Proving unsatisfiability of CNFs locally

Eugene Goldberg

Cadence Berkeley Labs, 2001 Addison,Berkeley,94704,USA

Abstract

We introduce a new method for checking satisfiability of Conjunctive Normal Forms (CNFs). The method is based on the fact that if no clause of a CNF contains a satisfying assignment in its 1-neighborhood, then this CNF is unsatisfiable. (The 1-neighborhood of a clause is the set of all assignments satisfying only one literal of this clause). The idea of 1-neighborhood exploration allows one to prove unsatisfiability without generating an empty clause. The reason for avoiding the generation of an empty clause is that we believe that no deterministic algorithm can efficiently reach a global goal (deducing an empty clause) using an inherently local operation (resolution). At the same time when using 1-neighborhood exploration a global goal is replaced with a set of local subgoals, which makes it possible to optimize local steps of the proof. We introduce two proof systems formalizing 1-neighborhood exploration. An interesting open question is if there exists a class of CNFs for which the introduced systems have proofs that are exponentially shorter than the ones that can be obtained by general resolution.

1 Introduction

Almost all existing deterministic algorithms for solving the satisfiability problem can be described in terms of general resolution [11]. The Davis-Putnam procedure [6], the Davis-Longemann-Loveland algorithm [5], methods of search tree exploration enhanced by clause recording (e.g. [10]), and methods using non-trivial schemes of backtracking like those described in [7], are examples of algorithms that are special cases of general resolution. Since general resolution is a generic method for solving the satisfiability problem, considerable effort has been expended to find of lower bounds on the complexity of resolution proofs. (Henceforth under general resolution we mean non-deterministic general resolution.) In 1985 Haken [8] showed that any resolution proof for pigeonhole CNFs has exponential length. Afterwards, exponential lower bounds on resolution proofs were proven for graph based CNFs [15] and random CNFs [4].

However, little attention has been given to the following issue that is very

important from a practical point of view. Suppose that for a class of CNFs general resolution can always find a proof of polynomial length. Does it mean that there is a "practical" deterministic resolution algorithm that can find such a proof? In [1,3] it was shown that for some classes of CNFs all the proofs in tree-like resolution have exponential length while in general resolution there are proofs of polynomial size. This suggests that the performance of all practical algorithms based on tree-like resolution is far from the performance of general resolution. A bounded resolution was suggested as an alternative to tree-like resolution in [2]. Various methods employing bounded resolution have been used in practice for a long time. A common feature of these methods is that an empty clause is deduced in a set of iterations of increasing complexity. In each iteration every resolvent whose size (number of literals) exceeds a threshold value is immediately thrown away. If an empty clause cannot be deduced in the current iteration, a new iteration begins with the threshold value incremented by 1. In many cases bounded resolution works much better than tree-like resolution. For example, it has been shown in [1] that for so-called graph pebbling CNFs general resolution has linear complexity, tree-like resolution is exponential, and bounded resolution has fourth order complexity. However, bounded resolution can hardly be called a satisfactory solution. Algorithms whose complexity is bounded by high-order polynomials are impractical. Besides, it has not been proven yet that any class of CNFs that has polynomial proofs in general resolution also has proofs of polynomial length in bounded resolution.

The time is ripe to ask whether bridging the gap between non-deterministic and deterministic resolution algorithms is just a matter of time or there is a flaw in the very idea of looking for "good" deterministic algorithms based on general resolution. We believe that the latter is true. Namely, the reason for poor performance of the existing deterministic algorithms is in the following hidden contradiction: a global objective (the deduction of an empty clause) has to be achieved by using an inherently local operation (resolution). This contradiction by no means affects general resolution because it has an oracle at hand to point to the next pair of clauses to resolve. On the other hand, a resolution based determinstic algorithm has to solve this contradiction somehow. It is not hard to see that in the current deterministic resolution algorithms the contradiction is solved by "globalizing" the resolution operation that is by restricting resolution so that it loses its flexibility. In the algorithms based on the Davis-Longemann-Loveland procedure, variable splitting can be considered as a way of planning empty clause deduction by resolving the two clauses deduced in the left and right branches of the search tree. In the Davis-Putnam procedure one has to resolve *all* the clauses containing the variable to be eliminated at the current step. An algorithm using bounded resolution has to keep *all* the resolvents whose size is below a threshold, i.e. it also sacrifices the flexibility of resolution.

It is natural to ask the following question. "Is there a deterministic algorithm such that at each step it manages to make an optimal or sub-optimal (from the viewpoint of deducing an empty clause) choice of the pair of clauses to resolve?" Since currently there is no theory that could answer this question we accept the answer *no* as a postulate. In this paper, we try to resolve the contradiction mentioned above in a different way. Instead of globalizing resolution, we make the objective local. Such an objective is to explore the 1-neighborhood of the clauses of the initial CNF (the 1-neighborhood of a clause C is the set of all assignments satisfying only one literal of C). It is not hard to show (see Section 3) that if a CNF F is satisfiable then the 1-neighborhood of at least one clause of F has to contain a solution. So if no solution is found in the 1-neighborhood of the clauses of a CNF, the latter is unsatisfiable.

We introduce two proof systems formalizing 1-neighborhood exploration. The key idea is to specify 1-neigborhood by marking literals of clauses. (Marking a literal x of a clause C means that 1-neighborhood of clause C in direction x is yet to be explored.) A proof in the proposed system is performed by applying a set of operations correctly transforming literal marking. The proof terminates when either all clauses become unmarked (the original CNF is unsatisfiable) or there appear a clause with a marked literal that cannot be unmarked (the original CNF is satisfiable). So in contrast to general resolution, instead of pursuing the global goal of deducing an empty clause the proof is performed by achieving local subgoals of literal unmarking.

