Journal of Automated Reasoning **12:** 89–102, 1994.

Tractability Through Symmetries in Propositional Calculus

BELAID BENHAMOU and LAKHDAR SAIS
L.I.U.P. Case G, Université de Provence, 3, Place Victor Hugo, F13331 Marseille cedex 3, France.
Tel. 91.10.61.08. e-mail: Benhamou@gyptis.univ-mrs.fr e-mail: Sais@gyptis.univ-mrs.fr

(Received: 12 March 1992; accepted: 15 June 1993)

Abstract. Many propositional calculus problems – for example the Ramsey or the pigeon-hole problems – can quite naturally be represented by a small set of first-order logical clauses which becomes a very large set of propositional clauses when we substitute the variables by the constants of the domain D. In many cases the set of clauses contains several symmetries, i.e. the set of clauses remains invariant under certain permutations of variable names. We show how we can shorten the proof of such problems. We first present an algorithm which detects the symmetries and then we explain how the symmetries are introduced and used in the following methods: SLRI, Davis and Putnam and semantic evaluation. Symmetries have been used to obtain results on many known problems, such as the pigeonhole, Schur's lemma, Ramsey's, the eight queen, etc. The most interesting one is that we have been able to prove for the first time the unsatisfiability of Ramsey's problem (17 vertices and three colors) which has been the subject of much research.

Key words. Theorem proving, propositional calculus, symmetries.

1. Introduction

The subject of this paper is the study of symmetries in propositional calculus in order to make the automated deduction algorithms more efficient. Using resolution as a base proof system for the propositional calculus, Tseitin [12] has shown how certain tricky mathematical arguments can be considered as short proofs for some propositional tautologies representing mathematical statements. He suggested increasing resolution by the principle of extension, which consists of the introduction of auxiliary variables to represent intermediate formulas so that the length of proof can be significantly reduced by manipulating these variables, rather than the formulas that they represent.

On the other hand, Krishnamurty [7] proposed the principle of symmetries, which allows us to recognize that a tautology remains invariant under certain permutation of variables names, and uses this information to avoid repeated independent derivations of intermediate formulas that are permutations of others. Consider the logical formulas associated with a concrete problem. In most cases, we use first-order logic to express this formula, using variables, constants, predicates. In other words, we use the usual mathematical language. From this set of first-order logic clauses we obtain the propositional clauses by substituting the variables by

constants of the domain D. Thus the previous set of clauses becomes a large set for many problems and contains several symmetries. The property of symmetry can lead to a shorter proof for the problem. However, the current methods of theorem proving do not take advantage of symmetry properties, and in ref. [7] no algorithm is presented neither for searching for symmetries nor for using them in a formal way.

The purpose of this study is the detection and the use of symmetries. First we describe the method which computes the symmetry on a given set of clauses. The algorithm is given in greater detail in refs [2] and [3]. Second, we explain how symmetries are introduced and combined with different automated deduction algorithms like Sl-resolution [8], the Davis and Putnam procedure [5], and semantic evaluation [10]. Finally we apply SLRI [4] and the semantic evaluation methods with and without symmetry to some classical problems, such as the pigeonhole [13] and [15], Schur's lemma and Ramsey's problems [10] and the comparison shows that it is possible to obtain shorter proofs for hard propositional totaulogies and to significantly reduce the complexity of resolution in many cases. One of the most interesting results is that we have been able to prove the unsatisfiability of Ramsey's problem with 17 vertices and three colors, which had not been shown by a theorem proving program before.

2. Definitions and Notations

We shall assume that the reader is familiar with the propositional calculus. For a formal description see e.g. ref. [9]. Let V be the set of propositional variables, which are simply called variables. Variables will be distinguished from literals, which are variables with an assigned parity 1 or 0 (which means TRUE or FALSE, respectively). This distinction will be ignored whenever it is convenient but not confusing. For a propositional variable, p, there are two literals: p, the positive literal and $\neg p$, the negative one.

A clause is a disjunction of literals $(p_1, p_2, \ldots, p_n)$ such that no literal appears more than once and is denoted by $p_1 \vee p_2 \vee \ldots \vee p_n$. A system, S, of clauses is a conjunction of clauses. In other words we say that S is in the conjunctive normal form (CNF).

A truth assignment to a system of clauses S is a map I from the set of variable in S to the set {TRUE, FALSE}. Sometimes I can be considered as a set of literals which are true. If $I[p]$ is the value for the positive literal p then $I[\neg p] = 1 - I[p]$. The value of a clause $p_1 \vee p_2 \vee \ldots \vee p_n$ in I is the maximum value of its literals in I. By convention we define the value of the empty clause ($n = 0$) to be FALSE. The value $I[S]$ of the system of clauses is TRUE if the value of each clause of S is TRUE; FALSE otherwise. We say that a system of clauses, S, is satisfiable if there exists some truth assignment in which S takes the value TRUE; it is unsatisfiable otherwise. In the first case I is called a model of S. Note that a system which contain the empty clause is unsatisfiable.

