Iterative n-Literal Rule for Solving Satisfiability Problems

Rafael Rivera-Lopez [1], Olga Leticia Prado-de-la-Paz [2], and Juan Frausto-Solis

[1] Deparment of Systems and Computation, Technological Institute of Veracruz
Miguel A. de Quevedo 2779, 91860, Veracruz, Veracruz, Mexico
rrivera@itver.edu.mx

[2] Temic de Mexico, S.A. de C.V.
Av. Ignacio Allende Lote 20, Parque Industrial Cuautla, Cd. Ayala, Morelos, Mexico
olga.prado@temic.com

[3] Department of Computer Science, ITESM, Campus Cuernavaca
Av. Paseo de la Reforma 182-A, 62289, Temixco, Morelos, Mexico
juan.frausto@itesm.mx

Abstract. In this paper a new rule applicable to the process of resolution of satisfiability (SAT) problems is presented. This approach, called the n-literals rule (nLR), is an extension of the one-literal rule (1LR) proposed by Davis, Logemann and Loveland that eliminates a set of literals in each step of one iterative resolution process of a SAT problem. This document describes the elements that define the nLR and presents some experimental results showing an improvement by using this new rule.

1 Introduction

In recent years, many approaches have emerged for solving the satisfiability (SAT) problem. These approaches include local search ([1], [2]), backtracking ([3], [5]), Binary Decision Diagrams [6], and so on. Gu describes in several papers [8] and [9]) many methods for solving instances of SAT problems and proposes various classifications for them. The great amount of SAT methods, are derived from the fact that the SAT problem is one of the most important problems in complexity theory, artificial intelligence and other disciplines. In particular, the method developed in 1960 for Davis and Putnam, is considered one of the major practical approaches for solving SAT problems, because it is a complete method. The original DP method is based on unit resolution [10] and their revised version (DPL by Davis, et al. in [11]) uses splitting for avoiding memory explosion. method uses four rules for testing the unsatisfiability of a set of clauses: tautology rule (TR), one-literal rule (1LR), pure-literal rule (PLR) and splitting rule (SR).

The most important rules in the DPL method are the 1LR and the SR; the permits the elimination of unit clauses in the original logical formula and SR divides the problem into two smaller sub-problems. In [12], Frausto-Solis and Sanchez-An propose two variants of the 1LR in aims to improve the performance of DPL method: the structured 1LR with support set and the structured n-literal rule and in 3] one generalization of 1LR for two-literals is proposed using a recursive method. This paper presents one rule based on the structured n-literal rule, that eliminates a set of literals in each step of an iterative process of resolution. This document describes the elements that define the nLR and presents some experimental results that indicate the existence of an improvement in the resolution of problems using this rule.

Background

This section describes the concepts used for the developed of this approach: The SAT problem, the DPL method and the 1-literal rule.

2.1 Satisfiability (SAT) Problem

SAT problem was the first known NP-complete problem [14] and it has a fundamental role in complexity theory (this is the engine that impulses the search of an efficient method for solving it). The general question for the SAT problem is to decide if one assignment for the literals of a logical formula that makes it true exists or not. A SAT formula is in its conjunctive normal form (CNF) as in (1), where one clause is a disjunction of literals, as in (2), and a literal is a Boolean variable negated or not).

$$F = Clause_1 \wedge Clause_2 \wedge \ldots \wedge Clause_m \qquad (1)$$

$$Clause_i = x_1 \vee x_2 \vee \ldots \vee x_n \qquad (2)$$

2.2 DPL Method

Herbrand method (HM) is based on a refutation procedure and is the basis for the modern automatic theorem proving. Several researchers, as Gilmore, tried to applied HM [16], but their methods were inefficient. Davis and Putnam ([10], [11]) introduce a more efficient method for deciding the satisfiability of a set of clauses as 1). DPL method consists of four rules showed in Table 1, where S, S' and S'' are set of clauses, L is a literal and $\sim L$ is its negation.

In the implementation of the DPL method, both the tautology and pure-literal rules are often deleted. The tautology rule can applied before of start the DPL and the pure-literal rule can be considered the splitting rule with the set S_2 without clauses B_i. Figure 1 shows the algorithm for the DPL method.