The paper is organized in the following way. In Section 2, we give basic definitions. The main idea of 1-neighborhood exploration is described in Section 3. In Section 4, a proof system formalizing 1-neighborhood exploration is introduced. This proof system is modified in Section 5 to incorporate the resolution operation. The relationship of our approach to local search algorithms is considered in Section 6. Finally, in Section 7 we discuss directions for future research.

2 Main definitions

Definition 1 *A literal of a Boolean variable x is one of the two single argument Boolean functions of x. The identity function is called the* ***positive literal*** *of x and is denoted by x. The complement function is called the* ***negative literal*** *of x and is denoted by $\overline{x}$.*

Definition 2 *A disjunction of literals of Boolean variables not containing opposite literals of the same variable is called a* ***clause****. A conjunction of clauses is called a* ***Conjunctive Normal Form (CNF)****.*

Definition 3 *Given a CNF F,* ***the satisfiability problem*** *is to find an assignment to the variables of F (called* ***a solution*** *to the satisfiability problem) for which F evaluates to 1 or to prove that such an assignment does not exist. A clause C of F is said to be* ***satisfied*** *by assigning value $w \in \{0,1\}$ to a variable x if a literal of x is in C and the assignment of value w sets this literal to 1. We will say that this literal of x is satisfied by the assignment of value w.*

Definition 4 *Let C and C' be clauses having the opposite literals of a variable (say variable x) and there is only one such a variable.* ***The resolvent*** *of C,C' in variable x is the clause that contains all the literals of C and C' but literals x and $\overline{x}$. The operation of producing the resolvent of C and C' is called* ***resolution****.*

Definition 5 *The set of points of the Boolean space for which a clause C evaluates to 0 is called the* ***unsatisfiability cube*** *of C. (By* ***a point of the Boolean n-space*** *we mean a set of assignments to all n variables.) The unsatisfiability cube of C is denoted by $Unsat(C)$. We will say that a point s is covered by a clause C if $s \in Unsat(C)$. A clause C is said to* ***subsume*** *a clause C' if $Unsat(C) \supseteq Unsat(C')$*

Definition 6 *Given a CNF F,* ***general resolution*** *is a method of checking the satisfiability of F by successively applying the resolution operation and adding resolvents to the current CNF. If this procedure results in producing an empty resolvent then F is unsatisfiable. If at some point any new resolvent (that can be produced by resolving two clauses of the current CNF) is subsumed by a clause of the current CNF, then F is satisfiable.*

Definition 7 *The set of n different points of the Boolean n-space is said to be the* ***1-neighborhood of a point*** *s if each point of the set is obtained by changing the value of exactly one variable of s. Denote by $Nbhd(s)$ the 1-neighborhood of point s. The* ***1-neighborhood of a subset*** *S of the Boolean n-space is the set of points p such that $p \in Nbhd(s)$, $s \in S$, $p \notin S$. In other words, a point p is in the 1-neighborhood of S if p is not in S and it is in the 1-neighborhood of a point from S.*

Definition 8 *The 1-neighborhood of the unsatisfiability cube $Unsat(C)$ of clause C is called the* ***1-neighborhood of clause*** *C (written $Nbhd(C)$).*

Remark. *It is not hard to see that the 1-neighborhood of C consists of all the points of the Boolean space that satisfy (i.e. set to 1) exactly one literal of C.*

Definition 9 *The set of assignments satisfying only one literal of clause C, say a literal of variable x, is called* ***the 1-neighborhood of*** *C* ***in direction*** *x (written $Nbhd(C,x)$).*

Definition 10 *Clause C' is said to be* ***symmetric to clause*** *C in variable x if C and C' contain the opposite literals of x and the other literals are identical in both clauses.*

3 Main idea of the method

The idea of 1-neighborhood exploration is based on the following proposition.

Proposition 1 *Let F be a satisfiable CNF consisting of at least one clause. Then there is a solution s (i.e. a point of the Boolean space satisfying all the clauses of F) that lies in the 1-neighborhood of a clause of F.*

Proofs of the propositions are given in the Appendix.

From Proposition 1 it follows that to prove that a CNF F is unsatisfiable one just needs to make sure that there is no solution in the 1-neighborhood of the clauses of F. This means that there is no need to produce an empty clause as the proof of search completeness. At first glance, exploring the 1-neighborhood of all the clauses of F is an appalling task. Suppose, for example, that $C{=}a + b + c$ is one of the clauses of F. The 1-neighborhood of C is specified by set $Unsat(C_1) \cup Unsat(C_2) \cup Unsat(C_3)$ where $C_1 = \overline{a}+b+c$, $C_2 = a+\overline{b}+c$, $C_3 = a+b+\overline{c}$. That is, 1-neighborhood of just one clause occupies 3/8 of the Boolean space. This means that exploring the 1-neighborhood of each clause *separately* is extremely inefficient. The remarkable fact however is that the use of *1-neighborhood inheritance* machinery can potentially make 1-neighborhood exploration very efficient.

Proposition 2 *If S_1 and S_2 are subsets of the Boolean space and $S_1 \subseteq S_2$ then $Nbhd(S_1) \subseteq Nbhd(S_2)$.*

The inheritance of the 1-neighborhood of a clause C means generating a set of new clauses $C_1, \ldots, C_p$ such that $Unsat(C) \subseteq Unsat(C_1) \cup \ldots \cup Unsat(C_p)$. Then according to Proposition 2 the union of 1-neighborhoods of $C_1, \ldots, C_p$ covers the 1-neighborhood of C. That is, with respect to 1-neighborhood exploration, C is replaced with these new clauses. The simplest example of clause replacement is to split a clause on a variable to produce two new clauses. Suppose that C is equal to $a + b + c$ and clauses $C_1 = a + b + c + d$ and $C_2 = a + b + c + \overline{d}$ are obtained from C by splitting on variable d. Since $Unsat(C) = Unsat(C_1) \cup Unsat(C_2)$, then $Nbhd(C) = Nbhd(C_1) \cup Nbhd(C_2)$.

