

Reasoning by Symmetry and Function Ordering in Finite Model Generation

Gilles Audemard^{1,2} and Belaid Benhamou²

¹ ITC-IRST, via Sommarive 16, 38050 Povo, Trento, Italy

² Laboratoire des Sciences de l'Information et des Systèmes de Marseille (LSIS)
39, Rue Joliot Curie - 13453 Marseille cedex 13 - France
{audemard, benhamou}@cmi.univ-mrs.fr

Abstract. Finite model search for first-order logic theories is complementary to theorem proving. Systems like Falcon, SEM and FMSET use the known LNH (Least Number Heuristic) heuristic to eliminate some trivial symmetries. Such symmetries are worthy, but their exploitation is limited to the first levels of the model search tree, since they disappear as soon as the first cells have been interpreted. The symmetry property is well-studied in propositional logic and CSPs, but only few trivial results on this are known on model generation in first-order logic.

We study in this paper both an ordering strategy that selects the next terms to be interpreted and a more general notion of symmetry for finite model search in first-order logic. We give an efficient detection method for such symmetry and show its combination with the trivial one used by LNH and LNHO heuristics. This increases the efficiency of finite model search generation. The method SEM with and without both the function ordering and symmetry detection is experimented on several interesting mathematical problems to show the advantage of reasoning by symmetry and the function ordering.

1 Introduction

A finite model of a first-order theory is an interpretation of the variables, the function symbols and the predicates over a finite domain of individuals that satisfies the axioms of the theory. A finite model generator is an automated tool which computes such interpretations. Model generation can be used to prove the consistency of a first-order theory (i.e., a model exists) or to find a counter model to disprove its validity. It can find a counter example of some expected conjecture and can help theorem provers to refute validity of formulas. In this sense, finite model generation is complementary to theorem proving.

Symmetry elimination is crucial in finite model generation. To get efficient model generators one has to detect and exploit symmetrical structures. Different approaches to symmetry elimination are used: For instance, the static approach used by James Crawford et al. in [5] for propositional logic theories consists of adding constraints expressing symmetry. The same technique is used by Masayuki Fujita et al. in [6] to search for finite models of quasigroup problems.

One drawback of this approach is that only the structural symmetries appearing in the initial problem are considered.

The method FMC [7] eliminates some symmetries during the search to avoid isomorphic interpretations. Finite model generators like SEM [12] and FMSET[2] use the LNH (Least Number Heuristic) heuristic [12] to suppress some trivial symmetries existing between individuals of the domains. The previous methods use the *CNF* form of first-order logic where all the variables are universally quantified to express problems. As a consequence of this quantification, the individuals in the domains are all trivially symmetrical. That is, if during the search all the first individuals $\{0, \dots, mdn\}$ of an ordered finite domain $D_n = \{0, \dots, n-1\}$ (with $0 \leq mdn \leq n-1$) are used in the partial interpretation, then only the individuals of the part $\{mdn+1, \dots, n-1\}$ remain symmetrical. All the individuals are symmetrical before starting the model search ($mdn = 0$). To keep a maximum number of symmetrical individuals, the LNH heuristic assigns in priority cells whose individuals have already been used.

Such trivial symmetries are very useful but they disappear as soon as the first cells are interpreted. The propagation process forces new individuals to be used, thus increases the *mdns* and decreases the subset $\{mdn+1, \dots, n-1\}$ of symmetrical individuals. This subset becomes empty when *mdn* reaches the value $n-1$. On the other hand, the subset $\{0, \dots, mdn\}$ of the used individuals increases and become quickly identical to the whole domain D_n .

A lot of other symmetries exist between individuals of the part $\{0, \dots, mdn\}$ which can be exploited to increase the efficiency of model generation. In this article, we show how to detect and use such symmetries in combination with a function strategy that selects the terms to be interpreted and which preserves the trivial symmetries. This ordering gives a new heuristic that we call “LNHO” which improves the LNH one. We also show how to combine the detected symmetries with the trivial ones exploited by both LNH and LNHO heuristics to maximise the symmetry gain.

This article is organised as follows: In section 2, we give a short description of many-sorted first-order logic theories. In section 3, we study the principle of symmetry for finite model generation and prove that the trivial symmetry is a particular case of this study. We describe how to exploit such symmetry in section 4. Section 5 gives a short description of the method SEM. We describe in section 6 the function ordering strategy that we use in SEM to select the next terms to be interpreted. Section 7 presents an experimental evaluation of the known finite model generator SEM method with and without both advantages of symmetry and the function ordering on mathematical problems.