Table 1. The rules of the DPL method

Rule	Description
Tautology	All tautologies are deleted from S and the set S' is constructed the remaining clauses. If S' is unsatisfiable then the original set unsatisfiable too.
One-Literal	If a unit clause L exists in S, all clauses containing L are deleted the set S' is constructed with the remaining clauses. S is satisfiabl. S' is empty. Otherwise, the set S'' is obtaining from S' by dele- the literals $\sim L$ from all clauses in S' (S' is simplified in $\sim L$). If clause in S'' is the null clause (•), then S'' and S are unsatisfiable.
Pure-Literal	A literal L is pure in S if the literal $\sim L$ does not appear in any in S. If L is pure, all clauses containing L can be deleted. If maining set S' is unsatisfiable, then S is unsatisfiable too.
Splitting	If S can be written as $(A_1 \vee L) \wedge \ldots \wedge (A_m \vee L) \wedge (B_1 \vee \sim L) \wedge \ldots \wedge$ $\sim L) \wedge R$, then the sets S_1 and S_2 can be constructed as $S_1 = A_1$ $A_m \wedge R$ and $S_2 = B_1 \wedge \ldots \wedge B_n \wedge R)$, respectively. If both S_1 and unsatisfiables then the original set S is unsatisfiable too.

```
procedure DPL($S$);
{ S is a set of clauses in CNF }
begin
  if unit clause exists in $S$
  then begin { 1-literal rule }
    $L = L \in S$ and $L$ is a unit clause;
    $S' = S \setminus$clauses with $L$;
    if $S' = \varnothing$ then return SAT;
    else begin
      $S'' = S$ simplified in $\sim L$;
      if $\bullet \in S''$ then return UNSAT;
      else return DPL($S''$);
    end
  end
  else begin { Splitting rule }
    $L$ is a selected literal of $S$ using some heuristic;
    if DPL($S \cup L$) is SAT
    then return(SAT);
    else
      if DPL($S \cup \sim L$) is SAT
      then return(SAT);
      else return UNSAT;
  end
end.
```

Fig. 1. The DLP method

3 The n-Literals Rule (nLR) Method

The DPL method is based in the determination of one literal for testing the satisfiability of a logical formula because both the 1LR and the SR need one literal for evaluating the set of clauses. The nLR method uses n literals, instead of only one, in each step of the resolution procedure. This approach produces that the both 1-literal rule and the splitting rule can be enunciated in one rule called the n-literals rule (nLR) and the resolution method can be called nLR method. Table 2 shows the description of the n-literals rule.

Table 2. The n-literals rule

Rule	Description
n-literals	If n literals are selected of *S*, these n literals are assigned with some truth-value and the clauses containing some of these n literals are removed of *S* for constructing the new set *S'*. *S* is satisfiable if *S'* is empty. Otherwise, the set *S''* is obtaining from *S'* by deleting the complement of the n literals from all clauses in *S'*. If for any combination of truth-values for these n literals, some clause in *S''* is the null clause (•), then *S''* and *S* are unsatisfiable.

3.1 Description of the Method

The nLR method needs to know the number of literals to eliminate in test and, once these literals are selected of the original set, they are assigned by some truth-value (one interpretation is assigned to the set of n literals). The method eliminates all the clauses that contain some of these literals. If the new set of clauses is empty, then the problem is satisfiable and the method finalizes; otherwise, the remaining clauses are simplified removing the complement of these n literals for them. If the result is unsatisfiable, then one new interpretation is assigned to the set of literals and newly is applied the procedure. If all possible interpretations produce that the set is unsatisfiable, then the original formula is unsatisfiable.

If when removing the complement of the n literals of the original set does not take place a null clause, then it is necessary to select another set of n literals and the procedure is repeated. When the null clause is founded in the reduced set (the remaining clauses simplified removing the complement of the n literals) it is necessary test all the possible interpretations for the set of n literals. The maximum number of interpretations for n literals is 2^n. Figure 2 shows the search three for the nLR when n=3 and Figure 3 shows the algorithm for the recursive nLR method.