It is well-known that for every propositional formula F there exists a formula F' in conjunctive normal form which is longer than F (see ref. [1]) and which is satisfiable

iff F is satisfiable. More precisely, for every polynomial $p(n)$, there is a formula F such that any equivalent formula in CNF is of length at least $p(|F|)$, where $|F|$ is the length of F. If F is a clause c then $|c|$ is the number of literals in the clause c. In the following we will assume that the formulas are given in a conjunctive normal form.

3. Symmetries

First of all, let us define the concepts of permutations and symmetry, and prove significant properties that will enable us to improve the proof algorithms. For more detail we refer the reader to ref. [2].

A bijection map $\sigma : V \to V$ is called a permutation of variables. If S is a set of clauses, c a clause of S and σ a permutation of variables occuring in S, then $\sigma(c)$ is the clause obtained by applying σ to each variable of c and $\sigma(S) = \{\sigma(c)/c \in S\}$.

DEFINITION 3.1. A set, P, of literals is called *complete* if $\forall l \in P$; then $\neg l \in P$.

DEFINITION 3.2. Let P be a complete set of literals and S a set of clauses of which all literals are in P. Then a permutation σ defined on P ($\sigma : P \to P$) is called a *symmetry of* S if it satisfies the following conditions:

(1) $\forall l \in P$, $\sigma(\neg l) = \neg\sigma(l)$
(2) $\sigma(S) = S$

DEFINITION 3.3. Two literals (*variables*) l and l' are symmetric in S notation ($l \sim l'$) if there exists a symmetry σ of S such that $\sigma(l) = l'$. A set $\{l, l_1, l_2, \ldots, l_n\}$ of literals is called a *cycle of symmetry in* S if there exists a symmetry σ defined on S, such that $\sigma(l) = l_1$, $\sigma(l_1) = l_2, \ldots, \sigma(l_{n-1}) = l_n$, $\sigma(l_n) = l$.

Remark 3.4. All the literals in a cycle of symmetry are symmetrical two by two.

Example 3.5. Let S be the following set of clauses:

$$S : \{c1 : a \vee \neg b, c2 : c\}$$

and σ the map defined on the set P of literals occurring in S:

$$\begin{aligned}
\sigma : \ & P \to P \\
& a \to \sigma(a) = \neg b \\
& \neg a \to \sigma(\neg a) = b \\
& b \to \sigma(b) = \neg a \\
& \neg b \to \sigma(\neg b) = a \\
& c \to \sigma(c) = c \\
& \neg c \to \sigma(\neg c) = \neg c
\end{aligned}$$

σ is a symmetry of S, a and $\neg b$ are symmetrical in S ($a \sim \neg b$). $\sigma(S) = \{c1 : \neg b \vee a, c2 : c\} = S$.

Remark 3.6. $\sigma(S)$ is the set of clauses obtained from S by exchanging the literal positions in the same clause or exchanging the order of the clauses in S. Generally, the two previous operations can be applied simultaneously.

- the identity map is a symmetry;
- the inverse map of a symmetry is also a symmetry;
- the composition of symmetries is a symmetry.

DEFINITION 3.7. Let P be a complete set of literals, σ a symmetry, I a truth assignment of P and S a set of clauses. Then:

- I/σ is the *truth assignment* obtained by substituting every literal l in I by $\sigma(l)$.
- S/σ is the *set of clauses* obtained by substituting every literal l in S by $\sigma(l)$.

4. Symmetry Properties

If I is a model S and σ a symmetry, then we can get another model of S by applying σ on the variables which appear in I. In the following propositions we assume that σ is a symmetry of the system of clauses S.

PROPOSITION 4.1. *I is a model of $S \Leftrightarrow I/\sigma$ is a model of S.*

Proof. Cf. ref. [2].

PROPOSITION 4.2. *Let l be a literal such that $l' = \sigma(l)$ and $I' = I/\sigma$. If I is such that $I[l] = 1$, then I' is such that $I'[l'] = 1$.*

Proof. Direct consequence of Proposition 4.1: if $I[l] = 1$ then I is a model of l or , I/σ is a model of $\sigma(l)$(Proposition 4.1). Then I' is a model of l', thus $I'[l'] = 1$.