2 Background: Many-sorted First-Order Logic Theories

2.1 Syntax

As a knowledge representation, we use the many-sorted first-order logic with equality in clausal normal form (CNF) where all the variables are universally quantified. Formally, a many-sorted theory is a triple $T = (S, F, C)$ where S is the

set of sorts, F is the set of function symbols and C is the set of first order clause axioms. A function symbol $f \in F$ of arity k is specified by $f : s_1 \times \dots \times s_k \mapsto s$ where s_1, \dots, s_k, s are in S . Constants are considered as 0-arity functions. Terms are built recursively with the function symbols and the variables. Predicates are considered as function symbols with the Boolean target sort. Problems are then expressed by the set C of first-order logic clause axioms. A clause is a disjunction of literals. A literal is an equation $t_1 = t_2$ or its negation $t_1 \neq t_2$ where t_1 and t_2 are terms or any predicate p . We denote by C^t the expansion of the original theory axioms of C to the set of their ground instances (i.e. ground clauses).

2.2 Semantic

We consider only *finite* interpretations of a given theory T , i.e., the domain D_s of each sort $s \in S$ is finite. Without loss of generality, the corresponding domain to a sort $s \in S$ of cardinality n , is represented by the set $D_s = \{0, \dots, n-1\}$ of natural numbers.

An interpretation of a function term $f : s_1 \times \dots \times s_k \mapsto s$ is entirely specified by giving a value (in D_s) to each ground term $f(e_1, \dots, e_n)$, called *cells*, where (e_1, \dots, e_n) is in $D_{s_1} \times \dots \times D_{s_n}$. When it is not confusing, constants can be represented by some individuals of the domains. An interpretation is total if each cell is associated to a value, it is partial otherwise.

The set C^t of ground clauses is obtained by substituting each variable of the sort $s \in S$ in the terms by all the possible individuals in D_s . If I is a partial interpretation, then C_I^t denotes the remain set of ground clauses under the interpretation I (we say C^t simplified by I), and T_I the resulting theory.

Example 1. Let be a theory $T = (\{s\}, \{h\}, C)$ where C contains the two following axioms.

- $\forall x, \quad h(x, x) = x$
- $\forall x, \forall y \quad h(h(x, y), x) = y$

Suppose $D_s = \{0, \dots, 3\}$. This theory admits a lot of models, among which the one given by the function table of h (denoted I).

h	0	1	2	3	
0	0	2	3	1	
1	3	1	0	2	This is interpreted as: $h(0, 0) = 0, h(0, 1) = 2, \dots$ etc.
2	1	3	2	0	
3	2	0	1	3	

The set of ground instances is $C^t = \{h(0, 0) = 0, h(1, 1) = 1, \dots, h(h(0, 0), 0) = 0, h(h(1, 0), 1) = 0, h(h(0, 1), 0) = 1 \dots\}$.

3 Permutations and Symmetries

First, we give some basic notions on permutations. We then define the symmetry principle for individuals of the domains.

Definition 1. Given a finite set E , a permutation σ of E is a bijection mapping from E onto E . We note $Perm(E)$ the set of all permutations of E .

The pair $(Perm(E), \circ)$ forms the permutation group of E . The composition of two permutations and the converse of a permutation are permutations. If σ is a permutation then σ^k denotes k compositions of σ . The order of a permutation σ is the smallest integer n such that $\sigma^n = Id_E$ (Id_E denotes the identity mapping). The permutation σ defined as $\sigma(a_1) = a_2, \sigma(a_2) = a_3 \dots \sigma(a_n) = a_1$ forms a cycle on the elements $\{a_1, \dots, a_n\}$ of E . We denote by (a_1, \dots, a_n) such a cycle, where n is its order. A permutation σ can be viewed as a composition of disjoint cycles.

Definition 2. A permutation σ on individuals of the domains of the set of sorts S is a tuple $\sigma = (\sigma_1, \dots, \sigma_n)$ where each $\sigma_i \in Perm(D_{s_i})$.

Definition 3. Let $f \in F$ be a function symbol specified by $f : s_{i_1} \times \dots \times s_{i_k} \mapsto s_j$ and σ a permutation defined on the individuals of the domains of the corresponding sorts. If $f(e_{i_1}, \dots, e_{i_k})$ is a cell then $\sigma(f(e_{i_1}, \dots, e_{i_k})) = f(\sigma_{i_1}(e_{i_1}), \dots, \sigma_{i_k}(e_{i_k}))$.

