A Finite Model Search System for Equational Theories (FMSET) and applications

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
phone number : 91.11.36.22
e-mail : Benhamou@gyptis.univ-mrs.fr
e-mail : henocque@esil.univ-mrs.fr

Abstract. Finite model and counter model generation is a potential alternative in automated theorem proving. In this paper, we introduce a system called FMSET which generates finite structures representing models of equational theories. FMSET performs a satisfiability test over a set of special first order clauses called "simple clauses". The algorithm's originality stems from the combination of a standard enumeration technique and the use of first order resolution. Our objective is to obtain more propagations and still achieve good space and time complexity. Several experiments over a variety of problems have been pursued. FMSET take advantage of symmetry to prune the search space from unwanted isomorphic branches.

Topics: Finite model search, Computer Algebra Systems and Automated Theorem Provers.
Keywords: Finite model, equational theories, symmetry.

1 Introduction

Model generation is well known as a difficult problem in mathematical logic, undecidable in the general case. In this paper, we study finite model generation for equational theories.

Constraint programming tools usually involve cooperating constraint solvers, and the system's global efficiency is positively impacted by an increase in the propagations performed by the solvers, because it reduces the search space, and it improves the quality of the heuristic predictions. It is thus interesting to look for finite model generation algorithms having better propagation efficiency. The equational theories form a challenging goal in that respect and equations over functional symbols abound in constraint based models, where many relations have functional semantics. Equational theories are frequent in abstract algebra, which has been a typical application of theorem provers.

It appears that, although intractable in practice, the propositional clause representation of an equational theory generates more boolean constraint propagations than a dedicated algorithm as in [7], or than a general purpose logic programming one as in [1]. This work is an attempt to design an algorithm for equational theories expressed with first order clauses having the same propagation efficiency as the methods used in propositional calculus.

Equational theories provide a great number of difficult problems. Zhang in [6] defines a set of problems which can form a challenge of finite model search systems. Several open problems were solved with different approaches: FALCON [7], FINDER [4], MGTP-G [1], LDPP, SATO [5], and MACE [3]. FALCON system has been proved to be an efficient solver for equational theories, thus serves us as a first basis of comparison. Because their results where obtained on a powerfull parallel machine, we also compare FMSET with MGTP-G [1] on quasi group problems.

An equational theory is a set of axioms: first order logic formulas involving equality (ex: $\forall x, \forall y, \forall z : h(f(x,y)) = f(z,x)$). We consider the theories in which all the variables are universally quantified. Finding a finite model for such a theories amounts to finding an interpretation of functional symbols over a finite domain D_n which satisfies all the corresponding axioms. The existence of a model proves the consistency of the theory. The existence of a counter model may prove refutation of a conjecture. This is why model generation is a possible approach to automated theorem proving.

In this paper, we present a new finite model generator, the system FMSET (Finite Model Search for Equational Theories). FMSET first translates the set of axioms of an equational theory to an equivalent set of "simple" first order clauses. Model generation then operates by variable domain enumeration and first order resolution, in the spirit of the Davis and Putnam procedure. This enumeration is performed directly over the simple first order clauses obtained after the translation phase. It has the double advantage of: 1) generating only the propositional clauses involved in the proof, 2) achieving efficient unit clause propagation as in the propositional case using unification techniques.

Symmetry detection is used to eliminate isomorphic branches in the search tree, thus improving FMSET's performance. A heuristic for choosing literals similar to the one used in Zhang[7] helps keeping symmetries as long as possible during the search. The paper is organised as follows:

Section 2 defines equational theories. Section 3 describes the translation of equational theories to simple first order clauses. The enumeration procedure, the mechanism of propagations and the symmetry elimination procedure are discussed in section 4. Experimental results are listed in section 5. Section 6 concludes the work.