The replacement operation is used for modifying clauses of the current CNF so that 1-neighborhoods of the new clauses are *easier to explore*. The notion of "easier to explore" has at least two meanings. First, one can try to replace a

clause with new ones whose 1-neighborhood in some direction can be trivially checked. Suppose, for example, that clause $C = a+b+c$ has been replaced with $C_1 = a+b+c+d$, $C_2 = a+b+c+\overline{d}$ as described above. Suppose that in the current CNF there is clause $C_3 = \overline{a}+c+d$. Then we can immediately conclude that $Nbhd(C_1, a)$ (i.e. the 1-neighborhood of C_1 in direction a) does not have a solution because no assignment from $Nbhd(C_1, a)$ satisfies C_3. Second, using clause replacement one can try to generate a new smaller set of clauses that inherit the 1-neighborhood of the original set of clauses.

If the initial CNF is unsatisfiable then by applying the operations of clause replacement and 1-neighborhood exploration for "easy-to-check" clauses, we will eventually reach the point that the 1-neighborhood of each clause of the current CNF has been explored and no solution found. Since the clauses of the current CNF have inherited the 1-neighborhood of the clauses of the initial CNF, this means that according to Proposition 1 the latter is unsatisfiable.

4 A proof system for 1-neighborhood exploration

In this section we introduce a system for checking satisfiability of CNFs that formalizes the idea of 1-neighborhood exploration. We will call it System NE (System for 1-Neighborhood Exploration). Let F be a CNF to be checked for satisfiability. To keep track of the unexplored part of 1-neighborhood we use the notion of clause marking. Marking a clause C is to mark literals of C corresponding to the directions in which the 1-neighborhood of C has not been explored yet. For example, if literals b and d are marked in clause $C = a+b+v+d$ then $Nbhd(C, b)$ and $Nbhd(C, d)$ are yet to be explored. It should be noted that the fact that literals a,v are not marked does not necessarily imply that there is no solution in $Nbhd(C, a) \cup Nbhd(C, v)$. It may just mean that this part of $Nbhd(C)$ has been "inherited" by some other clauses.

Definition 11 *Let C be a clause with marked literals. The union of the sets $Nbhd(C, x)$ where x is a marked literal of C is called the* ***marked 1-neighborhood*** *of clause C (written $Mrkd\_Nbhd(C)$).*

Definition 12 *The* ***marked 1-neighborhood of CNF*** *F is the union of the marked 1-neighborhoods of the clauses of F.*

Definition 13 *Clause C is called* ***unmarked*** *(respectively* ***marked*** *) if it does not have (respectively has) a marked literal.*

Definition 14 *Clause C is called* ***inextinguishable*** *in F in direction x (or just inextinguishable for short) if the following two conditions hold:*

- *In clause C there is a literal of variable x and any clause C' of F containing the opposite literal of x also contains a literal of a variable y such that C contains the opposite literal of y.*
- *Any clause C' of F not containing a literal of x (of either polarity) contains a literal of a variable z such that C contains the opposite literal of z.*

Proposition 3 *If clause C is inextinguishable in CNF F in direction x then any assignment from $Nbhd(C, x)$ satisfies all the clauses of CNF F.*

Proposition 4 *Let C be a clause of a CNF F and let $Nbhd(C, x)$ contain a solution s. Then there is a clause subsumed by C that is inextinguishable in direction x.*

Definition 15 *Given a CNF F, a clause C is called* ***complete*** *if it contains literals of all variables appearing in the clauses of F.*

The main idea of checking satisfiability in system NE is as follows. Let F be the CNF to be checked for satisfiability. Initially all the literals of F are marked. So according to Proposition 1 if F is satisfiable, the marked 1-neighborhood of F has to contain a solution. At each step of the proof, a marked clause C of the current CNF is picked and one of the following two kinds of operations is performed on C. Operation of the first kind is to unmark a marked literal of C, say literal x. The unmarking means that either $Nbhd(C, x) \subseteq Unsat(C')$, $C' \in F$ (and so there is no solution in $Nbhd(C, x)$) or $Nbhd(C, x)$ is contained in the marked 1-neighborhood of some other clause of the current CNF. (The latter case of unmarking is done to avoid work duplication). Operation of the second kind is to unmark clause C by adding two clauses C_1 and C_2 obtained by splitting on a variable. The clauses are marked so that $Mrkd\_Nbhd(C) = Mrkd\_Nbhd(C_1) \cup Mrkd\_Nbhd(C_2)$. Clauses C_1 and C_2 are added to the current CNF and all the literals of clause C are unmarked so it becomes an unmarked clause. Clause C remains in the current CNF unless it is subsumed by a clause of the current CNF. Suppose, for example, that C_1 is one of the two clauses produced by splitting C and C_1 becomes unmarked later. Then it can be removed from the current CNF as being subsumed by an existing clause of the current CNF,namely, clause C.

After applying an operation, the marked 1-neighborhood of the current CNF either remains unchanged (operation of the second kind) or is correctly decreased. By correct decreasing we mean removing from the 1-neighborhood of the current CNF points that are proven not to contain a solution. If the initial CNF is unsatisfiable then at some point all the clauses of the current CNF will become unmarked, which means that the 1-neighborhood of the initial CNF does not contain a solution. If the initial CNF is satisfiable then an inextinguishable clause will be produced.