PROPOSITION 4.3. *If l has the value true in a model of S, then l', such that $l' = \sigma(l)$, will have the value true in a model of S.*

Proof. Direct consequence of Proposition 4.2: if $l =$ true in a model I of S, then $I[l] = 1$, applying Proposition 4.2 we get $I'[l'] = 1$ such that $I' = I/\sigma$, then $l' =$ true in I' which is a model of S.

THEOREM 4.4. *Let l and l' be two literals of S. If $l \sim l'$ in S, then [l has a model in $S \Leftrightarrow l'$ has a model in S] (l has a model in S if only if there exists a model I of S such that $I[l] = 1$).*

Proof. ($\Rightarrow$) Suppose that l has a model in $S \Leftrightarrow \exists I$ a model of S such that $I[l] = 1$. $l \sim l'$ in $S \Rightarrow \exists \sigma$ a symmetry of S such that $\sigma(l) = l'$. I is a model of $S \Rightarrow I/\sigma$ is a model of $S/\sigma = S$ (Proposition 4.1). We also have $\{\sigma(l) \in S/\sigma$ and $I[l] = 1\} \Rightarrow \sigma(l) \in I/\sigma$ (Proposition 4.2). Then $l' \in I/\sigma \Rightarrow I/\sigma$ is a model of l'.

($\Leftarrow$) Let $l' = \sigma^{-1}(l)$. The proof is then identical.

It is very important to draw a distinction between symmetric literals and equivalent literals. Equivalent literals have exactly the same models. But if l and l' are symmetrical then: [l has a model in $S \Leftrightarrow l'$ has a model in S], and the two models

are generally different, so it is easier to find symmetric literals than equivalent ones. Note that Theorem 4.4 expresses an important property that we shall use in many cases to make cuts in the resolution trees. Indeed, if l has no model in S and $l \sim l'$, then l' will have no model in S, thus we cut the branch which corresponds to the assignment of l' in the resolution tree. Therefore, if there are n symmetrical literals we can cut $n-1$ branches in the resolution tree.

PROPOSITION 4.5 (Necessary conditions for symmetry). *If two literals (variables) l and l' are symmetric in a set of clauses S then the number of occurrences of the variable l in S is the same as the number of occurrences of l' and there must be a correspondence between the length of clauses in which l occurs and the length of clauses in which l' occurs. I.e.*

$$l \sim l' \Rightarrow \textit{occurrence\_number}(l) = \textit{occurrence\_number}(l')$$

and

$$\textit{occurrence\_number}(\neg l) = \textit{occurrence\_number}(\neg l').$$

if $\exists c \in S$ *such that* $|c| = n$ *and* $l \in c \Leftrightarrow \exists c' \in S$ *such that* $|c'| = n$ *and* $l' \in c'$.

Proof. (1) The first condition is not satisfied: Let l and l' be two symmetric literals in S. Suppose that occurrence_number$(l) = n$ and occurrence_number$(l') = m$ with $(n \neq m)$. Then there are m clauses which contain l' in the set S and n clauses which contain l. After all permutations of l and l' the new set S' will have m clauses with the literal l and n clauses with the literal l', but $n \neq m$. Then $S \neq S' \Rightarrow$ there does not exist a symmetry σ of S such that $\sigma(l) = l'$ or $\sigma(l) = l$. Then l and l' are not symmetric. This gives a contradiction with the previous hypothesis. ■

The same proof can be performed if we suppose that occurrence_number$(\neg l) \neq$ occurrence_number$(\neg l')$.

(2) If the second condition is not satisfied: then $\exists c \in S$ such that $|c| = n$ and $l \in c$ and $\nexists c' \in S$ such that $|c'| = n$ and $l' \in c'$. After all permutation of l and l' the new system S' will have a clause of length n in which l' occurs. Hence, $S \neq S' \Rightarrow$ there does not exist a symmetry σ of S, such that $\sigma(l) = l'$ or $\sigma(l) = l$, then l and l' are not symmetric. This gives a contradiction with the previous hypothesis. ■

5. Symmetry Detection Method

Let S be a set of clauses. Finding a symmetry on S is equivalent to finding a permutation σ which keeps S invariant. Many symmetries may exist, but at each resolution step we are interested in only one of them. For example, in the SLRI method [4] for a refuting literal l, it is more interesting to get the symmetry which has more literals occurring in the same clause as l. But in the Davis and Putnam or in semantic evaluation procedures it is better to compute the wider cycle of a

given literal l – i.e. we try to find a set of literals $\{l, l_1, l_2, \ldots, l_n\}$ so that $\sigma(l) = l_1$, $\sigma(l_1) = l_2, \ldots, \sigma(l_{n-1}) = l_n$, $\sigma(l_n) = l$ with σ a symmetry defined on S.