Therefore, the permutation σ of individuals can be easily generalised in the natural way to the set of ground clauses C^t of the theory T and to any partial interpretation I . This results from the individuals permutation in the ground terms forming C^t (resp. I). We now give the definition of symmetry.

Definition 4. Let $T = (S, F, C)$ be a theory, I a partial interpretation, C_I^t the set of ground clauses corresponding to C which is simplified by I , and σ a permutation defined on the domains of the sorts of S . The permutation σ is a symmetry of T_I if the following two conditions hold:

1. $\sigma(I) = I$, and
2. $\sigma(C_I^t) = C_I^t$

Definition 4 gives new conditions of symmetry in comparison to the propositional case [3]. To verify the symmetry conditions at a given node of the search tree, the permutation has to leave invariant both the partial set of ground clauses (C_I^t) and the partial interpretation I . Indeed, a partial interpretation is formed by a set of equations such as $f(e_1, \dots, e_k) = e_j$, where $e_i \in D_{s_i}$, and which are generally sensitive to individual permutation. This is different from the propositional case where the partial interpretation is not concerned by the permutation.

Definition 5. Let $T = (S, F, C)$ be a theory. Two individuals e_1 and e_2 of a sort domain D_s are symmetrical if there exists a symmetry σ of individuals of the domain D_s such that $\sigma(e_1) = e_2$.

Definition 6. Let $T = (S, F, C)$ be a theory. Two interpretations I and J are symmetrical if there exists a symmetry σ such that $\sigma(I) = J$.

If I and J are symmetrical interpretations ($\sigma(I) = J$) of the theory T then I is a model of T if and only if J is a model of the theory T too.

3.1 Symmetry Detection

The symmetry detection algorithm consists in two steps: The first one partitions the individuals of the different sorts into primary classes with respect to the necessary conditions of Proposition 1. Two individuals will be candidates to symmetry if they are in the same primary class. The second step is a backtrack search which builds the symmetry.

Necessary Conditions

To be symmetrical, two individuals of a same domain have to satisfy some necessary conditions (Proposition 1).

Definition 7. *If I is a partial interpretation of the theory $T = (S, F, C)$, e an individual of a domain D_s and $f \in F$ a function symbol, then $\#_{I_{f_i}}(e)$ denotes the number of occurrences of e in I as the i th position argument of the function f and $\#_{I_{f_{val}}}(e)$ denotes the number of occurrences of e in I as a value of the function f .*

Example 2. In example 1, we have $\#_{I_{h_1}}(0) = 4$, $\#_{I_{h_2}}(0) = 4$, and $\#_{I_{h_{val}}}(0) = 4$. In this interpretation we have $\#_{I_{h_i}}(a) = 4$, $\forall i \in \{1, 2\}$, $\forall a \in \{0, 1, 2, 3\}$ and $\#_{I_{h_{val}}}(a) = 4$, $\forall a \in \{0, 1, 2, 3\}$.

Proposition 1 (Necessary Conditions). *Let $T = (S, F, C)$ be a theory and a and b two individuals of a domain D_s . Let I be a partial interpretation of T . If the individuals a and b are symmetrical then the following condition hold:*

$$\begin{aligned} & - \forall f \in I \cap F, \quad \forall i \in \{1, \dots, \text{arity}(f)\}, \quad \#_{I_{f_i}}(a) = \#_{I_{f_i}}(b) \\ & - \forall f \in I \cap F, \quad \#_{I_{f_{val}}}(a) = \#_{I_{f_{val}}}(b). \end{aligned}$$

Proof. Let us prove the second condition. Let I be a partial interpretation of the theory T , σ a symmetry of T_I , and two individuals a and b of a domain D_s such that $\sigma(a) = b$. Suppose there exists a function f such that $\#_{I_{f_{val}}}(a) \neq \#_{I_{f_{val}}}(b)$. Thus we get $\sigma(I) \neq I$ since the number of occurrences of b as a value of f is different from the one of a . We can then deduce that σ is not a symmetry and we obtain a contradiction. The proof of the first condition can be done in a similar way. \square

Verification of the necessary conditions is an important step of the symmetry detection algorithm. They allow to reduce drastically the permutation search space, since they form domain partitions including the equivalence classes of individuals which are potential candidates for symmetry.