In this paper we use the iterative version of the nLR method due to the recursive version of any algorithm needs additional time and memory space that the iterative version [17]. The iterative version of the nLR method is shown in figure 4.

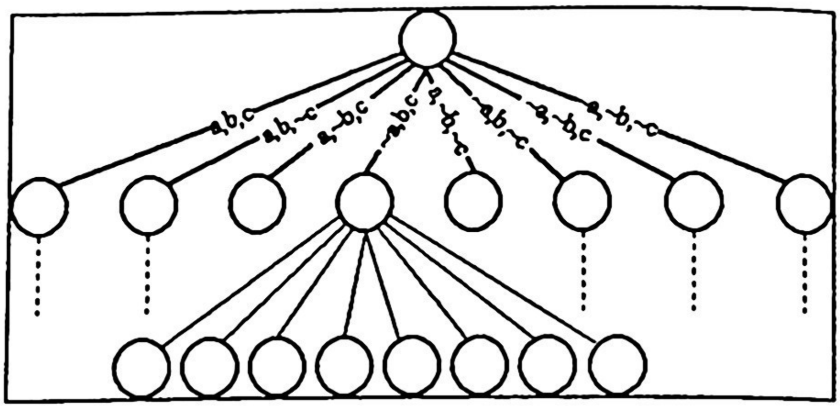

Fig. 2. Partial search three for the nLR method when n=3

```
procedure recursive_nLR($S$, $N$, $i$)
{ S is a set of clauses in CNF }
{ N is a set of n literals (L) selecting of S using some heuristic}
{ i is the number of interpretation, the maximum value of i is $2^n$}
begin
  $\forall L_k \in N$ ($L_k$ is assigned with truth-value, not assigned previously);
  $S' = S \setminus$ clauses in $N$;
  if $S' = \varnothing$ then return SAT;
  else begin
    $S'' = S$ simplified with the complement of clauses in $N$;
    if $\bullet \in S''$
    then
      if $i = 2^n$ then return UNSAT;
      else return recursive_nLR($S$, $N$, $i+1$);
    else begin
      { The set S is reduced and other n literals are selected}
      $S = S''$;
      $N = \{L_k \mid L_k \in S$ and $k : 1, \ldots, n\}$;
      $i = 0$;
      recursive_nLR($S$, $N$, $i+1$);
    end
  end
end.
```

Fig. 3. The recursive nLR method.

3.2 Selection of the Set of Literals

The selection of n literals is another important aspect in the nLR method. In simil form of the splitting rule on the DPL method, the n literals selected for testi satisfiability of a logical formula can be deterministic or random. These heuristi can be based over the number of occurrences of the literals in the original formula, selecting the most frequently or the less frequently, or if the literals are independents or not, or by column (selecting the literals in one clause), or in lexicographic al order and so on. Figure 5 shows several heuristics that can be applied in the nLR method.

```
procedure iterative_ nLR(S, $n$);
{S is a set of clauses in CNF }
{ n is the number of literal to eliminate }
{ N is a set of n literals (L) selecting from S using some heuristic }
begin
  $i = 1$; $j = 1$;
  $N_i = \{L_k \mid L_k \in S$ and $k : 1, \ldots, n\}$;
  repeat
    $\forall L_k \in N_i$ ($L_k$ is assigned with truth-value, not assigned previously);
    $S' = S \setminus$ clauses in $N_i$;
    if $S' = \varnothing$ then $S$ is SAT;
    else begin
      $S'' = S$ simplified in the complement of clauses in $N_i$;
      if $\bullet \in S''$
      then
        if $j = 2^n$
        then $S$ is UNSAT;
      else begin
        { other interpretation of literals in $N_i$ is necessary }
        $j = j + 1$;
      end
      else begin
        { The set S is reduce d and otherf n literals $N_i$ is selected }
        $S = S''$;
        $i = i + 1; j = 1$;
        $N_i = \{L_k \mid L_k \in S$ and $k : 1, \ldots, n\}$;
      end
    end
  until $S$ is SAT or UNSAT;
end.
```

Fig. 4. The iterative n-literal rule.