Another way to look at a proof of unsatisfiability produced in System NE is as follows. Let $C_1, \ldots, C_n$ be the set of clauses of a CNF F. By using the splitting operation we eventually produce a set of clauses $F' = \{C'_1, \ldots, C'_p,\}$ $p \geq n$ such that

- CNF F' is equivalent to CNF F.
- Each clause C'_i is subsumed by a clause C_k of F.
- For each clause C'_i it can be trivially proven that $Nbhd(C'_i)$ is solution-free. Namely, if a literal of a variable x is in C'_i , there exists a clause C_k of F subsuming the clause symmetric to C'_i in x.

The 1-neighborhood of the clauses of F' is solution-free and so F' (and hence F) is unsatisfiable.

System NE is specified by the following rules.

(1) All literals of the clauses of the initial CNF F are marked.
(2) At each step of the proof a marked clause C is picked and one of the Rules 3, 4, 5, 6 is applied. These rules can be applied in an arbitrary order.
(3) Clause C is split on a variable x such that clause C does not have a literal of x. This splitting means that clauses $C' = C + x$ and $C'' = C + \overline{x}$ are generated. If a literal of C is marked (not marked) it is marked (respectively not marked) in C' and C''. Literals of the splitting variable (i.e. variable x) are not marked. All literals of C get unmarked and so C becomes an unmarked clause.
(4) If a literal of a variable y is in C and this literal is marked and there is a clause in the current CNF that subsumes the clause symmetric to C in y , then this literal gets unmarked in C.
(5) If a literal of a variable y is in C and this literal is marked and in the current CNF there is a clause C' such that $Mrkd\_Nbhd(C') \supseteq Nbhd(C, y)$ then this literal gets unmarked in C.
(6) If an unmarked clause C is subsumed by some other clause of the current CNF, the former is removed from the current CNF.
(7) If all the clauses in the current CNF are unmarked, then the initial CNF is unsatisfiable (see Proposition 5). Proof terminates.
(8) If in the current CNF F there is a complete clause C' such that a literal of a variable y is marked in C and this literal cannot be unmarked by applying Rules 4, 5, then C' is inextinguishable in direction y (see the proof of Proposition 6). This means that the initial CNF is satisfiable and so proof terminates. The point specified by $Nbhd(C', y)$ is a solution.

Proposition 5 *If all the clauses of the current CNF become unmarked after completing a step of a proof in System NE, then the initial CNF is unsatisfiable.*

Proposition 6 *System NE is sound, that is any conclusion made about the satisfiability of the initial CNF F is correct.*

Proposition 7 *Let C be a complete marked clause obtained when checking CNF F for satisfiability in System NE. Then one can unmark any marked literal of C, say a literal of variable x, by applying Rule 4. If this literal cannot be unmarked then clause C is inextinguishable in direction x and so $Nbhd(C, x)$ contains a solution.*

Proposition 8 *System NE is complete. That is for any CNF F, after a finite number of applications of rules, the "right" termination condition is encountered in System NE.*

5 Enhancing System NE with resolution

In this section we introduce System NER (System for 1-Neighborhood Exploration enhanced by Resolution). In this system, in contrast to System NE, we use two different kinds of unmarking: unconditional and conditional. The notions of unconditional and conditional unmarking distinguish the unmarking done by applying Rules 4 and 5. The reason for such distinction is that we introduce two kinds of resolution: marked and unmarked. Unmarked resolution is applied only to clauses that are unconditionally unmarked and so their 1-neighborhood is solution-free (see Definition 17). The objective of unmarked resolution is to produce new clauses whose 1-neighborhood is solution-free. Marked resolution is applied to clauses that may contain conditionally unmarked literals. The objective of marked resolution is to reduce the number of marked clauses.

Definition 16 *The unmarking of a literal (say literal x) of a clause C is called* ***conditional*** *if x is unmarked by applying Rule 5 in the description of System NE.*

This kind of unmarking does not imply that clause C has no solution in $Nbhd(C, x)$. It just means that $Nbhd(C, x)$ is covered by the marked 1-neighborhood of some clause C'. So one can claim that there is no solution in $Nbhd(C, x)$ under the condition that there is no solution in $Mrkd\_Nbhd(C')$.

Definition 17 *The unmarking of a literal (say literal x) of a clause C is called* ***unconditional*** *if x is unmarked by applying Rule 4 in the description of System NE. Clause C is called* ***unmarked unconditionally*** *if all literals of C are unmarked unconditionally.*

The unconditional unmarking of a literal x in C means that there is no solution in $Nbhd(C, x)$. If a clause is unconditionally unmarked it means that $Nbhd(C)$ does not have solutions.

Let us now consider how System NER benefits from using resolution. Suppose that C is a marked clause and C' is an unconditionally unmarked clause (and so its 1-neighborhood is solution-free) and C' subsumes C. Since $Nbhd(C') \supseteq Nbhd(C)$, then $Nbhd(C)$ is solution-free as well. In System NE we do not have a special unmarking rule covering the described situation because the latter is taken care of by Rule 4. Indeed, since clause C' is unconditionally unmarked, then for each literal of C' (say literal y) there is a clause C^* subsuming the clause symmetric to C in y. Since C' subsumes C then the latter contains literal y as well. It is not hard to see that literal y can be unconditionally unmarked in C by using C^*. Indeed, the clause symmetric to C in y is subsumed by the clause symmetric to C' in y and so is subsumed by C^*. In other words, the set of clauses used for the unconditional unmarking of the literals of C' can be used for the unconditional unmarking of the literals of C. However, if one is allowed to resolve clauses, the situation changes.