It is known that each permutation can be expressed as a product of elementary permutations called transpositions. Thus we write all permutations, β, as a product in the form $\beta = (x_1, y_1)(x_2, y_2) \cdots (x_n, y_n)$. In order to check whether two literals x_0 and y_0 are symmetric in S we shall compute a symmetry, σ, of S, such that $\sigma(x_0) = y_0$. To that end, we try to express σ as a product of the form $\sigma = (x_0, y_0)\sigma'$, with σ' a sub-permutation of σ that we shall define as we do the other transpositions. Therefore, to substitute x_0 into y_0, we replace it by y_0 in each clause in which it appears. Thus we insert relationship clauses in which x_0 appears into the clauses in which y_0 does.

To do this, clauses and other variables will be related (taking into account the previously necessary conditions of symmetry: Proposition 4.5). For instance, let us consider the two following clauses: $c = x_0 \vee x_1 \vee x_2 \cdots \vee x_n$, $c' = y_0 \vee y_1 \vee y_2 \cdots \vee y_n$ and try to relate them. After replacing x_0 by y_0 we process the other variables in c. Thus each variable, x_i, $i \in \{1 \cdots n\}$, must be linked to a variable y_j, $j \in \{1 \cdots n\}$ and at each step we add the transposition (x_i, y_j) to σ.

If it is possible to substitute all the variables, we say that c is transformed into c' with σ i.e. $\sigma(c) = c'$. If all clauses containing x_0 have been related, then we consider the next transposition and we apply the same process. When there is no more transposition, the σ found is a symmetry of S.

The other possibility is that when we try the transposition (x_i, y_j) we cannot put into the relationship a clause in which x_i occurs. Thus, if $(x_i, y_j) = (x_0, y_0)$ then the symmetry σ does not exist. Otherwise we backtrack and try to link x_i to another variable (either in the same clause in which y_j occurs, or else we try to relate c to another clause).

Note that the symmetry between x_{i-1} and y_{j-1} depends on the one between x_i and x_j: i.e. if $\sigma = (x_0, y_0) \ldots (x_{i-1}, y_{j-1})(x_i, x_j) \ldots (x_k, y_k)$, then $(x_0 \sim y_0) \Rightarrow (x_{i-1} \sim y_{i-1}) \Rightarrow (x_i \sim y_i) \Rightarrow (x_k \sim y_k)$. In actual implementation we fix a level of backtracking in order not to spend too much time on failure cases. Thus we take another variable to substitute x_i iff the current backtracking number is not greater than the fixed level. Otherwise we reject the symmetry, since we are interested only in the ones which do not take more time than resolution. In practice our algorithm computes all symmetry in less than twenty backtrackings; when we go over this number, generally the symmetry does not exist. In many cases we find the symmetry with no backtracking. Our method therefore has a linear complexity aspect.

The problem is now to find a clever strategy (or a choice function) for substituting the variables in order to build σ in a satisfactory time.

5.1. STRATEGY

It is clear that the efficiency of the algorithm depends on the ordering of the variables to be permuted. We propose the following strategy:

(1) first consider the variables which can only substitute themselves;
(2) begin the processing with the variable of which we compute the cycle of symmetry;
(3) Each clause we begin to relate must be achieved as soon as possible.

In other words, variables in this clause are in priority in order to backtrack quickly in case of wrong choice.

(4) Identity is applied for the main variables, when there are no new transpositions to try.

For more details about this method and its implementation we refer the reader to ref. [2].

6. Advantage of Symmetries

The SLRI method is the algorithm of SL resolution [8], augmented by the variant of Cubbada [4]. This method is generally much more efficient than the others.

6.1. INTRODUCING SYMMETRIES IN THE SLRI METHOD

Using the result of Theorem 4.4, we can improve the method of semantic evaluation as well as the SLRI method. Instead of refuting each literal separately, it is possible to refute symmetric literals simultaneously. Thus, let $C = \{l_1, l_2, \ldots, l_{i-1}, l_i, \ldots, l_n\}$ be a clause in a system of clauses S. Suppose that the literals $l_1, l_2, \ldots, l_{i-1}$ have been refuted and that l_i is the literal we try to refute. If we prove that l_i is symmetric to one of the previous literals, then it will be directly refuted by the symmetry property.

Indeed, if $l_i \sim l_j$ in S with $1 \leqslant j \leqslant i-1 \Rightarrow [l_i$ has a model in $S \Leftrightarrow l_j$ has a model in $S]$, as l_j has no model in $S \Rightarrow l_i$ has no model in S, then l_i is refuted and we can cut the branch which corresponds to l_i assignment in the resolution tree. For optimizing the use of symmetries, we try to get more symmetric literals in the same clause, in order to cut a large part of the resolution tree.

