Resolution Methods for Many-Valued CNF Formulas *

Carlos Ansótegui[1], Ramón Béjar[1], Alba Cabiscol[1], Chu Min Li[2] and Felip Manyà[1]

[1]Departament d'Informàtica i E. Industrial, Universitat de Lleida
Jaume II, 69, E-25001 Lleida, Spain
{carlos,ramon,alba,felip}@eup.udl.es

[2]LaRIA, Université de Picardie Jules Verne
33 Rue Saint Leu, 80039 Amiens Cedex 01, France
cli@laria.u-picardie.fr

1 Introduction

The satisfiability problem of Boolean CNF formulas, or *SAT*, is a central topic in Computer Science and Artificial Intelligence. This is due to several factors: (i) it is one of the syntactically and conceptually simplest NP-complete problems; (ii) it captures the essence of many computationally difficult problems; (iii) the discovery of new systematic (e.g. [2, 19, 20, 26]) and local search (e.g. [24, 25]) algorithms has extended the range and size of *SAT* instances that can be efficiently solved; and (iv) many real-world problems can be efficiently solved by encoding them into *SAT* (e.g. planning [18], and scheduling [10, 12]).

A related problem that has recently attracted the interest of the many-valued logic community is *Signed-SAT*: the satisfiability problem of signed CNF formulas [5]. A *signed CNF formula* is a classical propositional conjunctive clause form based on a generalised notion of literal, called signed literal. Given a truth value set N ($|N| \geq 2$), a *signed literal* is an expression of the form $S : p$, where p is a propositional variable and S, its sign, is a subset of N. A many-valued interpretation I assigns an element of N to each propositional variable, and satisfies a literal $S : p$ iff $I(p) \in S$. Like in classical logic, I satisfies a clause if satisfies at least one literal, and satisfies a formula if satisfies all its clauses. So SAT is *Signed-SAT* in case signs are singletons and $N = \{0, 1\}$.

Two well-studied subproblems of *Signed-SAT* are *Regular-SAT* and *Monosigned-SAT*. *Regular-SAT* is the satisfiability problem of regular CNF formulas. A signed CNF formula is said to be regular in case N is equipped with a total ordering $\leq$ and all the occurring signs are either of the form $\uparrow i = \{j \in N \mid j \geq i\}$ or $\downarrow i = \{j \in N \mid j \leq i\}$ for some $i \in N$. *Monosigned-SAT* is the satisfiability problem of monosigned CNF formulas. A signed CNF formula is said to be monosigned in case all the signs of the literals are singletons.

In the last years, some interesting results for signed, regular and monosigned CNF formulas have been published: (i) NP-completeness proofs [4, 22]; (ii) polynomially solvable subproblems of *Signed-SAT* [3, 14, 17, 22]; (iii) resolution calculi [1, 16, 17]; (iv) clause form translation algorithms [1, 16]; (v) phase transitions phenomenona [8]; and (vi) systematic [17, 21] and local search [7, 9, 15] satisfiability algorithms.

Nowadays, there exist efficient implementations of local search algorithms for *Regular-SAT* [7, 9] and *Monosigned-SAT* [15], and there is experimental evidence that some realistic problems (e.g. graph coloring,

*Research partially supported by the project CICYT TIC2001-1577-C03-03. The first author is supported by a doctoral fellowship of Intelligent Software Components (iSOCO).

quasigroup completion, sports scheduling) can be efficiently solved by encoding them into *Regular-SAT* [6, 9, 11] and *Monosigned-SAT* [15]. At least for the problems considered, the generic problem solving approaches based on *Regular-SAT* and *Monosigned-SAT* outperform or compete with state-of-the-art approaches like constraint programming and *SAT*.

Concerning systematic search, there exists an implementation of a complete *Regular-SAT* solver, Regular-DP [7, 6], but it is not competitive with state-of-the-art *SAT* solvers like Grasp [26], Relsat [2] and Satz [20]. The problem with Regular-DP is that it does not incorporate data structures and variable selection heuristics as powerful as the ones defined for Boolean SAT solvers.