Experimentation and Results

For the test of this method several problems were resolved. Several hard instances of the problem were generated: instances of satisfiable formulas with 50 variables in formulas with 50 or 100 clauses and instances of unsatisfiable formulas for 10 variables in formulas with 403 clauses.

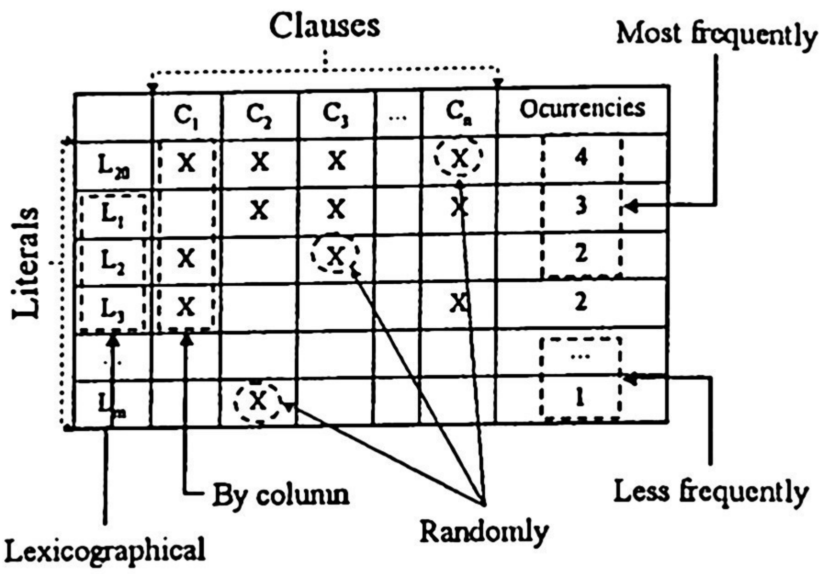

Fig. 5. Several forms for selecting the n literals in the nLR method

In Figure 6 is showed the behavior of nLR for n=1,2 and 3 in instances of satisfiable formulas. In this case the nLR is more efficient when n is increased because the number of iterations has a decremented behavior. The nLR have a improvement of 58% in the case of 50 clauses and 65% in the case of 100 clauses.

When the approach was probed with unsatisfiable hard instances (figure 7), the performance of nLR is degraded due to the combinatorial explosion in the number of interpretations that must be tested.

Another group of tests was made modifying the number of literals. Table 3 shows these test. In this case is observed that the number of iterations for the nLR is better than the 1LR.

5 Conclusions

The nLR method is a generalization of the rules proposed for Davis, Putnam and Loveland for solving satisfiability problems; when n = 1 the nLR is the original DPL method. In this paper is used the iterative approach for nLR due to its uses less memory and time for resolution of SAT problems.

For the results of the experimentation is showed that the effectiveness of the nLR has a better performance when n is increased. In the case of satisfiable formulas is clear the improvement in the number of iterations when n is increased. Is obvious that in the case for unsatisfiable formulas the nLR has not better performance due the increasing in the number of combinations in the interpretation of the set of literals.