Example 6.1. Let us consider the pigeonhole problem. (Put n pigeons in $n-1$ holes such that one pigeon at most can be in a hole at each moment.) For instance, let $n = 3$ (3 pigeons and 2 holes), this problem is described by the following set of clauses:

C_1 : $p_1(1) \vee p_1(2)$

C_2 : $p_2(1) \vee p_2(2)$

C_3 : $p_3(1) \vee p_3(2)$

C_4 : $\neg p_1(1) \vee \neg p_2(1)$

C_5 : $\neg p_1(1) \vee \neg p_3(1)$

C_8 : $\neg p_2(1) \vee \neg p_3(1)$

C_6 : $\neg p_1(2) \vee \neg p_2(2)$

C_7 : $\neg p_1(2) \vee \neg p_3(2)$

C_9 : $\neg p_2(2) \vee \neg p_3(2)$

The resolution tree of the SLRI method is shown in Figure 1.

The resolution tree of the SLRI method associated with symmetry is shown in Figure 2.

It is clear that the subtree built from $p_1(1)$ in the resolution tree of SLRI (Figure 1) is symmetric with the one built from $p_1(2)$. This is due to the fact that the two literals $p_1(1)$ and $p_1(2)$ are symmetric in S. Therefore, in the second resolution tree (Figure 2) the sub-tree of $p_1(2)$ disappears, because $p_1(2)$ is refuted directly according to its symmetry with $p_1(1)$. When the number of pigeons grows, symmetries are used at different levels of resolution. Thus we cut more than one branch in the resolution tree; the complexity of the problem is linear in number of steps. Let us remark that, when the number of pigeons is greater than 8, the resolution of this problem with the SLRI method without symmetry becomes impossible. The pigeonhole problem is studied in greater detail in the next section in order to show the influence of the symmetry property on the SLRI method.

6.2. SYMMETRIES IN THE SEMANTIC EVALUATION METHOD

DEFINITION 6.2.1. Let S be a system of clauses and $p_1, p_2, \ldots, p_n$ distinct literals which occur in S such that the set $\{p_1, p_2, \ldots, p_n\}$ does not contain both a literal and its opposite.

$Tp_1, p_2, \ldots, p_n(S)$: The set of clauses obtained from S by removing all the clauses

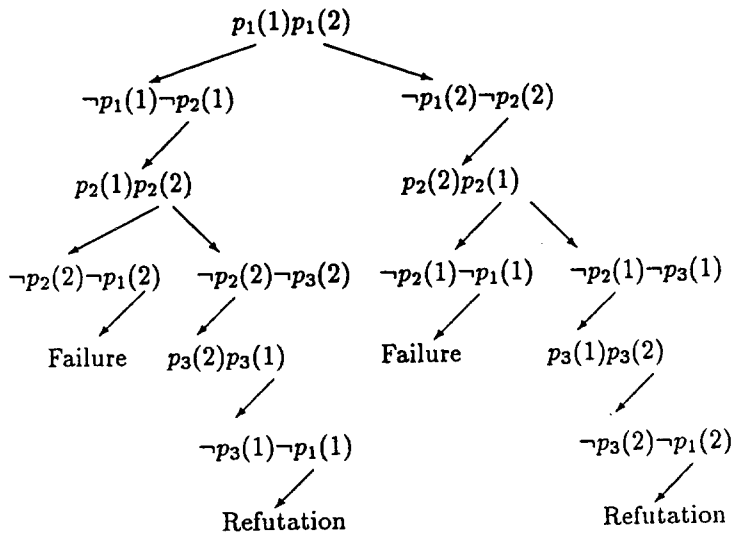

Fig. 1. Resolution tree of the SLRI method. (Pigeonhole problem, $n = 3$.)

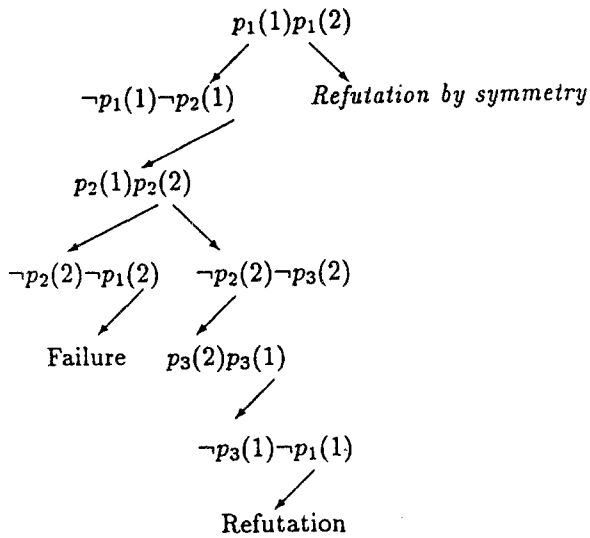

Fig. 2. The $n = 3$ pigeonhole problem resolution tree, SLRI method with symmetry.

containing one of the following literals $p_1, p_2, \ldots, p_n$ and all the occurrences of $\neg p_1, \neg p_2, \ldots, \neg p_n$ in the other clauses.