Based on our experience in many-valued logics, in developing efficient satisfiability solvers and in solving combinatorial problems by encoding them into satisfiability problems, we have started a new project. Our goal is to design and implement efficient local and systematic search solvers for a new class of CNF formulas that we call *many-valued CNF formulas* (defined in Section 2), and use them to solve challenging combinatorial problems. These CNF formulas are the union of regular and monosigned CNF formulas.

We consider many-valued CNF formulas because they produce more concise encodings, but at the same time retain the good properties of regular and monosigned CNF formulas for developing satisfiability solvers. Suppose we want to encode the constraint

$$p_1 = 1 \vee p_2 = 3 \vee p_3 \neq 4 \vee p_4 \neq 100$$

and that the domain of values is $N = \{1, \ldots, 100\}$. It can be be easily encoded as a many-valued CNF formula as follows:

$$\{1\} : p_1 \vee \{3\} : p_2 \vee \downarrow 3 : p_3 \vee \uparrow 5 : p_3 \vee \downarrow 99 : p_4$$

Observe that for representing negation we use one or two regular literals, but if we only consider monosigned literals, we need disjunctions of $|N| - 1$ monosigned literals. On the other hand, if we only consider regular literals, we cannot represent in a compact way disjunctions of positive constraints, because a positive constraint, using only regular literals, has to be represented as the conjunction of two regular literals. So, encoding a disjunction of positive constraints using only regular literals produces a CNF formula with an exponential number of clauses.

In this paper we first extend the Davis-Putnam-Loveland (DPL) procedure [13] and then binary and positive unit resolution to the setting of many-valued CNF formulas. Moreover, we prove that the satisfiability problem of many-valued CNF formulas (Many-Valued SAT) is polynomially solvable in case all the clauses of the formula are either binary (Many-Valued 2-SAT) or Horn (Many-Valued Horn SAT). Our motivation in writing this paper is to investigate logical and complexity aspects of many-valued CNF formulas as a first step to develop a many-valued version of the solver Satz [20]. We are therefore in front a theoretical paper motivated by practical applications.

2 Many-Valued CNF Formulas

Definition 1 *A* truth value set*, or* domain*, is a non-empty set $N = \{i_1, i_2, \ldots, i_n\}$, equipped with a total ordering $\leq$. A* sign *is a set $S \subseteq N$ of truth values. For each element i of the truth value set N, let $\uparrow i$ denote the sign $\{j \in N \mid j \geq i\}$, and let $\downarrow i$ denote the sign $\{j \in N \mid j \leq i\}$. A sign S is* regular *if it is identical either to $\uparrow i$ (positive) or to $\downarrow i$ (negative) for some $i \in N$. A sign S is* monosigned *if it is identical to $\{i\}$ for some $i \in N$. A sign S is* many-valued *if it is identical to $\{i\}$, to $\uparrow i$ or to $\downarrow i$ for some $i \in N$.*

Definition 2 *A* many-valued literal *is an expression of the form $S{:}p$, where S is a many-valued sign and p is a propositional variable. A* regular literal *is a many-valued literal with a sign either of the form $\uparrow i$ (positive) or of the form $\downarrow i$ (negative). A* monosigned literal *is a many-valued literal with a monosigned*

```
function MV-DPL (Γ: set of clause) : boolean
var L: literal
begin
  Γ := mv-unit-resolve(Γ);
  if Γ = ∅ then return(true);
  if □ ∈ Γ then return(false);
  L := pick-literal(Γ);
  if MV-DPL (Γ ∪ {L}) then
    return(true)
  else
    return(MV-DPL (Γ ∪ {L̄}))
  endif
end
```

```
function mv-unit-resolve (Γ: set of clause) : set of clause
var L: literal
var C: clause
begin
  while ∃{L} ∈ Γ and □ ∉ Γ do
    Γ := {C| ∄L' ∈ C ∈ Γ such that L ⊆ L'};
    Γ := {C \ {L'|L' ∈ C and L' ⊆ L̄}|C ∈ Γ}
  endwhile;
  return(Γ)
end
```