Definition 18 *Let C_1 and C_2 be unconditionally unmarked clauses that can be resolved in a variable x. Then their resolvent C is said to be produced by* ***unmarked resolution****.*

Proposition 9 *Let C_1 and C_2 be unconditionally unmarked clauses (and so $Nbhd(C_1)$ and $Nbhd(C_2)$ are solution-free) that can be resolved in a variable x. Then $Nbhd(C)$ is solution-free as well.*

By using unmarked resolution one can produce clauses with fewer literals whose 1-neighborhood is solution-free. Then it is beneficial to have the following special rule. Whenever a marked clause C is subsumed by a clause whose 1-neighborhood is solution-free, C can be unconditionally unmarked. Since resolution "reshapes" clauses this new rule is not subsumed by Rule 4 of System NE.

One more possible application of the resolution operation is to reduce the number of marked clauses (the splitting operation tends to produce a large number of marked clauses). For example, by using resolution one can produce a clause whose marked 1-neighborhood covers marked 1-neighborhoods of a few clauses. For this purpose we need to introduce a "marked" version of resolution that can be incorporated into the 1-neighborhood inheritance paradigm.

Definition 19 *Let C_1 and C_2 be two clauses that can be resolved in a variable x. Clause C is said to be produced from C_1 and C_2 by* ***marked resolution*** *with respect to C_1 if the following conditions hold:*

- *Clause C_1 is marked.*

- *The clause symmetric to C_1 in x is subsumed by C_2.*
- *All marked literals of C_1 (except the literal of x) get marked in C as well.*

Definition 19 gives a very restricted version of resolution, namely, clauses C_1 and C_2 can be resolved in variable x only if C_2 subsumes the clause symmetric to C_1 in x. The reason is that in this case all literals of C_1 can be unmarked (see Proposition 10) because the resolvent inherits the marked 1-neighborhood of C_1. In a general case, when resolving a marked clause C_1 with a clause C_2, the 1-neighborhood of the resolvent does not subsume $Mrkd\_Nbhd(C_1)$. So we restrict marked resolution to make the 1-neighborhood inheritance machinery work.

Proposition 10 *Let clause C be obtained by the marked resolution of C_1 and C_2 with respect to C_1. Then $Mrkd\_Nbhd(C_1) \subseteq Mrkd\_Nbhd(C) \cup Unsat(C)$.*

Now we formulate the rules of System NER. It inherits all the rules of System NE listed in Section 4 with a slight modification due to distinguishing conditional and unconditional marking. We just mention here how a rule of System NE is modified in System NER. In Rule 3 (the splitting of a clause C), clause C becomes conditionally unmarked. In Rule 4, literal y becomes unconditionally unmarked. In Rule 5, literal y becomes conditionally unmarked. Rule 6 is applicable to any clause C whose all literals are unmarked (no matter conditionally or not). The termination condition described in Rule 7 does not distinguish whether the clauses of the current CNF are unmarked conditionally or not. Finally, the termination condition in Rule 8 says about a clause having a literal that cannot be unmarked both conditionally and unconditionally.

System NER has the following three extra rules:

- If a marked clause C is subsumed by a clause C' whose literals are unconditionally unmarked, then all the marked literals of C are unconditionally unmarked.
- Unmarked resolution rule. If C_1 and C_2 can be resolved in a variable x and all their literals are unconditonally unmarked, then the resolvent C of C_1 and C_2 is produced. The literals of C are unconditionally unmarked. This rule is applied only if C is not subsumed by an existing clause of the current CNF.
- Marked resolution rule. If C_1 and C_2 are two clauses of the current CNF that can be resolved by using marked resolution with respect to C_1 then the resolvent C is produced whose literals are marked according to Definition 19. The marked literals of C_1 are conditionally unmarked. The rule is applied only if C is not subsumed by an existing clause of the current CNF.

New rules can be used in combination with Rules 3, 4, 5, 6 of System NE in an arbitrary order. Proposition 6 (that System NE is sound) is applicable to System NER as well and can be proven in the same way. However, one should

revise the proof of completeness because combining the splitting operation with resolution may potentially lead to looping.

Proposition 11 *System NER is complete. That is for any CNF F, after a finite number of applications of rules, the "right" termination condition is encountered in System NER.*

Looping is avoided in System NER by imposing the requirement that no resolvent produced by either resolution rule is subsumed by a clause of the current CNF. Checking for subsumption is quite expensive from a practical point of view. However, a cheaper way to avoid looping can be used in practice (for example, by imposing partial variable ordering). System NER just gives a general framework for using resolution in 1-neighborhood exploration.

We would like to emphasize the difference in using resolution in System NER and in general resolution. In general resolution all resolvents have to eventually contribute into deducing an empty clause. So no two resolvents produced in a general resolution proof are independent. As we already mentioned in the introduction, this suggests that general resolution cannot be implemented by a local deterministic algorithm. In System NER we use two kinds of resolutions. Unmarked resolution is local because we can produce resolvents that are independent of each other. Suppose, for example, that by unmarked resolution we have replaced two sets of unconditionally unmarked clauses M_1 and M_2 with smaller sets of unconditonally unmarked resolvents M'_1 and M'_2. Since we do not have to deduce an empty clause there is no relation between clauses of M'_1 and M'_2 (unless we want to keep resolving these clauses to reduce the number of unconditionally unmarked resolvents even more). Marked resolution is local for a similar reason. Suppose that we have two sets of marked clauses M_1 and M_2. Suppose that by marked resolution we have replaced them with smaller sets of resolvents M'_1 and M'_2 that inherit the marked 1-neighborhood of clauses of M_1 and M_2 respectively. Resolvents from M'_1 and M'_2 are independent in the sense that one can try to unconditionally unmark them independently of each other (unless we want to reduce the set of marked clauses even more).