$T''p_1, p_2, \ldots, p_n(S)$ is the set $Tp_1, p_2, \ldots, p_n(S)$ simplified by subsumption: i.e. if $Tp_1, p_2, \ldots, p_n(S)$ contains both a clause c and a sub-clause of c then the clause c does not stand in $T''p_1, p_2, \ldots, p_n(S)$.

The QUINE algorithm has a semantic aspect, and recursively checks the satisfiability of a system of clauses.

Let S be a system of clauses.

Satisfiable (S):
if $S = \emptyset$
then S is satisfiable.
else if S contains an empty clause
  then S is insatisfiable
  else begin
    choose arbitrarily one litteral p which occurs in S
    if $T''p(S)$ *is satisfiable*
    then S is satisfiable
    else if $T''\neg p(S)$ *is satisfiable*
      then S is satisfiable
      else S is not satisfiable
    end.

Many improvements have been added to the basic algorithm (QUINE). The Davis and Putnam procedure is obtained by introducing the following two rules.

The first one consists to satisfy the unit clauses in priority. The second one consists in assigning the monotone literals in priority. Indeed, Davis and Putnam uses the following property. If p is a monotone literal in S, then S is satisfiable iff $T''p(S)$ is satisfiable.

Assigning a pure literal allows to make a cut in the proof tree. Generalizing this property leads to the theorem of model partitioning [10] which allows us to cut more branches in the proof tree when a monotone literal has been assigned a value.

The method of semantic evaluation has its origin in the previous property. It is more efficient than the two previous algorithms.

6.3. INTRODUCING SYMMETRIES

It is also possible to introduce symmetries into the semantic evaluation method. We will take account of the same property as above (if the literals l and l' are symmetric in S, then [l has a model in $S \Leftrightarrow l'$ has a model in S]).

At each step of resolution we assign to a literal the value TRUE or FALSE. If it generates the empty clause, then we insert in the model which is being built the opposite of that literal, together with all the other opposites of its symmetric literals.

Indeed: if l is a literal in a system S such that $T''l(S) \models \square$ ($\square$ is the notation of the empty clause and $\models$ the logical implication symbol), then l has no model in $S \Rightarrow \forall I$ a model of $S'I[\neg l] = 1$, if $l \sim l' \Rightarrow l'$ has no model in S (Theorem 4.4), thus $\forall I$ a model of $S'I[\neg l'] = 1$. Now we can cut in the resolution tree the branch which corresponds to the assignment of l' and insert $\neg l$ and $\neg l'$ in the model which is being built.

As a consequence, the system of clauses is significantly simplified, and many branches of the resolution tree are cut simultaneously, because in the backtracking case we do not take care of negation assignments of symmetric literals. Thus, for each symmetric literal, we cut a branch in the resolution tree. Then with n symmetric literals we can make $n - 1$ cuts.

To implement this new algorithm we need a procedure which, for a given literal, first computes a cycle of its symmetric literals which generates the empty clause. Such a cycle is obtained by putting literals in the same clause into correspondence. If this cycle exists, then we backtrack immediately, because we know that there exists no model at the current level; otherwise we compute the larger cycle (which contains more symmetric literals) in order to quickly obtain the empty clause or a model of the system of clauses.

7. Application

7.1. SLRI APPLICATIONS

We have seen that the symmetry property improves the efficiency of the resolution of classical problems, such as the pigeonhole problem, Shur's lemma, the eight queens, etc.

The two graphs below (Figure 3) illustrate the results of the pigeonhole problem solved by the SLRI method, with and without advantage being taken of symmetries.

7.1.1. *Interpretation*

Figure 3(a) shows that the complexity of resolution of the SLRI method associated with symmetry is linear in the number of elementary steps; whereas SLRI without symmetry cannot solve the problem when the number of pigeons is greater than eight. The complexity of resolution is exponential. Figure 3(b) shows that the CPU time is no longer exponential, but becomes proportional to n^2.