Figure 1: The MV-DPL procedure

sign. The complement *of a many-valued literal* $L = S : p$ *is* $\overline{L} = (N \setminus S) : p$. *A* many-valued literal $S : p$ subsumes *a many-valued literal* $S' : p'$, *denoted by* $S : p \subseteq S' : p'$, *iff* $p = p'$ *and* $S \subseteq S'$. *A* many-valued clause *is a finite set of many-valued literals. A* many-valued Horn clause *is a many-valued clause containing either at most one regular positive literal or at most one monosigned literal. A* many-valued binary clause *is a many-valued clause containing two literals. A* many-valued CNF formula *is a finite set of many-valued clauses.*

Definition 3 *An* interpretation *is a mapping that assigns to every propositional variable an element of the truth value set. An interpretation* I *satisfies a many-valued literal* $S : p$ *iff* $I(p) \in S$. *An interpretation satisfies a many-valued clause iff it satisfies at least one of its many-valued literals. A many-valued CNF formula* Γ *is* satisfiable *iff there exists at least one interpretation that satisfies all the many-valued clauses in* Γ. *A many-valued CNF formula that is not satisfiable is* unsatisfiable. *The empty many-valued clause, denoted by* $\Box$, *is always unsatisfiable and the empty many-valued CNF formula is always satisfiable.*

3 A DPL Procedure for Many-Valued CNF Formulas

In this section we describe a satisfiability checking procedure for many-valued CNF formulas which is a generalization of the Davis-Putnam-Loveland (DPL) procedure [13]. We first define the concepts of many-valued branching rule and many-valued one-literal rule. Then, we describe the procedure and show that it is a complete algorithm for solving Many-Valued SAT.

Proposition 1 many-valued branching rule *Let Γ be a many-valued CNF formula, let $S:p$ be a many-valued literal occurring in Γ and let N be the truth value set. Then, Γ is satisfiable iff $\Gamma \cup \{S:p\}$ is satisfiable or $\Gamma \cup \{(N \setminus S):p\}$ is satisfiable.*

Proof: Assume that Γ is satisfiable. By definition of interpretation, $I(p) \in N$ for any model I of Γ. If $I(p) \in S$, then $\Gamma \cup \{S:p\}$ is satisfiable. Otherwise, $I(p) \in (N \setminus S)$ and $\Gamma \cup \{(N \setminus S):p\}$ is satisfiable.

Assume that $\Gamma \cup \{S:p\}$ is satisfiable or $\Gamma \cup \{(N \setminus S):p\}$ is satisfiable. If $\Gamma \cup \{S:p\}$ is satisfiable, then Γ is satisfiable. If $\Gamma \cup \{(N \setminus S):p\}$ is satisfiable, then Γ is satisfiable. ■

Proposition 2 many-valued one-literal rule *Let Γ be a many-valued CNF formula that contains a many-valued unit clause $\{S:p\}$. Let Γ' be obtained from Γ by first removing all clauses that contain a literal of the form $S':p$ such that $S \subseteq S'$, and second by removing all occurrences of literals of the form $S'':p$ such that $S \cap S'' = \emptyset$ from the remaining clauses. Then, Γ is satisfiable iff Γ' is satisfiable.*

Proof: Suppose that Γ is satisfiable. Since $\{S:p\} \in \Gamma$, there exists a model I of Γ such that $I(p) \in S$. On the one hand, I is a model of any subset of Γ. On the other hand, if we remove the literals $S'':p$ such that $S \cap S'' = \emptyset$, then I continues being a model of Γ' because $I(p) \notin S''$.

Suppose that Γ' is satisfiable. One can realize that by construction of Γ' there exists a model I of Γ' such that $I(p) \in S$. To obtain Γ from Γ' we add new clauses to Γ' and literals to clauses of Γ'. The addition of literals has no effect on the satisfiability and all the new clauses contain a literal $S':p$ such that $S \subseteq S'$. But I also satisfies these new clauses because $S \subseteq S'$ and $I(p) \in S$. Thus, Γ is satisfiable. ■

The pseudo-code of a satisfiability checking procedure based on the concepts of many-valued branching (cf. Proposition 1) and many-valued one-literal rule (cf. Proposition 2) is shown in Figure 1. The main function is MV-DPL: it returns true if the input many-valued CNF formula Γ is satisfiable, and it returns false if Γ is unsatisfiable. Function mv-unit-resolve applies repeatedly the many-valued one-literal rule and function pick-literal selects the next literal to which the many-valued branching rule is applied.