2 Equational theories

2.1 Syntax

An equational theory is a set of axioms such as $t_1 = t_2$, and $t_1 \neq t_2$, where t_1, t_2 are terms and $\{=, \neq\}$ are the equality, respectively, the disequality first order logic predicates. Each term is recursively built upon functional symbols and univesally quantified variables. The theories we consider form a restriction of the first order logic theory in which only the predicates $\{=,\neq\}$ appear and in which all the variables forming the therms are universally quantified.

2.2 Semantic

A model of an equational theory is a structure consisting in a non empty set of individuals and functions that satisfy the axioms. In this work, we only consider finite models. Such model are often represented by the operation tables of the functions. Searching for a finite model is equivalent to define the operation tables of the functions which satisfies the axioms. Finite model is characterized by the number n of individuals (values) of its domain which defines its size. The domain D_n of a model of size n is represented by the set of integers $D_n = \{0, 1, ..., n-1\}$ which represents the n individuals (values).

Example 1. Let T1 be the equational theory expressed by the following axioms:

- A1: $\forall x, h(x, x) = x$
- A2: $\forall x, \forall y, h(h(x, y), x) = y$

T_1 has a model of size 4 which is given by:

$$\begin{array}{l}
h(0,0)=0,\ h(0,1)=2,\ h(0,2)=3,\ h(0,3)=1\\
h(1,0)=3,\ h(1,1)=1,\ h(1,2)=0,\ h(1,3)=2\\
h(2,0)=1,\ h(2,1)=3,\ h(2,2)=2,\ h(2,3)=0\\
h(3,0)=2,\ h(3,1)=0,\ h(3,2)=1,\ h(3,3)=3
\end{array}$$

3 Translation of an equational theory to a set of simple first order clauses

In the sequel, except when necessary, the word *equation* designates equations as well as disequations and all the variables are universally quantified.

Definition 1. – The number of functional symbol occurrences in an equation defines its degree.
- An equation of degree 1 is a simple equation and represents a literal.
- A first order clause consisting only on disjunctions of simple equations (literals) is a simple clause.

Example 2. The clause $\forall x \forall y \forall z (f(x, y) \neq z \vee h(z, y) = x)$ is a simple first order clause which is formed by the disjunction of the literals $f(x, y) \neq z$ and $h(z, y) = x$. The literal $f(x, y) \neq z$ is the negation of the literal $f(x, y) = z$.

The translation of equational theories to simple first order clauses is based on the systematic replacement of axioms by simple equations, by means of auxiliary variables. The principle is the following: when applied to the axiom

$A_2 : (h(h(x,y),x) = y)$ from the above example T1, it produces the simple clause $h(x,y) \neq v_1 \vee h(v_1,x) = y$. The innermost term $h(x,y)$ is replaced by an auxiliary variable v_1 so as to turn the axiom into a disjunction of simple equations (literals). The result can be read "for all x, y, v_1 if $h(x,y) = v_1$ then $h(v_1,x) = y$". Applied to an axiom of initial degree 3: $f(h(x,g(y)),x) = y$, we obtain the simple clause $g(y) \neq v_1 \vee h(x,v_1) \neq v_2 \vee f(v_2,x) = y$, which can be read "for all x, y, v_1, v_2 if $g(y) = v_1$ and if $h(x,v_1) = v_2$ then $f(v_2,x) = y$".

The translation of an equational theory is performed in two steps :

- Normalisation : consists in changing the original axioms into clauses where equations have the form $t = x$, x being a variable and t a term.
- Simplification : consists in transforming the clauses resulting from the normalisation into simple clauses.