The statement of the Ramsey problem with 17 vertices and 3 colors is: 'Use three different colors to color a complete graph with 17 vertices, such that no monochrome triangles will appear'. In the SLRI method we use symmetry on literals which occur

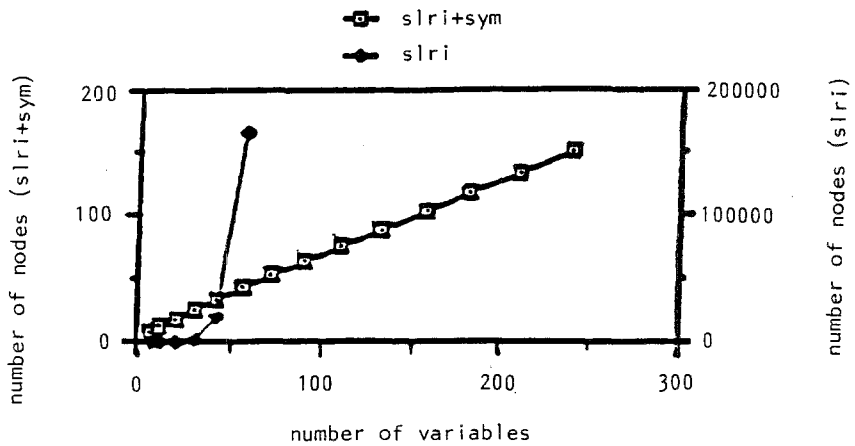

Fig. 3(a). The pigeonhole problem: comparison of the number of nodes.

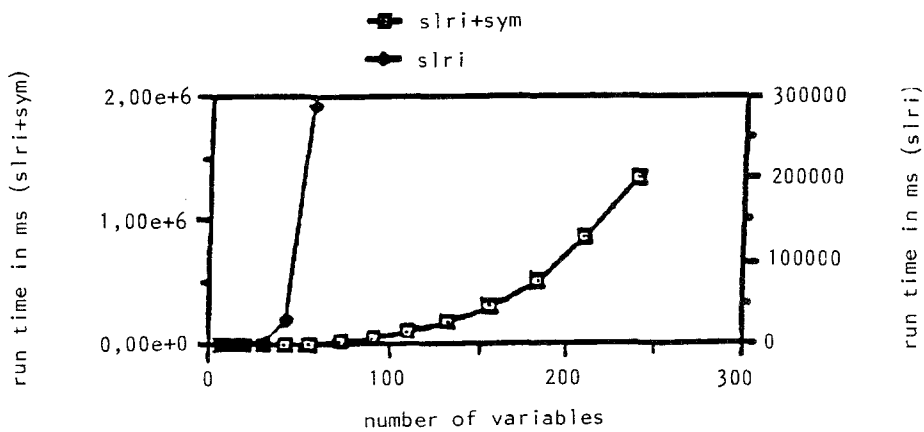

Fig. 3(b). The pigeonhole problem: comparison of run times.

in the same clause. Unfortunately such symmetries do not exist at different levels of resolution in the Ramsey problem. Moreover, there are symmetric literals in different clauses at all levels of resolution; therefore, the method of semantic evaluation can give better results. In that case, we need a good heuristic to decide which is the literal to assign next, in order to retain the symmetries in the next resolution levels.

7.2. SEMANTIC EVALUATION APPLICATIONS

We now present some results on the semantic evaluation method with and without taking advantage of the symmetry property. For comparison we present in program Eval_Sem some results obtained with the semantic evaluation method and in program Eval_Sem + Sym we give the corresponding results of the method, taking advantage of symmetries.

Table I shows some results on Schur's Lemma and on the Ramsey problems. Schur's Lemma consists of coloring the first n integers with three colors such that if i and j are colored with the same color then the integer k, such that $k = i + j$ and $k \leqslant n$, must be colored with a different one. For all integer i the color of every integer k, such that $k = 2 * i (k \leqslant n)$ must be different. For the Ramsey problem, the edges of a complete graph on n vertices are colored with three different colors such that no monochromatic triangle appears.

The most satisfying result is that we have proved for the first time the unsatisfiability of the Ramsey problem with 17 vertices and three colors. This result is given by our algorithm, implemented in Pascal, in 30 min CPU time on a SUN4/110. The semantic evaluation algorithm without the symmetry property, has run for 15 h of CPU time and more than 1 400 000 steps on an HP 9000/350 without success.

Table II contains some results on the pigeonhole problem described above. In this problem there are symmetries at each level. Thus SLRI and the semantic evaluation method with symmetry detection have a linear resolution complexity. Without the symmetry property both methods fail to find the proof when the number of pigeons is greater than 10.

Table I. Schur's lemma and Ramsey's problem.