It is worth mentioning that by being persistent in reducing the number of marked clauses in System NER we can actually deduce an empty clause. For example, we can reduce the set of marked clauses to only two single literal clauses: $C' = \overline{w}$ and $C'' = w$. At this point C' and C'' can be resolved by using the marked resolution rule, which leads to producing an empty clause. Then proof terminates because according to Definition 19 both C', C'' and the resolvent become unconditionally unmarked and so no marked clause is left. (Of course, if even after deducing an empty clause still there are some marked clauses in the current CNF, they can be all unmarked by applying Rule 4). In contrast to general resolution, the steps of the proof in System

NER, in which an empty clause is deduced, can be potentially optimized. This becomes possible because an empty clause is deduced as a "by-product" of 1-neighborhood examination.

6 1-neighborhood exploration and stochastic local search

One of the most significant achievements of the last decade in solving the satisfiability problem has been the development of stochastic local search algorithms like Gsat [14] and Walksat [12]. (Further on by local search we mean stochastic local search). For some classes of CNFs these methods considerably increase the size of satisfiable CNFs for which a solution can be found in a reasonable time. Inspired by this success, Selman et. al. posed the problem of designing "a practical stochastic local search procedure for proving unsatisfiability". This problem was listed under number 5 in the set of ten challenge problems formulated in [13] in 1997 . These problems were suggested to be the focus of attention for the satisfiability problem community in the near future. Judging by the time frame allotted for solving Challenge 5 (which is 5-10 years) it is one of the two most difficult challenges. In [13] a way of solving this problem was also outlined. Namely, it was suggested that the satisfiability problem should be reformulated in the space of resolution proofs so that a CNF unsatisfiable in the space of value assignments can be replaced with a satisfiable CNF. In the space of proofs a satisfying assignment is just an encoding of a resolution proof that the original CNF is unsatisfiable.

It is interesting to analyze this challenge, the way it was proposed to be solved, and local search algorithms in general from the viewpoint of the introduced proof systems. Challenge 5 and the suggested way of solving it were formulated under the assumption that there was no way of proving unsatisfiability locally in the space of value assignments. However, the proof systems we introduced imply that this is not so. In a sense these systems are a (partial) solution to Challenge 5. However, it is an unexpected solution. Instead of proving satisfiability in an artificial space by local search, it is proposed to use a deterministic local algorithm proving unsatisfiability in the original space of value assignments. Moreover, the following assumption looks very plausible. On the basis of the proposed systems (especially System NER) an algorithm can be constructed that will target satisfiable CNFs and be more successful for real life CNFs than the existing local search algorithms. In other words, in a sense, the success of the local search algorithms can be explained by the fact that they are a simple version of a generic algorithm that is applicable to both satisfiable and unsatisfiable CNFs.

It is not hard to see that a local search algorithm also examines the 1-neighborhood of the clauses. Indeed, suppose that the algorithm stops im-

mediately after finding a solution s. The previous assignment s' (from which s was obtained by flipping the value assigned to a variable) was not a solution. So s' does not satisfy a clause and hence s is a point in the 1-neighborhood of this clause. However, a major flaw of the existing local search algorithms is that they explore only the 1-neighborhood of the clauses of the initial CNF. The problem is that there are many implicates of the initial CNF (i.e. clauses that can be deduced from clauses of the initial CNF) that are not in the CNF. Then the number of unsatisfied clauses is not a good measure of how close the current assignment is to a solution because one does not know how many missing clauses are unsatisfied by the current set of assignments. The absence of short implicates may be especially damaging because the shorter a missing clause is the easier it is for the algorithm to "inadvertently" make it unsatisfied. If the number of missing short clauses is large and these clauses are "hard" to deduce (in terms of the number of resolution operations) there is only a very slim chance to find a solution by local search. It seems that the described situation is typical for many real life CNFs. On the other hand, if the share of short implicates among the missing clauses is small or the deduction of missing short clauses is "simple" then local search algorithms may work well.

In contrast to local search algorithms, in System NER the 1-neighborhood of both clauses of the initial CNF and clauses generated by resolution is explored. So, if some important implicates are missing in the initial CNF, an algorithm based on System NER can try to deduce them and so avoid repeatedly making them unsatisfied. In other words, this algorithm combines "in a natural way" 1-neighborhood exploration (which is also the key idea of local search) and generation of new implicates (a feature missing in local search algorithms).

7 Directions for future research

An interesting question is the relationship between general resolution and Systems NE and NER. In particular, an open question is the existence of a class of CNFs for which there is a polynomial length proof in these systems (or at least in System NER) while all the proofs in general resolution are exponential. The reason why such a class may exist is that in System NE and NER the unsatisfiability of a CNF can be proven without deducing an empty clause. For each partial assignment, an algorithm deducing an empty clause eventually produces a resolvent that this assignment does not satisfy. (This idea is the basis for finding lower bounds on the length of resolution proofs [4,8,15]). On the other hand, in the proposed systems, proofs can be performed without deducing an empty clause and so one cannot claim that for each partial assignment a clause unsatisfied by this assignment is produced. A promising candidate to examine is the class of random CNFs [9]. It was shown in [4] that

general resolution has exponential complexity for random CNFs. However, it may be the case that proving unsatisfiability for such CNFs is much easier than generating an empty clause because these CNFs are inherently "local". If this conjecture is true then general resolution just makes random CNFs "appear" to be hard not being able to make use of their natural locality.

From a practical point of view an important direction for research is to try to design an algorithm based on Systems NE and/or NER that is competitive with current solvers, at least for some classes of CNFs (like random CNFs mentioned above).

References

[1] E. Ben-Sasson, R. Impagliazzo, A. Wigderson. Near optimal separation of Treelike and General resolution. Presented as SAT-2000,Third Workshop on the Satisfiability Problem, May 14-18, Renesse,the Netherlands. The paper can be downloaded from web page www.cs.huji.ac.il/~elli/

[2] E. Ben-Sasson, A. Wigderson. Short proofs are narrow - resolution made simple. In the proceedings of STOC 99.