If $var(t)$ denotes the variable which replaces the term t, then the translation algorithm uses the following set of rules:

1. the equation $x = t_1$ becomes $t_1 = x$ (variable x is moved to the right)
2. the equation $x \neq t_1$ becomes $t_1 \neq x$ (variable x is moved to the right)
3. the equation $t_1 = t_2$ becomes $((t_1 \neq var(t_1) \vee t_2 = var(t_1)) \wedge ((t_2 \neq var(t_2) \vee t_1 = var(t_2))$ (a conjunct of two clauses)
4. the equation $t_1 \neq t_2$ becomes $((t_1 \neq var(t_1) \vee t_2 \neq var(t_1)) \wedge ((t_2 \neq var(t_2) \vee t_1 \neq var(t_2))$ (a conjunct of two clauses)
5. the equation $f(t_1,t_2) = z$ becomes $(t_1 \neq var(t_1)) \vee (t_2 \neq var(t_2)) \vee (f(var(t_1), var(t_2)) = z)$
6. the equation $f(t_1,t_2) \neq z$ becomes $(t_1 \neq var(t_1)) \vee (t_2 \neq var(t_2)) \vee (f(var(t_1), var(t_2)) \neq z)$

Normalisation is the result of a single application of one of the rules 1 to 4 and simplification results from the repeated application of rules 5 and 6 to the equations until a fixed point is reached. In the rules 3 to 6, $var(t)$ is t when the term t is itself a variable.

This algorithm terminates because rules 1 to 4 are applied at most once, and rules 5 and 6 strictly reduce the degree of their argument. The rules 5 and 6 are defined for functions or arity 2, but can be generalized in a natural way to functions of greater arities.

Remark. This algorithm produces simple Horn clauses, containing at most one positive literal ($t_1 \neq x$ is the negation of $t_1 = x$). Thus all the equational theories we consider are expressed in terms of simple Horn clauses, we call them Horn equational theories.

4 The enumeration procedure

We have tried two enumeration techniques inspired from the Davis and Putnam procedure.

4.1 Working with propositional clauses

Given a domain D_n, it is straightforward to translate a set of simple clauses $\mathcal{C}$ into a set of propositional clauses. It simply requires to generate the terminal instances $c[..., e_i/x_i, ...]$ for all $c \in \mathcal{C}$, and for every possible substitution $< e_i/x_i >$ where e_i belongs to D_n and x_i is a variable that occurs in c. For a domain of size n, and a clause having k variables, the number of propositional clauses produced is n^k in the worst case.

For instance, given a domain of size 2, the clause $(h(x,y) \neq v_1 \vee h(v_1,x) = y)$ representing the axiom A_2 in the theory T_1 (example 1 before) is expressed by the following set of propositional clauses:
$1 : \neg h(0,0) = 0 \vee h(0,0) = 0$ $\quad 2 : \neg h(0,0) = 1 \vee h(1,0) = 0$
$3 : \neg h(0,1) = 0 \vee h(0,0) = 1$ $\quad 4 : \neg h(0,1) = 1 \vee h(1,0) = 1$
$5 : \neg h(1,0) = 0 \vee h(0,1) = 0$ $\quad 6 : \neg h(1,0) = 1 \vee h(1,1) = 0$
$7 : \neg h(1,1) = 0 \vee h(0,1) = 1$ $\quad 8 : \neg h(1,1) = 1 \vee h(1,1) = 1$
to which the set of clauses describing the mutual exclusion of function values must be added. For instance, take h(0,0), and a domain of size 3 or more :
$h(0,0) = 0 \vee h(0,0) = 1 \vee h(0,0) = 2 \vee ...$
$\neg h(0,0) = 0 \vee \neg h(0,0) = 1$
$\neg h(0,0) = 0 \vee \neg h(0,0) = 2$
$\neg h(0,0) = 1 \vee \neg h(0,0) = 2$
...

This simple example shows that such a set of clauses allows for monoliteral propagation even in the presence of negative facts. For instance, if $h(1,1) \neq 0$ is true, clause 6 propagates $\neg h(1,0) = 1$. Such propagations are not performed by the algorithm described in [7], which only reacts to the introduction of positive facts and thus loses in search efficiency.