Symmetry Search (verification of the sufficient condition)

In the second step, we compute the permutation (the symmetry) using the equivalence classes of the previous step. A symmetry in propositional calculus

[3] is a variable permutation under which the set of clauses remains invariant. Such condition can be time consuming when the number of clauses grows. This condition is simplified in finite model generation. Indeed, if the set of clauses contains no individual constant, it is sufficient to look for permutations which leave invariant the partial model. The invariance of the partial set of clauses is guaranteed. This avoids checking its invariance. Formally:

Proposition 2. *Let $T = (S, F, C)$ be a theory, I a partial interpretation and σ a permutation on the domains of the sorts in S . If C contains no individual of any sort $s \in S$ and if $\sigma(I) = I$ then $\sigma(C_I^t) = C_I^t$.*

Proof. If C does not contain individuals then the permutation σ satisfies $\sigma(C^t) = C^t$. Thus, $\sigma(C_I^t) = \sigma(C^t)_{\sigma(I)} = C_{\sigma(I)}^t = C_I^t$. \square

Proposition 2 gives an important result for symmetry detection, which simplifies the sufficient condition of Definition 4. This increases drastically the efficiency of symmetry detection.

Remark 1. If some individuals appear in the set of clauses $C' \subset C$ of the theory T , then the permutation σ has to leave the subset $C' \subset C$ of clauses where the individuals occur invariant in addition to the invariance of the partial interpretation I (i.e. $\sigma(I) = I$ and $\sigma(C'^t) = C'^t$). The invariance of C'^t is checked in a preprocessing phase before starting the model search process.

We show now that the trivial symmetry (the one existing between the individuals not used so far) treated by the LNH heuristic is a particular case of our symmetry notion.

Proposition 3. *If $T = (S, F, C)$ is a theory, I a partial interpretation and σ a trivial symmetry, then $\sigma(I) = I$.*

Proof. The trivial symmetry σ is formed by the cycle $(mdn + 1, \dots, n - 1)$ of the unused individuals. The permutation σ does not permute the individuals $\{0, \dots, mdn\}$ which are the only ones used in I . I is independent of the permutation σ , thus $\sigma(I) = I$. \square

As the condition $\sigma(I) = I$ holds for each trivial symmetry σ then the symmetry exploited by the LNH heuristic is a particular case of our symmetry study (proposition 2). Such symmetries are very important, since they do not need detection. By the same way, we can show that the symmetry treated by the extension *XLNH* [1] of LNH is also a particular case of our study. Algorithm 1 gives a short description of the symmetry detection method which is a backtrack procedure. Such algorithm has an exponential complexity in the worst case. But, in practice, when symmetry exists, it is computed with few backtracks and does not affect the run time of model generation. In theory the problem of symmetry detection is equivalent to the graph isomorphism detection [4] that still has an unknown complexity (between P and NP).

In Algorithm 1, the symbol $Cl(e, D_s)$ denotes the equivalence class of the individual $e \in D_s$ with respect to the necessary conditions (Proposition 1), and the function $\text{Partition}(D_s)$ computes the equivalence classes of the individuals of D_s . The symbol $Mdn(D_s)$ denotes the Mdn number of D_s .

Proposition 5 gives an important result that we exploit to reduce the search space of a finite model generation. That is, if the interpretation $ce = e$ is contradictory then the interpretation $\sigma(ce = e)$ is contradictory too. We avoid then exploring the branch corresponding to the assignment $\sigma(ce = e)$ (i.e. $\sigma(ce) = \sigma(e)$). That is what we call the symmetry cut.

Now, we prove that a symmetry of order k can make $k - 1$ cuts in a search tree. To be efficient, one has to detect symmetries with great orders.

Proposition 6. *Let $T = (S, F, C)$ be a theory, σ a permutation of individuals of the domains and I a partial interpretation. If $\sigma(I) = I$ then $\forall j \in \mathbb{N} \quad \sigma^j(I) = I$.*

Remark 2. If a partial interpretation I of the theory $T = (S, F, C)$ is invariant under σ ($\sigma(I) = I$) then $\forall j \in \mathbb{Z} \quad \sigma^j(C_I^t) = C_I^t$.