[3] M. Bonet, C. Domingo, N. Galesi, J. Johannsen. Exponential separations between restricted resolutions and cutting planes proof systems. FOCS-1998,638-647.

[4] V. Chvatal, E. Szmeredi. Many hard examples for resolution. J. of the ACM,vol. 35, No 4, pp.759-568.

[5] M. Davis, G. Longemann, D. Loveland. A Machine program for theorem proving. In Communications of the ACM,5:394-397,1962.

[6] M. Davis, H. Putnam. A computing procedure for quantification theory. Journal of the ACM,7,1960,pp.201-215.

[7] M. Ginsberg. Dynamic backtracking. Journal of Artificial Intelligence Research, 1:25-46,1993.

[8] A. Haken. The intractability of resolution. Theoretical Computer Science. 39:297-308,1995.

[9] D. Mitchell, B. Selman, H. Levesque. Hard and easy distribution of sat problems. Proceedings of AAAI-92, San Jose,CA, 459-465.

[10] J. Marques-Silva, K. Sakallah. Grasp - A new search algorithm for satisfiability. Proceedings of International Conference on Computer-Aided Design, November 1996.

[11] J. Robinson. A machine-oriented logic based on resolution principle. Journal of the ACM,12,1965,N 1,23-41.

[12] B. Selman, H. Kautz, B. Cohen. Noise strategies for local search. Proc. of AAAI-94, Seattle, WA, 1994, pp. 337-343.

[13] B. Selman, H. Kautz, D. McAllister. Ten challenges in propositional reasoning and search. Proceedings of IJCAI-97.

[14] B. Selman, H. Levesque, D. Mitchell. A new method for solving hard satisfiability problems. Proc. of AAAI-92, San Jose,CA,1992,440-446.

[15] A. Urquhart. Hard examples for resolution. JACM 34(1):209-219.

A Proofs

Proof of Proposition 1. Let s be a satisfying assignment. We need to show that if s itself is not in the 1-neighborhood of a clause we can always construct such a solution by modifying s. Let C be a clause of F. If s is not in the 1-neighborhood of a clause of F, it satisfies at least two literals of C (and any other clause of F). Then by flipping any value assigned to a variable in s we still get a solution. Let M be the set of the variables whose literals in C are set to 0 by s. Let s' be the vector obtained from s by flipping the value assigned to a variable $y \in M$. If s' is still not in the 1-neighborhood of a clause of F we flip the value of a variable from $M \setminus \{y\}$. After m flips where $1 \leq m \leq |M|-1$ we will either transform s into a solution s^* that is in the 1-neighborhood of C or s^* will reach the 1-neighborhood of some other clause of F.

Proof of Proposition 2. The proof trivially follows from Definition 7.

Proof of Proposition 3. Let s be a point from $Nbhd(C, x)$. Let us assume for the sake of clarity that C contains the positive literal of x. Then value 1 is assigned to x in s. Let us show that s satisfies all the clauses of F. The clauses of F can be divided into the following tree subsets : clauses containing literal x, clauses containing $\overline{x}$ and clauses not containing a literal of x. Let us consider the three subsets separately.

- Clauses containing x. Assignment x=1 in s satisfies any clause containing the positive literal of x including clause C .
- Clauses containing $\overline{x}$. By definition of an inextinguishable clause if C' is a clause containing $\overline{x}$ then C and C' contain opposite literals of a variable y different from x. Then C' is satisfied by the value assigned to y in s.
- Clauses not containing a literal of x. By definition of an inextinguishable clause, if C'' is a clause not containing a literal of x then C'' and C must containin the opposite literals of a variable z. Then C'' is satisfied by the value assigned to z in s.

Proof of Proposition 4. Denote by C' the clause whose unsatisfiability cube consists of the point s' obtained from s by flipping the value of x. It is not hard to see that C' is inextinguishable in direction x. Indeed, since s is a solution then each clause C of F containing $\overline{x}$ must have a literal of a variable y, different from x, such that C' contains the opposite literal of y. (If we assume that a clause C containing $\overline{x}$ does not contain such a literal of a variable y then s does not satisfy C and so cannot be a solution.) For the same reason, any clause not containing a literal of x must contain a literal of a variable z such that C' contains the opposite literal of z. So according to Definition 14, C' is an inextinguishable clause.

Proof of Proposition 5. Assume the contrary i.e. the initial CNF F is satisfiable and at some step of the proof all the clauses of the current CNF become unmarked, which leads to the wrong conclusion that F is unsatisfiable. Since F is satisfiable and in System NE we mark all the clauses of the initial CNF, then according to Proposition 1 a solution s is in $Mrkd\_Nbhd(F)$. Since by our assumption all clauses become unmarked at some point, then a mistake is made at a step of the proof. That is before making this step $s \in Mrkd\_Nbhd(C)$ where C is a clause of the current CNF, while after making this step there is no clause C' of the current CNF such that $s \in Mrkd\_Nbhd(C')$. This can happen only if at this step we apply an operation leading to the unmarking of a literal of C. Then this operation is an application of Rule 3, 4, 5. Consider each of the three posibilities.