Infortunately the number of clauses grows quickly as the domain sizes grow, then using a classical model search procedure (like Davis and Putnam) for such sets of propositional clauses must be intractable, except when the domain sizes are small. Actually, even simple theories generate a huge number of clauses. Take for instance the following theory T_2 describing a non commutative group :

$$\begin{gathered}
h(x,0) = x \\
h(0,x) = x \\
h(x,g(x)) = 0 \\
h(g(x),x) = 0 \\
h(h(x,y),z) = h(x,h(y,z)) \\
h(1,2) \neq h(2,1)
\end{gathered}$$

With a size 6 domain, the translation of the axioms above to propositional clauses requires 252 literals and nearly 100000 clauses. Memory consumption, and the time needed to simply generate the clauses renders this approach irrealistic. This is why we prefer to use directly the simple first order clauses.

4.2 Working with the simple first order clauses

The clauses produced by the translation algorithm of section 3 contain at most one positive literal (Horn simple first order clauses). As soon as a positive literal propagates (for instance $f(0) = 0$), implied negative facts are numerous (here: $f(0) \neq 1, f(0) \neq 2$, ...). Many propositional clauses are removed immediately (because they are true), and would have been created needlessly with the first approach. It is thus realistic to envision the dynamic creation of the only useful propositional clauses.

Our enumeration procedure $FMSET$ is described by the recursive function of figure 1 where the stack *pmod* contains the current partial model.

```
function FMSET(F : a set of clauses):boolean;
begin
    for each literal b on the stack pmod, Assign(b);
    if the empty clause appeared then return(false);
    if all the clauses in F are satisfied then return(true);
    choose next literal a
    push a on the stack pmod
    if FMSET(F) return(true)
    push ¬a on the stack pmod
    return(FMSET(F))
end
```

Fig. 1. The enumeration procedure

We use an intermediate representation between first order and propositional clauses, producing exactly the same propagations as the latter, at a lower cost. Propagations occur when a clause is shortened so that its length becomes equal to one. To obtain these propagations without using propositional clauses, it is enough to keep the simple clauses which generate the clauses shortened by the current interpretation.

For instance : let c_1 be the clause $f(z) \neq y \vee h(x) \neq y \vee g(y) = x$. When the literal $f(0) = 1$ becomes true, we generate the substitution $\sigma = \{< 1/y >, < 0/z >\}$ with which we produce the "pseudo" clause $c_2 : h(x) \neq 1 \vee g(1) = x$. Later on, when $g(1) = 3$ becomes false, we build $c_3 : h(3) \neq 1$ of length one. This forces the propagation of $h(3) \neq 1$.

More interesting, the simple clause $c_4 : f(y) \neq z \vee h(x) = y$, when the literal $f(0) = 0$ becomes true, produces the monoliteral clause $c_5 : h(x) = 0$ which forces the propagation of the literals: $h(0) = 0, h(1) = 0, h(2) = 0, ..., h(n-1) = 0$.

The simple clauses generated as a result of literal propagation are called pseudo clauses in FMSET. They are well described by a substitution of part of the set of variables of the original simple clause by values in the domain.

Formally:

Definition 2. Let $c \in \mathcal{C}$ be a simple clause, V_c the set of variables occuring in c, V a subset of V_c, and σ a substitution $\{< e_i/x_i >| \ e_i \in D_n, x_i \in V\}$. The application of σ to the variables of c produces the pseudo clause c_σ.

Initially, every simple clause c maps to the pseudo clause $c_\emptyset$ where $\emptyset$ is the empty substitution.

Definition 3. Two substitutions σ_1 et σ_2 are incompatible iff there exists $< a/x >\in \sigma_1$ and $< b/x >\in \sigma_2$ such that $a \neq b$, compatible otherwise.

```
Procedure Assign(a ∈ literals)
begin
      for each pseudo clause c_σ1
          for each simple equation s ∈ c_σ1
              if unifiable(s,a) then
              begin
                    let σ2 be the substitution due to unification of a and s
                    if σ1 and σ2 are compatible then
                    begin
                          build the pseudo clause ć = (c_σ1 − s)_σ1∪σ2
                          if ć is of length 1 then
                            push on stack pmod all the monoliterals due to ć
                          else if not a tautology(ć) then
                            add ć
                    end
              end
end
```

Fig. 2. Propagation procedure

The function $FMSET$ described in figure 1 calls the propagation procedure $assign$ described in figure 2 in which the technique of unification is used.