Theorem 1 *Given a many-valued CNF formula Γ, function MV-DPL terminates and returns either true or false. If it returns true, then Γ is satisfiable. If it returns false, then Γ is unsatisfiable.*

Proof: First, we show that if function MV-DPL terminates, then either some branch of the proof tree created by MV-DPL contains the empty formula (i.e. returns true) or every branch of the proof tree contains the empty clause (i.e. returns false). Suppose we have a proof tree with no branch containing the empty formula and some leaf node of a branch with a many-valued CNF formula Γ' which does not contain the empty clause; we show that MV-DPL does not terminate. If Γ' contains many-valued unit clauses, then MV-DPL calls mv-unit-resolve and applies the many-valued one-literal rule; otherwise, MV-DPL selects a many-valued literal of Γ' and applies the many-valued branching rule. Either way, MV-DPL does not terminate.

Next we show that every proof attempt must terminate. Function MV-DPL applies the many-valued one-literal rule and the many-valued branching rule. On the one hand, the many-valued one-literal rule decreases the number of distinct many-valued literals occurring in the formula. On the other hand, after applying the many-valued branching rule we obtain two new formulas to which MV-DPL applies the many-valued one-literal rule and so decreases the number of distinct many-valued literals occurring in them. Since we began with a finite number of occurrences of distinct literals, these rules can only be applied a finite number of times.

Proposition 2 and Proposition 1 state that Γ is satisfiable iff mv-unit-resolve($\Gamma \cup \{L\}$) is satisfiable or mv-unit-resolve($\Gamma \cup \{\overline{L}\}$) is satisfiable. Then, it is clear that when MV-DPL returns false (i.e all the branches of the proof tree contain the empty clause), the input CNF formula Γ is unsatisfiable, and when MV-DPL returns true (i.e. there is a branch with the empty formula), Γ is satisfiable. ■

4 Resolution for Many-Valued CNF Formulas

In this section we define a refutation complete resolution calculus for many-valued CNF formulas. Moreover, we show that both 2-SAT and Horn-SAT are polynomially solvable.

Definition 4 *Let S:$p \vee D_1$ and S':$p \vee D_2$ be two many-valued clauses. The* many-valued binary resolution rule *is defined as follows:*

$$\frac{S : p \vee D_1 \quad S' : p \vee D_2}{D_1 \vee D_2} \quad \textit{if } S \cap S' = \emptyset \tag{1}$$

Definition 5 *A* proof *of a many-valued clause C from a many-valued CNF formula Γ, denoted by $\Gamma \vdash_{Res} C$, is a finite sequence of clauses $C_1, \ldots, C_m$ such that $C_m = C$ and, for each k $(1 \leq k \leq m)$, either C_k is a clause of Γ or C_k is obtained from C_i and C_j $(k > i, j)$ applying the many-valued binary resolution rule. A proof of the empty clause from a many-valued CNF formula Γ is a* refutation *of Γ.*

In order to prove that the many-valued binary resolution is refutation complete we first give a lemma and then the soundness and completeness proof.

Lemma 1 *Given an unsatisfiable set of many-valued unit clauses Γ, it contains two clauses $S : p$ and $S' : p$ such that $S \cap S' = \emptyset$.*

Proof: Assume that Γ is unsatisfiable and that for any two many-valued unit clauses with the same propositional variable the intersection of their signs is non-empty. We can assume, without loss of generality, that there are at most three literals of the form $S_i : p$ for each propositional variable p: the negative and positive regular literals whose signs have the smaller cardinality (observe that the remaining regular literals are subsumed by these literals) and one monosigned literal $\{k\} : p$ (observe that the intersection of the signs of two monosigned literals with the same variable and different sign is empty). For a given variable p, if there is a monosigned literal $\{k\} : p$, the intersection with the negative and positive regular literals whose signs have the smaller cardinality is $\{k\}$, and therefore the intersection of all the literals containing the variable p is $\{k\}$. Otherwise, we have at most two literals with the variable p and we have assumed that the intersection of the signs of two literals with the same variable is non-empty. We have then that, for every propositional variable p, the intersection of all the signs occurring in literals with the variable p is non-empty. The interpretation I that assigns to every propositional variable p any value of the intersection of all the signs occurring in literals with the variable p clearly satisfies Γ. Thus, Γ must contain two many-valued unit clauses $S : p$ and $S' : p$ such that $S \cap S' = \emptyset$. ∎

