each cell for all their possible values. This approach is similar to the one described in [4] for propositional logic. The algorithm of this heuristic and value elimination process is shown in the algorithm 2.

*Remark 1.* In algorithm 2,  $Mark[ce,e]=True$  means that the value  $e$  of the cell  $ce$  can be suppressed. The number  $Nb$  equals the number of propagations.

## 4 Experimentations

We compare SEM, VESEM (SEM + Value Elimination), CTSEM (SEM + Clause Transformation preprocessing) and CTVESEM (SEM + Clause Transformation + Value Elimination) on a set of well known problems. Run times are in seconds. All experiments were carried out under Linux on a K6II 400 PC with 128 MB of RAM. A '+' indicates that a program fails to solve a problem in less than two hours.

```

Function Up_Heuristic( $A, B, C$ ) : Return Next cell to choose
For All  $(ce, D) \in T$  Do
  For All  $e \in D$  Do  $\text{Mark}[ce, e] = \text{True}$ 
For All  $(ce, D) \in T$  Do
   $Nb = 0$ 
  For All  $e \in D$  such that  $\text{Mark}[ce, e] = \text{True}$  Do
     $(A', B', C') = \text{Propa}(A \cup (ce, e), B, C)$ 
    For All  $(ce', e')$  propagated Do  $\text{Mark}[ce', e'] = \text{False}$ 
     $Nb = Nb + 1$ 
  If  $C' = \text{False}$  Then
     $D = D - \{e\}$ 
    If  $|D| = 1$  Then return  $ce$ 
  Else
     $w(ce) = w(ce) + Nb$ 
Return  $ce$  with the smallest domain and maximising  $w$ 

```

Algorithm 2: Value Elimination and Heuristic

#### 4.1 Quasigroup Problems

A quasigroup is a binary operator '.' such that the equations  $a.x = b$  and  $x.a = b$  have an unique solution for all  $a, b$ . We deal here with idempotent quasigroups, satisfying the additional axiom ( $x.x = x$ ). Adding different extra axioms leads to several problem instances, fully described in [6]. None of these axioms are reducible. The results obtained with quasigroups are listed in table 1.

The results show that VESEM always explores fewer nodes than SEM. The amount of memory required to solve these problems is the same with both algorithms. Because of the cost of computing the heuristic, computation times are not significantly improved in general except on one example (QG6). Only two examples (QG7 and QG1) exhibit results slightly worse with VESEM than with SEM. Although the quasigroup problems do not clearly prove a superiority of VESEM, they show that the value elimination and lookahead strategy generally results in a favorable tradeoff and should be used.

#### 4.2 Group and Ring Problems

We compare VESEM and CTSEM and CTVESEM to SEM on a list of group and ring problems described by J. Zhang in [10]. The results are listed in table 2. Our algorithms explore fewer nodes than SEM. The lookahead strategy implemented in VESEM generally leads to improved computation times. The execution time ratio is sometimes very important: about 60 for NG and GRP.

CTSEM and VESEM not only solve problems faster, but solve problems of much higher orders (NG, GRP, RU). To the best of our knowledge, it is the first time that a program ever computes a finite model for NG34 and RU24 or proves the inconsistency of GRP38.

The program CTVESEM combining both techniques (Clause Transformation and Value Elimination) visits fewer search tree nodes. But, almost all the values suppressed (leading to skipped nodes) are due to the clause rewriting technique.

**Table 1.** Quasigroup Problems - Comparison.

Problem	Nb Model	SEM		VESEM	
		Time	Nodes	Time	Nodes
QG1 7	4	22	411	24	194
QG2 6	0	1.4	17	1.4	9
QG2 7	3	63	871	59	401
QG3 9	0	7.4	48 278	6.3	40 015
QG3 10	0	1416	7 948 372	1335	3 558 564
QG4 9	74	6.3	38 407	5.3	17 116
QG4 10	0	1263	6 946 603	1 099	2 941 094
QG5 14	0	83	320 728	53	106 703
QG5 15	0	2031	7 518 920	1306	2 251 311
QG6 11	0	40	840 542	2.3	13 690
QG6 12	0	2519	50 290 872	142	929 781
QG7 13	2	14.5	69 053	16	37 132
QG7 14	0	443	2 015 778	528	1 107 404

Thus, adding value elimination to clause transformation is redundant and results in increased computation times. All results obtained and a fully detailed description of the different algorithms described in this paper are available in [1].