4.3 Removing symmetries

Many symmetries exist in equational theories. They slow down the algorithms because of unwanted exploration of isomorphic branches within the search space.

As it was done in [7], some symmetries can be removed at low cost because they are due to trivial symmetries in the set of domain individuals. Especially, at the beginning of the search, all individuals in the domain are interchangeable.

Definition 4. A set of simple first order clauses $\mathcal{C}$ is symmetrical with respect to a subset D_{sym} of the domain D_n iff it remains unchanged under any permutation of the individuals in D_{sym}.

In particular, if D_n is the set of integers $\{0, 1, \ldots, m, m+1, \ldots, n-1\}$ and $\{0, 1, \ldots, m\}$ is the subset of D_n used by the literals in the current partial model *pmod* that were set by a choice point, then the property of symmetries is characterised by the following theorem :

Theorem 5. *The set of simple clauses $\mathcal{C}_{pmod}$ generated by the partial model pmod from $\mathcal{C}$ is symmetrical with respect to $D_{sym} = \{m+1, \ldots, n-1\}$*

Proof. There are only two possibilities of occurrence of an individual bigger than m in *pmod*:
- because of the assignment of a positive literal of the form $t = k$, with $k \leq m$. In that case, for all $k2 > m$, the literal $t \neq k2$ is also in *pmod*. Hence this propagation keeps the symmetry of *pmod* w.r.t $D_{sym} = \{m+1, \ldots, n-1\}$.
- because of the propagation of a simple clause of length one. Such a clause being universally quantified, the propagation naturally maintains the symmetry of *pmod* w.r.t $D_{sym} = \{m+1, \ldots, n-1\}$. Thus, in both cases, the system $\mathcal{C}_{pmod}$ is itself symmetrical w.r.t D_{sym}.

This theorem is exploited in FMSET by a cut in the search tree. Let m be the maximum integer used in the choosen literals for assignment in *pmod*. The cut operates as follows : when a terminal literal $t = a$ (no occurence of variables in t and a is a value of D_n) is selected for assignment, we compute $m_{t=a}$ the greatest individual involved in the literal $t = a$ (for instance 2 in the terminal literal $f(1,2) = 0$), we set $m = max(m_{t=a}, m)$, and add to *pmod* the negative literals $t \neq l$ for all $l \in [m+2, n-1]$. This cuts the subspace corresponding to values of l greater than $m+1$. This gives a total of $n - m - 2$ cuts in the search tree corresponding to values (individuals) in $[m+2, n-1]$ which are all symmetrical with the value $m+1$.

Assignment of negative literals $(v \neq l)$ may induce more literal propagations which prune the search tree.

The Least Number Heuristic Plus (LNHP):

We use a variant of the Least Number Heuristic (LNH) defined in [7] which consists in choosing variables which keep the symmetries as long as possible. These variables are the ones which do not involve individuals over m. When this fails, it favors the variables that minimise the increase of m. Among several possibilities, it takes one variable wich has the smallest domain size after the cuts.

5 Experimentation

5.1 The first group of problems

Description of the problems: the first group of problems that we have experimented are described in [7]. AG is an abelian group and the source code for AG is listed in figure 3. NG is a non commutative group. RU is a ring with unit (see figure 4). GRP is a non commutative group satisfying the additional axiom $(xy)^4 = x^4y^4$. RNA is a ring, plus a counter example of associativity $p(2, p(3, 4)) \neq p(p(2, 3), 4)$. In this case, the existence of models proves the independence of the associativity law. RNB is a non boolean ring plus the axiom $x^7 = x$, listed in figure 5.

h(x,0) = x
h(0,x) = x
h(x,g(x)) = 0
h(g(x),x) = 0
h(h(x,y),z) = h(x,h(y,z))
h(x,y) = h(y,x)

Fig. 3. AG problem code

s(x,0) = x
s(x,m(x)) = 0
s(x,y) = s(y,x)
s(x,s(y,z)) = s(s(x,y),z)
p(x,1) = x
p(x,y) = p(y,x)
s(p(x,y),p(x,z)) = p(x,s(y,z))
s(p(x,z),p(y,z)) = p(s(x,y),z)

Fig. 4. RU problem code

These six problems are not open problems, but provide a good comparison basis for FMSET, FALCON, MACE and FINDER.