Now we can generalise the result of proposition 5 to any power of σ . Formally:

Proposition 7. *If a partial interpretation I of the theory $T = (S, F, C)$ is consistent then [the extension $I \cup \{ce = e\}$ is consistent if and only if $\forall j \in \mathbb{N}, \quad I \cup \{\sigma^j(ce = e)\}$ is consistent]*

Proof. A trivial proof can be derived from propositions 5 and 6. □

Proposition 7 gives a good exploitation of the symmetry σ . For each symmetry of order k , we cut $k - 1$ branches in the search tree. Indeed, when the assignment $ce=e$ is contradictory, all the assignments $\sigma^j(ce=e)$ for all $j \in \{1, \dots, k-1\}$ are contradictory too. This leads to $k - 1$ symmetry cuts.

Many symmetries can be detected for a given theory. But, for efficiency reasons, we detect and exploit only one at each node of the search tree. This symmetry is combined with the trivial ones treated by both the LNH and LNHO heuristics to maximise the gain.

Combination of the detected symmetry (DSYM) with the trivial one

Consider a theory with one sort whose individuals are in $D_n = \{0, \dots, n-1\}$ and I the partial interpretation. If the individuals used (appear in some terms of I) at the current node of the search tree are all in the part $\{0, \dots, mdn\}$, ($0 \leq mdn \leq n-1$), then all the extensions $I \cup \{t = mdn+1\}, \dots, I \cup \{t = n-1\}$ are trivially symmetrical. These trivial symmetries are exploited with respect to both the LNH and the LNHO heuristics. We add to these the symmetries that we detect on the part $\{0, \dots, mdn\}$ of used individuals. The trivial symmetry and the one we detect here are independent, since they are defined on two disjoint parts of the domain. Their combination is then straight forward and $n - mdn - 1 + k$ symmetry cuts can be made when they are associated.

5 SEM Description

As an initial preprocessing stage, SEM expands the original theory axioms to the set of their ground instances (i.e. ground clauses). SEM's efficiency relies on an

efficient encoding of the terms and a direct access to the leaf terms. This allows fast upward propagation of cell assignments into embedded terms. The memory resources required by this transformation are far smaller than those needed by a propositional approach. In the latter case, the same kind of expansion is needed, but clause “flattening” (see [2] for instance) requires additional auxiliary variables at an obvious cost.

SEM naturally treats many-sorted theories as particular constraint satisfaction problems (CSP) where the variables are the cells, their associated domains are sorts and the constraints are specified by the resulting set of ground clauses.

Algorithm 2 SEM Search Algorithm

Function Search(A, B, C)

Return : True If C is satisfiable by A ,
otherwise False

Begin

If $B = \emptyset$ **Then Return** TRUE

Choose and delete (ce_i, D_i) from B %%with respect to LNH or LNHO heuristic

If $D_i = \emptyset$ **Then Return** FALSE

for All $e \in D_i$ **Do**

$(A', B', C') = \text{Propa}(A \cup (ce_i, e), B, C)$

If $C' \neq \text{False}$ **Then return** Search(A', B', C')

End

SEM’s finite model search procedure is described in the algorithm 2. It uses the following parameters :

- A : a set of assignments $\{(ce, e) | e \in D_{\text{sort}(ce)}\}$
- B : a set of unassigned cells and their possible values $\{(ce, D) | D \subseteq D_{\text{sort}(ce)}\}$
- C : a set of clauses

Initially, A is empty and B contains all the cells $\{(ce, D_{\text{sort}(ce)})\}$. The back-track procedure **Search** calls **Propa** and itself recursively. The procedure **Propa** propagates the assignments from A in 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.

Before talking about the experiments we introduce the function ordering strategy that we use in SEM to select the terms to be interpreted first.

6 The Function Ordering Strategy

Interpretation of function symbols impacts on propagations, symmetry detection and then on computation times. We choose the next term to be interpreted

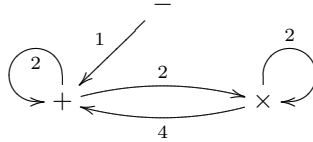
in such a way as to maximise the amount of propagations and preserve the symmetry. The function ordering we propose is based on the *dependency graph* of the functions that we define below. Intuitively, if the formula $f(g(x), x) = x$ is one of the axioms of a given theory T then f depends on g and the function g will be before the function f in the ordering.

Definition 8. Let $T = (S, F, C)$ be a theory. Let $G = (X, W)$ be the weighted and oriented dependency graph defined as follows:

- the set of vertices is the set F without constant symbols.
- the edge $(f_i \rightarrow f_j)$ belongs to W if there exists an axiom in C where f_i appears as a sub-term of f_j .
- the weight of the edge $(f_i \rightarrow f_j)$ equals the number of times there exists an axiom in C where f_i appears as a sub-term of f_j .