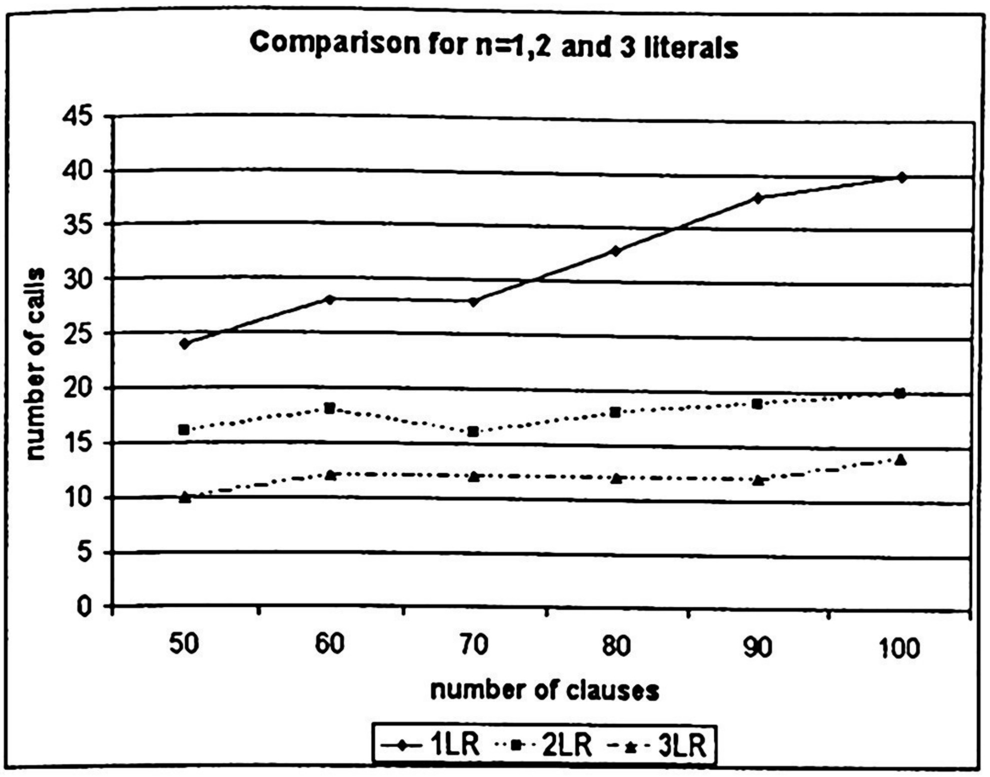

Fig. 6. Number of iterations for the nLR method when n=1, 2 and 3 and hard instances of 50-100 clauses with 50 literals.

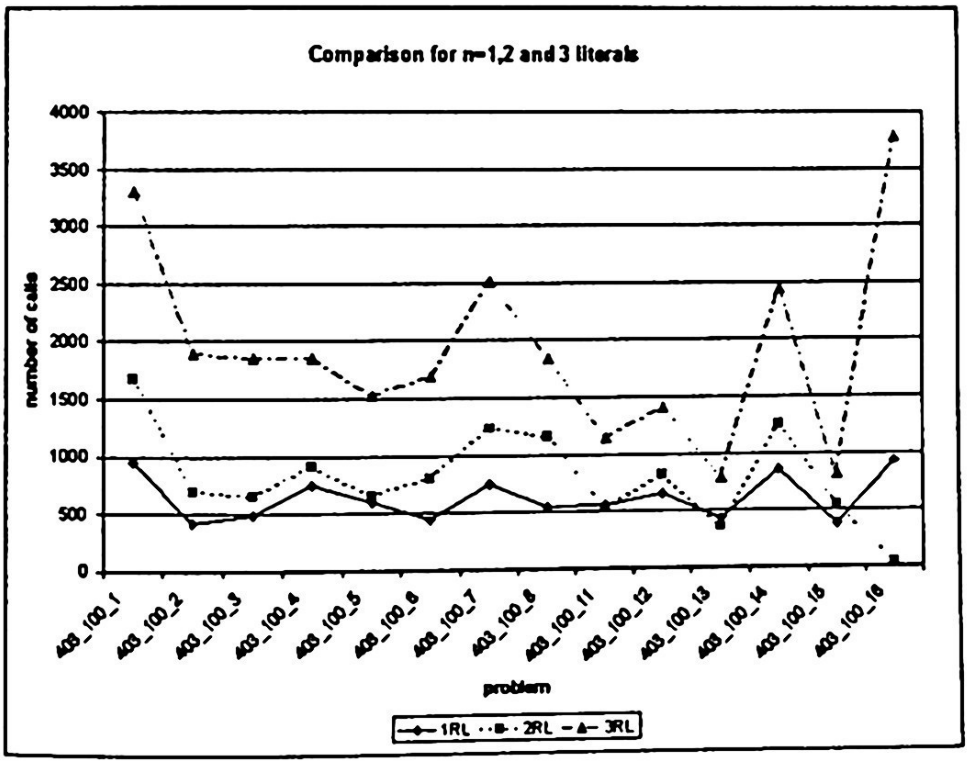

Fig. 7. Number of iterations for the nLR method when n=1, 2 and 3 and hard instances of 403 clauses with 100 literals