Comparative results with MACE, FINDER and FALNCON: FMSET is developped in C++, the backtrack function FMSET of the previous section is implemented in its iterative version for reason of efficiency, our computation times in seconds are obtained on a Pentium 133. All of MACE, FINDER and FALCON are writen in C, the corresponding CPU times are the ones given in [7] which are obtained on a Sparc 2, $(t_0 + t_1)$ for both MACE and only t_1 for FALCON (t_0 : preprocessing time and t_1 : search time).

It is noteworthy that our execution times (tables 1,2) compare well to FALCON's times and some times better, and far outperform the ones of MACE and FINDER, which fail to conclude on several of these problems. The time spent generating the clauses in FMSET is negligible, never above 0.1 second, and is

s(x,0) = x
s(x,m(x)) = 0
s(x,y) = s(y,x)
s(x,s(y,z)) = s(s(x,y),z)
p(x,p(y,z)) = p(p(x,y),z)
s(p(x,y),p(x,z)) = p(x,s(y,z))
s(p(x,z),p(y,z)) = p(s(x,y),z)
p(1,1) $\neq$ 1
s(x,x) = 0
p(x,p(p(x,x),p(p(x,x),p(x,x)))) = x

Fig. 5. RNB problem code

integrated in the computation time. Note that '*' indicates that the program runs out of memory, and '+' indicates that the execution time of the program exceeds one hour.
Our results mention the number of branches, an indicator of FMSET's propagation efficiency. This number is always very small, compared to the theoretical complexity induced by the number of literals. None of these results involve the use of a ad hoc heuristic.

Table 1 displays results obtained on several problems for which all the models are computed with respect to elimination of some isomorphic models du to the presence of symmetry. From this table, we see that FMSET's computation times are close to those of FALCON and better than those of both MACE and FINDER.

In table 2, the execution times are given for the model existence problem (N denotes no existence and Y affirms existence of models). From this table, we observe a greater variability in the results, which proves that FMSET beats difinitively both MACE and FINDER and still concurent to FALCON. Thus FMSET offers a potential algorithmic alternative .

5.2 The second group of problems

Description of the problems: A quasi group (Dénes and Keedwel, in [2]) is a set on which is defined a binary operation "." (we also use the function name "p" in the sequel) such that the equations $a.x = b$ and $x.a = b$ have unique solutions for all a and b . Idempotent quasi groups satisfy the axiom $a.a = a$ for all a in addition to the previous one. Given the axioms of an idempotent quasigroup, it is interesting to study the spectrum of additional axioms. Problems QG3 to QG7 are idempotent quasi groups plus one added axiom (we use the problem numbering from MGTP-G [1]). Problem QG3 additionally includes the axiom $xy.yx = x$ (known as Schroeder's second law). Problem QG4 additionally includes the axiom $yx.xy = x$ (known as Stein's third law). Problem QG5 additionally includes the axiom $(xy.y)y = x$. Problem QG6 additionally includes