Example 3. Consider the unit ring theory of table 1.

The axioms of the theory yields the following dependency graph:



6.1 The Function Ordering

The function ordering that we perform in a preprocessing phase is defined with respect to the following strategy.

1. select first an n-ary function with no incoming arrow which has a minimal arity
2. secondly, select a function with the maximal sum of weights of arrows incoming from already selected functions
3. repeat recursively both step 1 and 2 on all the function symbols

For the example 3, the previous strategy, gives the following ordering: $-$, $+$ and \times . The choice of $-$ first is obvious, since it is a unary function without coming arrows. Secondly $+$ is chosen, because it depends on the already selected function $-$. Naturally the function \times is the last in the ordering.

6.2 The Term Selection Heuristic (LNHO)

According to both the function ordering strategy and trivial symmetry preservation, we define the heuristic LNHO (LNH with function ordering) which chooses as a next cell to be interpreted the term $f(e_1, e_2, \dots, e_k)$ satisfying the following conditions:

1. The maximum of the individuals $\{e_1, e_2, \dots, e_k\}$ is the smallest.

2. The function f is the smallest in the function ordering
3. If more than one cell satisfies both condition 1 and 2, then choose the one with the smallest domain.

Constants (functions of arity 0) have to be considered. In LNHO, they are considered as 0-arity functions. They deserve a distinct treatment depending whether they appear in equations or disequations. On the former case constants are considered in prior (before other functions) to favour propagation. But on the second case, they are selected at last, since their interpretation breaks several symmetries.

7 Experimentations

We have experimented and compared both LNH and LNHO heuristics on the SEM finite model generator. We also compared the performances of SEM with and without the advantage of the detected symmetries (DSYM). First, we study the behaviour of the symmetry detection on different problems, which we describe in section 7.1. We also show the interest to combine this dynamic symmetry elimination with the LNHO heuristic which itself improves LNH heuristic. All CPU times are given in seconds and are obtained on a PC (K6II, 400Mhz, 128Mb) running under Linux 2.2.

The program source is available at <http://www.cmi.univ-mrs.fr/~audemard>.

7.1 Problems

In a first time, our experiments are focused on mathematical problems. Some of them are described by Jian Zhang in [9, 10], the other ones are picked up from the TPTP library [8]. The symbol AG denotes Abelian groups, NG denotes non Abelian groups, GRP100-1 (from the TPTP library) express one of the proposed conjectures to represent an Abelian group with one axiom. RU is for unit rings, RNB denotes a non Boolean ring ($a \times a \neq a$ for some constant a) which satisfies the additional axiom ($x^7 = x$), RNA contains the ring axioms where the associativity of the operator \times is replaced by a counter example. RNG025-8 is picked up from the TPTP library. The axioms of both problems AG and RU are given in table 1 and most of the other problems are generated from the two previous ones by introducing or removing some axioms. In a second time, we deal with the linear-time temporal logic problems (LTL) expressed in first-order logic, see [11] for more details.

7.2 The Behaviour of the Symmetry Detection

Table 2 shows the behaviour of symmetry detection on three different problems when using SEM (with LNH) as a finite model generator. We can see for the Abelian groups that 82% of symmetry calls (i.e., the ratio between the result of the column “Find” to the result of the column “Calls” in table 2) succeed. This

Table 1. Problems AG and RU

$x + 0 = x$	$x + 0 = x$
$0 + x = x$	$x + y = y + x$
$x + (-x) = 0$	$x + (-x) = 0$
$(-x) + x = 0$	$x + (y + z) = (x + y) + z$
$x + (y + z) = (x + y) + z$	$x \times 1 = x$
$x + y = y + x$	$x \times y = y \times x$
	$(x \times y) + (x \times z) = x \times (y + z)$
	$(y \times x) + (z \times x) = (y + z) \times x$
	$x \times (y \times z) = (x \times y) \times z$
Abelian Group	Unit Ring

Table 2. Behaviour of the algorithm of symmetries detection

Pb	size	Time	Time for Sym	Nodes	Calls	Find	Suppressed
AG	24	117	55	124 753	12033	9803	118698
	25	132	59	130 644	13009	10737	133088
NAG	20	171	36	338 669	29388	25921	321123
	21	206	37	357 095	32480	27177	344578
RNA	8	5	0.4	51 491	7872	0	0
	9	10	0.8	81928	11 243	0	0
	10	62	5.3	252720	41 163	18 902	93184

eliminates a great number of isomorphic interpretations by symmetry cuts (see Column “Suppressed”). We can remark that symmetry detection is sometimes time consuming for the AG problems. This is because the necessary conditions of Proposition 1 does not reduce significantly the permutation search space.