## 5 Conclusion

We introduce two techniques that can be used to improve CSP approaches to finite model generation of first order theories. Their efficiency stems from the introduction of negative facts in the clause transformation technique case (CT-SEM), and from the elimination of domain values at some node of the search tree in the dynamic filtering case (VESEM).

The behaviour of the algorithms on the AG and RNA problems suggests to search for improvements in the heuristic strategy associated with the lookahead procedure in VESEM, and also to eliminate more isomorphic subspaces than is actually done with the LNH heuristic used in those programs. VESEM seems to provide the basis for a general algorithm for finite model search of first order theories.

## References

- [1] G. Audemard, B. Benhamou, and L. Henocque. Two techniques to improve finite model search. Technical report, Laboratoire d'Informatique de Marseille, 1999. accessible electronically at <http://www.cmi.univ-mrs.fr/~audemard/publi.html>.
- [2] G. Audemard, B. Benhamou, and P. Siegel. La méthode d'avalanche aval : une méthode énumérative pour sat. In JNPC, pages 17–25, 1999.

**Table 2.** Ring and Group Problems - Comparison

Problem	Nb Models	SEM		VESEM		CTSEM		CTVESEM	
		Time	Nodes	Time	Nodes	Time	Nodes	Time	Nodes
AG 28	162	328	642 103	321	76 663	336	57 941	394	41 859
AG 32	2 295	940	2 037 525	956	624 304	968	101 356	1 272	76 393
NG 28	51	6 934	8 359 103	806	108 120	432	100 036	1 105	88 832
NG 29	0	+		752	94 417	489	108 922	1 191	82 519
NG 34	3	+		5 450	504 182	3 469	478 337	+	
GRP 31	0	3 831	2 751 805	272	21 821	97	14 711	378	24 691
GRP 32	2 712	+		1 620	740 797	529	35 546	2 204	93 420
GRP 38	0	+		6 690	584 374	3 480	442 039	+	
RU 19	1	4 591	2 720 769	1729	94 326	848	197 953	1 666	15 741
RU 20	21	+		2 904	370 652	3 678	609 320	2 957	336 612
RU 24	445	+		5 029	434 006	+		5 019	366 597
RNA 14	0	592	646 421	+		354	131 355	426	45 162
RNA 15	0	592	646 421	1 021	150 538	513	144 613	682	56 623
RNA16	?	+		+		+		+	
RNB 17	0	15	13 148	20	6 389	15	2 287	21	1 309
RNB 18	0	16	13 238	36	2 171	16	2 377	34	852

- [3] B. Benhamou and L. Henocque. A hybrid method for finite model search in equational theories. *Fundamenta Informaticae*, 39(1-2):21–38, June 1999.
- [4] Chu Min Li and Anbulagan. Heuristics based on unit propagation for satisfiability problems. In *Proceedings of the 15th International Joint Conference on Artificial Intelligence (IJCAI-97)*, pages 366–371, August 23–29 1997.
- [5] Nicolas Peltier. A new method for automated finite model building exploiting failures and symmetries. *Journal of Logic and Computation*, 8(4):511–543, 1998.
- [6] J. Slaney, M. Fujita, and M. Stickel. Automated reasoning and exhaustive search: Quasigroup existence problems. *Computers and Mathematics with Applications*, 29(2):115–132, 1993.
- [7] J. Slaney. Finder : Finite domain enumerator. version 3 notes and guides. Technical report, Austrian National University, 1993.
- [8] H. Zhang and Mark E. Stickel. Implementing the davis-putnam algorithm by tries. Technical report, Department of Computer Science, University of Iowa, 1994.
- [9] J. Zhang and H. Zhang. Constraint propagation in model generation. In *proceedings of CP95*, Marseille 1995.
- [10] Jian Zhang. Constructing finite algebras with FALCON. *Journal of Automated Reasoning*, 17(1):1–22, August 1996.
- [11] Jian Zhang and Hantao Zhang. SEM: a system for enumerating models. In Chris S. Mellish, editor, *Proceedings of the Fourteenth IJCAI*, pages 298–303, 1995.