- Rule 3. Splitting of clause C on a variable d into clauses C_1 and C_2 cannot lead to "losing" solution s because $Mrkd\_Nbhd(C) = Mrkd\_Nbhd(C_1) \cup Mrkd\_Nbhd(C_2)$ and so s is in the marked 1-neighborhood of C_1 or C_2.
- Rule 4. Suppose that a literal of variable x is in C and it is unmarked because the clause symmetric to C in x is subsumed by a clause C^* of the current CNF. If s is "lost" in this unmarking then it cannot be a solution because s does not satisfy C^*.
- Rule 5. Suppose that a literal of variable x is in C and it is unmarked because $Nbhd(C, x)$ is subsumed by $Mrkd\_Nbhd(C')$ where C' is some other clause of the current CNF. This cannot lead to the loss of solution s. If $s \in Nbhd(C, x)$ then after unmarking the literal of x in C solution s is still in $Mrkd\_Nbhd(C')$ and so it is in the marked 1-neighborhood of the current CNF.

So in all three cases our assumption leads to a contradiction.

Proof of Proposition 6. In System NE we have only two termination points.

- If Rule 7 is applicable we conclude that F is unsatisfiable. According to Proposition 5 this is the correct conclusion.

- If Rule 8 is applicable, we conclude that F is satisfiable. Let us show that this is the correct conclusion. Let C be a complete clause and a literal of a variable x cannot be unmarked in C (assume for the sake of clarity that C contains the positive literal of x) . Let us show that C is indeed inextinguishable in direction x. Since Rule 4 cannot be used for unmarking literal x in C, then each clause of the current CNF containing literal $\overline{x}$ must have a literal of a variable z (different from x) that is the opposite to the literal of z contained in C. Besides, each clause of the current CNF that does not have a literal of x must have a literal of a variable y that is the opposite to the literal of y contained in C. This means that clause C is inextinguishable in direction x in the current (and initial) CNF. Then according to Proposition 3 our conclusion that F is satisfiable is correct.

Proof of Proposition 7. The unsatisfiability cube of C contains only one point of the Boolean space, say point s. Let s' be the point obtained from s by flipping the value assigned to a variable x such that a literal of x is in C and this literal is marked.

- If s is not a solution then there must be a clause C' of the current (and initial) CNF that is unsatisfied by x. This clause subsumes the clause symmetric to C in x. Then after applying Rule 4 this literal of x gets unmarked in C.
- If s is a solution then as it was shown in the proof of Proposition 4 clause C in inextinguishable in direction x.

Proof of Proposition 8. According to Proposition 6 System NE is sound. So to prove the proposition it is sufficient to show that after performing a finite number of steps one of the two possible termination conditions will be reached. Let us show first that there cannot be any deadlock in System NE i.e. we cannot get into the situation when none of the Rules 3, 4, 5, 6 can be applied and a termination condition has not been reached yet. If in the current CNF there is a marked clause C that is not complete then it can be always split on a variable whose literal is not in C yet. If all marked clauses are complete then according to Proposition 7 each marked clause can be either unmarked or proven to be inextinguishable.

Now let us show that the number of steps in a proof in System NE is finite. A clause of F cannot be split into more than 2^n clauses where n is the number of variables in F. Then the number of splitting operations is bounded by $m * 2^n$ where m is the number of clauses in F. Let p be the maximum number of literals occuring in one clause of the initial CNF F. Then the number of marked literals in the current CNF is bounded by $p * m * 2^n$. This value bounds the number of literal unmarking operations. The number of the op-

erations of removing subsumed unmarked clauses (Rule 6) is bounded by the total number of marked clauses that can be produced. The number of marked clauses produced from one clause is bounded by $2^{n+1} - 1$ (this is the number of nodes in a binary tree with 2^n leaves). Then the total number of produced marked clauses (and so the number of aplications of Rule 6) is bounded by $m * (2^{n+1} - 1)$. So after performing a finite number of operations, one of the two termination conditions will be encountered.

Proof of Proposition 9. From Definition 4 it follows that if C is obtained by resolving C_1 and C_2 then $Unsat(C) \subseteq Unsat(C_1) \cup Unsat(C_2)$. Then from Proposition 2 it follows that $Nbhd(C) \subseteq Nbhd(C_1) \cup Nbhd(C_2)$.

Proof of Proposition 10. To prove that $Mrkd\_Nbhd(C_1) \subseteq Mrkd\_Nbhd(C) \cup Unsat(C)$ it is sufficient to show that for any marked literal z in C_1 , $Nbhd(C_1, z)$ is covered either by $Mrkd\_Nbhd(C)$ or by $Unsat(C)$. Any marked literal y of C_1 is also marked in C. Since C subsumes C_1, then $Nbhd(C_1, z) \subseteq Nbhd(C, z)$ and so $Nbhd(C_1, z) \subseteq Mrkd\_Nbhd(C)$. Besides, $Nbhd(C_1, x) \subseteq Unsat(C)$. So even if x is marked in C_1 then $Mrkd\_Nbhd(C_1) \subseteq Mrkd\_Nbhd(C) \cup Unsat(C)$.

Proof of Proposition 11. First let us show that the number of clauses generated by a proof in System NER is finite. There are only three rules producing new clauses: the splitting rule, the marked resolution rule and the unmarked resolution rule. It is not hard to see that the number of clauses generated by applying the two resolution rules is bounded by 3^n where n is the number of variables in the initial CNF. Indeed, neither resolution rule is applied when the resolvent is subsumed by an existing clause of the current CNF. So these two rules do not generate duplicates and the number of generated resolvents is bounded by the total number of implicates of an unsatisfiable CNF of n variables. Then the number of clauses produced by applying the splitting rule is bounded by $3^n * 2^n$ This is true because each of the clauses of the original CNF and each of the resolvents produced by the two resolution rules (the total number of these clauses is bounded by 3^n) can be split into at most 2^n clauses. Finally, the number of unmarking operations is bounded by $3^n * 2^n * p$ where p is the maximal number of literals that occured in a clause of the original CNF.

Using the same arguments as in the proof of Proposition 8 we can show that there cannot be a deadlock in System NER and so after performing a finite number of steps a termination conditon will be encounted.