Replacing the last axioms of the Abelian group (AG) (see table 1) by the clause $(1+2 \neq 2+1)$ yields the non Abelian group problem (NAG) and forces the individuals 1 and 2 to be permuted together (see remark 1). This strengthens the symmetry necessary conditions and the symmetry search space is substantially reduced. Symmetry detection is then more efficient for these problems.

For the non associative ring problems (RNA), the added clause $((2 \times 3) \times 4 \neq 2 \times (3 \times 4))$ forces the individuals $0 \dots 4$ to be in the same class of symmetry. As in NAG, this clause reduce the symmetry search space and the detection time. We can see in table 2 that symmetries abound as the problem size grows.

Table 3 shows the behaviour of symmetry detection (DSYM) when using both LNH and LNHO heuristics. We remark that it is more profitable to use LNHO. When using LNHO heuristic, the constants of the axiom expressing non associativity in RNA problem are interpreted at the last. This helps to preserve symmetries.

Table 3. The impact of using LNHO heuristic on the symmetry detection

Problem	DSYM with LNH			DSYM with LNHO		
	Calls	Find	Suppressed	Calls	Find	Suppressed
RNA (8)	14 333	6 571	0	943	810	316
GRP100-1	129 521	0	0	29	9	15

7.3 The advantage of the detected symmetries (DSYM) when using both LNH and LNHO heuristics in SEM

In this section, we compare both LNH and LNHO heuristics and their combination with the detected symmetries (DSYM) on the problems described previously.

Table 4. Comparison - Group problems

Problem	size	LNH		LNHO		LNH + DSYM		LNHO + DSYM	
		Model	Time	Model	Time	Model	Time	Model	Time
AG	6	6	0.01	6	0.01	5	0.01	6	0.01
	32	2 295	962	2 295	1 086	551	1 298	1 037	1 429
	35	13	1 734	13	2 060	-	-	5	1 429
	37	-	-	1	3 425	-	-	1	3 018
	38	-	-	-	-	-	-	14	3 330
NAG	12	31	1.3	59	1	20	1.3	42	1.3
	24	1 130	2 671	3 786	587	493	1 777	1 654	473
	27	-	-	45	1 295	9	2 455	16	856
	30	-	-	301	3 191	-	-	135	1 853
GRP100-1	4	0	10.3	0	0.05	0	11.5	0	0.04
	5	0	2 008	0	0.05	-	-	0	0.56
	9	-	-	0	45	-	-	0	36
	10	-	-	-	-	-	-	0	2 284

Tables 4 and 5 show the results obtained when searching all the models (column model). The run time is limited to one hour. The mark (-) means that the program fails to answer the question in less than one hour time. We can see that SEM with the detected symmetry (DSYM) outperforms SEM without symmetry detection on the checked problems AG, NAG, GRP100-1, RU, RNA and RNB. With the advantage of the detected symmetry, we generate models of a great size that SEM cannot find in a reasonable time and several new isomorphic models that both LNH and LNHO heuristics do not consider, are detected. We remark that the function ordering improves the LNH heuristic (LNHO is generally better than LNH) on all the checked problems except for the RNB and RNG025-8 problems. Thus, the combination of LNHO with DSYM is better than the one of LNH with DSYM.

Table 5. Comparison - Ring problems

		LNH		LNHO		LNH + DSYM		LNHO + DSYM	
Problem	size	Model	Time	Model	Time	Model	Time	Model	Time
RU	6	1	0.05	1	0.03	1	0.05	1	0.02
	16	1 745	165	1 745	70	355	49	1 119	55
	19	1	1323	1	113	1	225	1	91
	27	-	-	298	2 606	-	-	196	1 809
	28	-	-	29	3 338	-	-	29	2 314
RNG025-8	5	320	6	20	930	320	7	20	1 026
	6	960	48	144	3 332	960	49	-	-
RNA	6	0	0.86	0	0.02	0	0.9	0	0.02
	8	2 496	10	280	0.74	2 496	11	224	0.74
	10	0	90	0	1	0	105	0	0.9
	15	0	1 047	0	20	0	1 061	0	18
RNB	16	90	8.3	800	132	30	3.5	331	60
	18	0	10	0	170	0	4	0	80
	30	0	79	-	-	0	30	-	-
	32	3 550	2 517	-	-	464	417	-	-
	34	-	-	-	-	0	465	-	-

Table 6 shows the results obtained on the LTL problems. For an instance of size n we know that there exist exactly n non-isomorphic models. We can see that the LNHO heuristic eliminates more isomorphic models than the LNH one and the advantage of the detected symmetry (DSYM) on these problems is substantial. The combination of DSYM with both LNHO and LNH increases individually their performances and DSYM with LNHO seems to be the best.