Problems	MACE	FINDER	FALCON	FMSET + LNHP			
	time (sparc 2)	time (sparc 2)	time (sparc 2)	time (pc 133)	number of models	literals	branches
Abelian groups							
AG.4	0.37	0.07	0.05	0.13	3	80	2
AG.5	1.44	0.39	0.22	0.27	1	150	9
AG.6	6.50	8.95	0.68	1.03	6	252	22
AG.7	28.78	253,20	1.97	2.23	1	392	40
AG.8	493.24	+	4.73	7.5	13	576	63
AG.9				12.8	3	810	101
Non commutative groups							
NG.4	0.37	0.07	0.05	0.08	0	80	6
NG.5	1.36	0.52	0.17	0.15	0	150	14
NG.6	5.13	12.95	0.48	0.56	2	252	24
NG.7	13.30	530.63	1.30	0.63	0	392	29
NG.8	198.28	+	3.37	1.5	1	576	41
NG.9	*	+	7.70	3.1	0	810	75
Unit rings							
RU.4	3.95	0.46	0.25	0.6	5	144	2
RU.5	19.64	6.00	1.27	1.4	1	275	8
RU.6	83.54	2078.23	3.50	7.5	1	468	29
RU.7	*	+	9.43	18.8	1	735	44

Table 1. Results obtained when counting all models

the axiom $xy.y = x.xy$ (Schroeder's first law), and problem QG7 additionally includes the axiom $yx.y = x.yx$. More details on casi group problems can be found in Benett and Zhu [8]

Because the input language of FMSET is currently restricted to equational theories, the quasi group axioms must be coded via two auxiliary functional symbols, representing two conjugates of the quasi group operation p. This restriction also forbids to code the problems QG1 and QG2 from [1]. As an example, the source code for problem QG3 is listed in figure 6. In this source code, the function p is the quasi group operation ".", whereas q and r are two auxiliary conjugate functions needed to encode the quasi groups axioms.

The rationale behind the use of these two conjugate functions is the following : the equation $p(x, a) = b$ has a unique solution iff there exists a function r such that $p(x, y) = z$ is equivalent to $r(z, y) = x$, for all x, y, z. This equivalence is obtained by two axioms: the axiom 3 in figure 6 means $r(x, y) = z$ implies $p(z, y) = x$, whereas the axiom 4 means $p(x, y) = z$ implies $r(z, y) = x$.

Comparative results with MGTP: The data listed in the table 3 for the MGTP program were obtained on a parallel machine running the parallel logic programming language KL1. The execution times are taken from [1]. The ma-

Problems	MACE	FINDER	FALCON	FMSET + LNHP			
	time (sparc 2)	time (sparc 2)	time (sparc 2)	time (pc 133)	model exists ?	literals	branches
Non commutative groups satisfying $(xy)^4 = x^4y^4$							
GRP.4	40.22	0.07	0.08	0.58	N	112	6
GRP.5	309.16	1.82	0.27	1.78	N	200	14
GRP.6	*	499.78	0.65	9.76	N	324	26
GRP.7		+	1.53	15.9	N	490	29
GRP.8		+	0.58	*	*	*	*
ring not associative							
RNA.5	23.60	+	24.72	5.43	N	275	28
RNA.6	*		107.72	52.4	N	468	172
RNA.7			562.97	426	N	735	605
RNA.8			350.87	11.3	Y	1088	9
non boolean ring							
RNB.4	3.81	3.32	0.2	0.46	Y	144	3
RNB.5	18.33	0.24	0.68	0.22	N	273	4
RNB.6	67.83	2.11	1.67	0.45	N	468	4
RNB.7	*	3.58	3.37	3.7	N	735	12
RNB.8	*	+	4.22	10.9	Y	1088	3
RNB.9				23.5	N	1539	34

Table 2. Results obtained for model existence

1: $p(x, q(x, z)) = z$
2: $q(x, p(x, z)) = z$
3: $p(r(x, y), y) = x$
4: $r(p(x, y), y) = x$
5: $p(a, a) = a$
6: $p(p(a, b), p(b, a)) = a$

Fig. 6. QG3 problem code

chine has 256 processors and the or-parallelism involved in the problems permits to achieve almost linear speedup. We have reproduced the conditions in [1], specially with the implementation of a similar static symmetry elimination procedure (SSE) : we post the additional constraint "for all a $p(O, a) \leq a + 1$". We use a very common variable choice heuristic, selecting the one with the smallest domain size.