Problems	Clauses	Variables	Eval_Sem		Eval_Sem + Sym	
			Steps	Times	Steps	Times
Schur13	178	39	684	3.3″	42	0.05″
Schur14	203	42	2061	11.2″	249	1.33″
Ramsey14	1456	273	273	5.7″	273	3.76″
Ramsey15	1785	315	4078	1′.30″	3649	1′.18″
Ramsey16	2160	360	4094	1′.29″	2957	1′.25″
Ramsey17	2584	408	–	–	27 000	30′

Table II. Pigeonhole problems.

Number of pigeons	Clauses	Variables	Eval_Sem + Sym	
			Steps	Times
14	1197	182	193	4.8″
16	1816	240	253	7.73″
18	2619	306	321	13.50″
20	3630	380	397	22.36″
22	4873	462	481	37.11″
24	6372	552	573	55.58″
26	8150	650	673	1′36″
28	10234	756	781	2′2″
30	12645	870	897	3′4″

8. Conclusion

Many problems cannot be solved with the classical resolution methods when they involve more than a certain number of variables. However, most of these problems include symmetries; the efficiency of the resolution methods can be significantly increased by taking advantage of this property.

To that end, we have proved several propositions, and we have implemented the corresponding results in two resolution methods. Thus we have obtained very satisfactory CPU times for the pigeonhole problem; we have also been able to solve the Ramsey problem with 17 vertices and 3 colors, which has always been impossible before.

We intend to use the symmetries in order to compute only the models which are not symmetric with a model which has already been found during resolution. Thus, a problem will be solved when all the basic models are found; all the other ones can be deduced from these basic models by applying the symmetries. In particular, this method can be applied when considering how to obtain the two solutions which are not isomorphic for the Ramsey problem with 16 vertices and 3 colors [14]. Promising results are expected in the near future.

We are also working on what we call 'strongly symmetrical cycles', i.e: for any order of the literals, a cycle is still a cycle. This will be useful when dealing with the statement. 'N literals are true among the literals of the clause c'.

Acknowledgement

This research was supported by the PRC-GDR Intelligence Artificielle, the BAHIA and the MRE-INTER-PRC 'Classes Polynomiales' projects. Also, our thanks to Pierre Siegel and Jean-Louis Lassez for their helpful comments.

References

1. Bauer, M., Brand, D., Fischer, M. J., Meyer, A. R., and Paterson, M. S., 'A note on disjunctive form tautologies', *SIGACT News* **5**, 17–20 (1973).

2. Benhamou, B. and Sais, L., 'Etude des symétries en calcul propositionnel', Rapport de recherche Maiup No. 91-01, Université de Provence (1991).
3. Benhamou, B. and Sais, L., 'Etude des symétries en calcul propositionnel', Mémoire de Dea GIA-Marseille-Luminy (France) (1990).
4. Cubbada, C. and Mouseigne, M. D., 'Variantes de l'algorithme de sl-resolution avec retenue d'information', Thèse de 3ème cycle, GIA, Marseille, Luminy (France) (1988).
5. Davis and Putnam, 'A computing procedure for quantification theory', *JACM* **7,** 201–215 (1960).
6. Krishnamurty, B. and Moll, R. N., 'Examples of hard tautologies in the propositional calculus', *Proc. Thirteenth ACM Symp. Th. of Computing*, pp. 28–37 (1981).
7. Krishnamurty, B., 'Short proofs for tricky formulas', *Acta Informatica* **22,** 253–275 (1985).
8. Kowalski, R.A. and Kuehner, D., 'Linear resolution with selection function', *Artificial Intelligence* **2,** 227–260 (1971).
9. Lyndon, R. C., *Notes on Logic*, Van Nostrand Mathematical Studies (1964).
10. Oxusoff, L. and Rauzy, A., 'L'evaluation semantique en calcul propositionnel', Thèse de 3ème cycle, GIA, Marseille, Luminy (France) (1989).
11. Siegel, P., 'Representation et utilisation de la connaissance en calcul propositionnel', Thèse d'etat, GIA, Marseille, Luminy (France) (1987).
12. Tseitin, G. S., 'On the complexity of derivation in propositional calculus', In: *Structures in Constructive Mathematics and Mathematical Logic*, H. A. O. Shsenko (Ed.), pp. 115–125 (1968).
13. Bibel, W., 'Short proofs of the pigeonhole formulas based on the connection method', *J. Automated Reasoning* **6,** 287–297 (1990).
14. Kalbfleisch, J. G. and Stanton, R. G., 'On the maximal triangle-free edge-chromatic graphs in three colors', *J. Combinatorial Theory* **5,** 9–20 (1969).
15. Cook, S. A., 'A short proof of the pigeonhole principle using extended resolution', *SIGACT News* **8,** 28–32 (1976).