Table 3. Set of tests for several values of n

Clauses	Literals	n	Iterations	
			1LR	nLR
400	100	10	9	2
380	102	5	8	3
360	50	3	1	1
450	80	18	9	2
450	150	27	8	1
492	140	2	9	6
300	50	2	9	5
420	110	5	8	3

References

1 Selman, B., Levesque, H. and Mitchell, D.: A new method for solving hard satisfiabi problems, in Proc. of the 10th National Conference on Artificial Intelligence (AAAI San Jose, USA (1992) 440-446.

2 Selman, B. and Kautz, H.: Domain-independent extensions of GSAT: Solving structured satisfiability problems, in Proc. of the Int. Joint Conf. on Artificial Intellige (IJCAI-93), Chambéry, France, (1993) 290-295.

3 Larrabee, T.: Test pattern generation using Boolean satisfiability, in IEEE Trans. Computer Aided Design, Vol. 11, No. 1 (1992) 4-15.

4 Bayardo, R. and Schrag, R.: Using CSP look-back techniques to solve real world instances, in Proc. of the 14th Nat. Conf. on Artificial Intelligence, (AAAI-9 Providence, USA (1997), 203-208.

5 Sosic, R. and Gu, J.: Fast search algorithms for the n-queens problem, in IEEE Trans. Systems, Man and Cybernetics, Vol. 21, No. 6 (1991) 1572-1576.

6 Ashar, P., Ghosh, A. and Devadas, S.: Boolean satisfiability and equivalence check using general Binary Decision Diagrams, in Proc. of IEEE Int. Conf. on Comp Design: VLSI in Computers and Processors, Cambridge, USA (1991) 259-264.

7 Gu, J.: Local search for satisfiability (SAT) problem, in IEEE Trans. on Systems, and Cybernetics, Vol. 23, No. 4 (1993) 1108-1129.

8 Gu, J.: Global Optimization for Satisfiability (SAT) problem, in IEEE Trans. Knowledge and Data engineering, Vol. 6, No. 3 (1994) 361-381.

9 Gu, J., Purdom, O.W., Franco, J. and Wah, B. W.: Algorithms for the satisfiability problem: A Survey, in DIMACS Series on Discrete Mathematics and Theore Computer Science: Satisfiability (SAT) Problem, Vol. 35 (1997) 19-152.

10 Davis, M. and Putnam, H.: A computing procedure for quantification theory, in Journal the ACM, Vol. 7, No. 3 (1960) 201-215.

11 Davis, M., Logemann, G. and Loveland, D.: A machine program for Theorem Proving. Communications of the ACM, Vol 5, No. 7 (1962) 394-397.

12 Frausto-Solis, J. and Sanchez-Ante, G.: Fundamentos de Lógica Computacional, 1st. edition. Trillas, México, 2000.

13 Ruiz-Cuevas, A., Torres-Jimenez, J. and Frausto-Solis, J.: Two literals Rule (2LR): A New Algorithm for Solving Satisfiability (SAT), in Int. Symp. in Intelligent Technologies, Apizaco, Mexico (2000).

14 Cook, S.A.: The complexity of theorem-proving procedures, in Proc. of the 3rd IEEE Symp. on the Foundations of Computer Science, Shaker Heights, USA (1971) 151-158.

15 Gent, I. P. and Walsh, T.: The search for satisfaction, in http://dream.dai.ed.ac.uk/group/tw/sat/sat-survey2.ps (1999).

16 Gilmore, P.C.: A proof method for quantification theory, in Journal fo the ACM, vol. 7 (1960) 201-215.

17 Stojmenovic, I.: Recursive Algorithms in Computer Science Courses: Fibonacci Numbers and Binomial Coefficients, in IEEE Trans. On Education, Vol. 43, No. 3 (2000), 273-276.