Both FMSET and MGTP branches do not count trivial fails. Hence the results in table 3 show that FMSET solves the quasi group problems with a lot fewer branches than MGTP, and in the average outperforms MGTP in cpu-time. Most of the problems experimented in table 3 used to be open problems which

Problems	MGTP		FMSET + SSE		
	time	branches	time	branches	number of models
$xy.yx = x$					
QG3.7	6	183	0.65	73	0
QG3.8	28	3875	9.9	1428	18
QG3.9	1022	312321	626	74392	0
$yx.xy = x$					
QG4.7	6	123	0.9	95	0
QG4.8	23	3516	17	1783	0
QG4.9	1127	314925	1040	69238	178
$(xy.y)y = x$					
QG5.7	3	9	0.4	21	3
QG5.8	7	34	0.6	45	1
QG5.9	12	239	2.3	212	0
QG5.10	66	7026	17	1743	0
QG5.11	224	51904	1197	169647	5
$xy.y = x.xy$					
QG6.7	2	7	0.3	4	0
QG6.8	6	20	0.54	13	2
QG6.9	14	160	1.2	52	4
QG6.10	43	2881	4	238	0
QG6.11	248	50888	40	2013	0
QG6.12	8300	2420467	503	19382	0
$yx.y = x.yx$					
QG7.7	4	182	0.47	17	0
QG7.8	5	160	1.5	106	0
QG7.9	90	37026	7.3	456	1
QG7.10	2809	1451992	51	3290	0
QG7.11			39.8	1797	0

Table 3. Results obtained by FMSET and MGTP

the sytem MGTP [1] solved for the first time.

6 Conclusion

We have implemented a system for searching finite models for equational theories. It is based on the direct use of first order clauses within a standard enumeration algorithm and some techniques of resolution. This hybrid approach produces more propagations, without the cost of either the propositional clauses, or the full first order logic. In a sense, the algorithm combines model enumeration and unification based resolution for simple first order clauses. In that framework, symmetries can be efficiently used and improve FMSET's efficiency. Our results compare well with the well known systems FALCON and MGTP.

The input language to FMSET is being extended to accept generalized first order clauses. This feature will augment the range of problems that FMSET can deal with, and specially allow more straightforward handling of the quasi group examples. Thus, FMSET will be integrated as dedicated algorithm in a first order theorem prover with constraints currently under development. We plan to use this approach for other categories of finite problems for which the translation to propositional logic is not tractable.

Facing the evidence that all isomorphic models are not eliminated by the symmetries, we currently study the possibility to detect other symmetries in the set of simple clauses and exploit them to increase the efficiency of search.

References

1. M. F. . J. S. . F. Bennett. Automatic generation of some results in finite algebra. *In proceedings of the 13th Internationnal Joint Conference on Artificial Intelligence, Chambery, France*, pages 52–57, 1993.
2. J. D. . A. Keedwell. Latin squares and their applications. Technical report, Academic Press, New York, 1974.
3. W. McCune. A davis putnam program and its application to finite fist order model search : quasi-group existence problems. Technical Report ANL/MCS-TM-1994, Technical report, Argonne National Laboratory, 1994.
4. J. Slaney. Finder: Finite domain enurator. version 3.0 notes and guide. Technical report, Austrian National University, 1993.
5. H. Z. . M. Stickel. Implementing the davis and putnam algorithm by tries. Technical report, Technical report, University of IOWA, 1994.
6. J. Zhang. Problems on the generation of finite models. *in proceedings of CADE-12, Nancy, France*, pages 753–757, 1994.
7. J. Zhang. Constructing finite algebras with falcon. *Journal of automated reasoning, 17*, pages 1–22, 1996.
8. F. B. . L. Zhu. Conjugate-orthogonal latin squares and related stuctures. Technical report, Contemporary Design Theory: A Collection of Surveys, ed J.H Dinitz, D.R. Stinson. New York, Wiley, 1992.