Table 6. Comparison - LTL problems

		LNH		LNHO		LNH + DSYM		LNHO + DSYM	
size	Model	Time	Model	Time	Model	Time	Model	Time	
4	23	0.01	19	0.01	12	0.01	11	0.01	
5	71	0.01	47	0.01	24	0.01	19	0.01	
6	243	0.11	117	0.05	55	0.11	42	0.03	
7	988	1.3	289	0.34	117	1	92	0.15	
8	4 950	23	724	5	299	11	205	2.5	
9	30 556	629	1 836	98	719	150	479	61	
10	-	-	-	-	-	-	1 161	1955	

8 Conclusion

We have shown some new results on symmetry for finite model generation in many-sorted first-order logic theories. The trivial symmetry is shown to be a

particular case of our symmetry notion and a symmetry detection algorithm is given. A function ordering resulting in a new heuristic LNHO which improves the LNH heuristic is studied. The trivial symmetry treated by both LNH and LNHO heuristics is combined with the detected symmetry (DSYM) to maximise the gain. The method SEM augmented with the property of the detected symmetry is experimented on several problems. The experimental results obtained with our first implementation are satisfactory and confirmed the advantage of using the detected symmetry. Many improvements of the implementation are possible. One can refine the necessary symmetry conditions to increase the efficiency of symmetry detection. Another point of interest is to exploit some other trivial symmetries (like the ones of XLNH) which do not need detection and combine them with the detected ones. Our experiments are still in progress. We also aim to apply our results for finite model generation of modal logics and planing problems.

Acknowledgements: Many thanks to the referees for their comments which helped us to provide the final version of this paper.

References

1. G. Audemard and L. Henocque. The extended least number heuristic. *Proceedings of the first International Joint Conference in Automated Reasoning (IJCAR)*, Springer Verlag, 2001.
2. B. Benhamou and L. Henocque. A hybrid method for finite model search in equational theories. *Fundamenta Informaticae*, 39(1-2):21–38, 1999.
3. B. Benhamou and L. Sais. Tractability through symmetries in propositional calculus. *Journal of Automated Reasoning*, 12(1):89–102, 1994.
4. J. Crawford. A theoretical analysis of reasoning by symmetry in first-order logic. In *Proceedings of Workshop on Tractable Reasoning, AAI92*, pages 17–22, 1992.
5. J. Crawford, M. L. Ginsberg, E. Luck, and A. Roy. Symmetry-breaking predicates for search problems. In *proceedings of KR'96*, pages 148–159. 1996.
6. M. Fujita, J. Slaney, and F. Bennett. Automatic generation of some results in finite algebra. In *Proceedings of International Joint Conference on Artificial Intelligence*, pages 52–57. Morgan Kaufmann, 1993.
7. N. Peltier. A new method for automated finite model building exploiting failures and symmetries. *Journal of Logic and Computation*, 8(4):511–543, 1998.
8. C. Suttner and G. Sutcliffe. The TPTP Problem Library. Technical Report, J. Cook University, 1997. <http://www.cs.jcu.edu.au/ftp/pub/techreports/97-8.ps.gz>
9. J. Zhang. Problems on the Generation of Finite Models. In *proceedings of the 12th International Conference on Automated Deduction*, LNAI 814, pages 753–757, Nancy, France, 1994. Springer-Verlag.
10. J. Zhang. Constructing finite algebras with FALCON. *Journal of Automated Reasoning*, 17(1):1–22, August 1996.
11. J. Zhang. Test Problems and Perl Script for Finite Model Searching. Association of Automated Reasoning , Newsletter 47, 2000. <http://www-unix.mcs.anl.gov/AAR/issueapril00/issueapril00.html>
12. J. Zhang and Hantao Zhang. SEM: a system for enumerating models. In *Proceedings of the Fourteenth International Joint Conference on Artificial Intelligence*, pages 298–303, 1995.