Theorem 2 *Many-valued binary resolution is refutation complete.*

Proof: As soundness of rule (1) is straightforward, we only prove completeness. Let $C = \{C_1, C_2, \cdots, C_m\}$ be an unsatisfiable set of many-valued clauses. Let n be the number of excess literals in C (i.e. the total number of literal occurences in C minus the number of clauses in C) and proceed by induction on n. If $n = 0$, then every clause is a unit clause. By Lemma 1, there must be two many-valued clauses $S : p$ and $S' : p$ such that $S \cap S' = \emptyset$. Since they are unit clauses, their resolvent is the empty clause.

Suppose now that there is a resolution proof of any unsatisfiable clause set with at most n excess literals, and suppose that C has $n + 1$ excess literals. At least one clause, say C_1, contains at least two literals. Let C_1' be the result of removing one literal, say L, from C_1, and let C' be the result of replacing C_1 by C_1' in C. Since any satisfying interpretation for C' would satisfy C, C' must be unsatisfiable. Since C' has n excess literals, there exists a resolution proof of C'. That proof produces the empty clause from C'. Thus, if it is applied to C, it will either produce the empty clause or a clause containing only the literal L. That clause may contain several copies of L, but they can be merged to create clause L.

We now have an unsatisfiable set of clauses containing $C'' = \{L, C_2, \cdots, C_m\}$; it is unsatisfiable since any satisfying interpretation would satisfy C. Finally, C'' has fewer excess literals than C, so there is a proof of C''; the two proofs together provide a proof of C. ∎

It turns out that many-valued binary resolution is not refutation complete for signed CNF formulas. For instance, given the truth value set $N = \{1,2,3\}$, the set of clauses $\{\{\{1,2\}:p\},\{\{1,3\}:p\},\{\{2,3\}:p\}\}$ is clearly unsatisfiable but cannot be proved with that rule. The same happens if we incorporate negation to monosigned literals: Observe that $\{\{\{1,2\}:p\},\{\{1,3\}:p\},\{\{2,3\}:p\}\}$ is equivalent to $\{\{\neg\{3\}:p\},\{\neg\{2\}:p\},\{\neg\{1\}:p\}\}$.

A refutation complete calculus for signed CNF formulas is formed by the following rules [23]:

$$\frac{S_1:p \vee D_1 \qquad S_2:p \vee D_2}{(S_1 \cap S_2):p \vee D_1 \vee D_2} \qquad\qquad \frac{\emptyset:p \vee D}{D} \tag{2}$$

Note that, unlike many-valued binary resolution, the literal resolved upon does not necessarily vanish and a so-called *residue* remains.

Proposition 3 *The Many-Valued 2-SAT problem is polynomially solvable.*

Proof: Using rule (1), resolvents of clauses of a many-valued 2-SAT instance have at most two literals. Therefore, the number of different resolvents that can be derived is polynomial in the number of distinct literals occurring in the formula. Since rule (1) is complete for many-valued 2-SAT instances, the many-valued 2-SAT problem is polynomially solvable. ∎

Many-Valued 2-SAT, like Regular 2-SAT and Monosigned 2-SAT, is polynomially solvable but Signed 2-SAT is NP-complete. NP-containment is straightforward to show. The 3-colorability for an undirected graph $G = (V,E)$ is polynomially reducible to Signed 2-SAT: we define three clauses $(\{2,3\}:u \vee \{2,3\}:v) \wedge (\{1,3\}:u \vee \{1,3\}:v) \wedge (\{1,2\}:u \vee \{1,2\}:v)$, where $N = \{1,2,3\}$, for each edge $[u,v] \in E$.

Definition 6 *Let $S:p$ be either a monosigned or a regular positive unit clause and let $S':p \vee D$ be a many-valued Horn clause. The* many-valued positive unit resolution rule (MVPUR) *is defined as follows:*

$$\frac{S:p \quad S':p \vee D}{D} \quad \textit{if } S \cap S' = \emptyset \tag{3}$$

Theorem 3 soundness and completeness *A many-valued Horn formula Γ is unsatisfiable iff there exists a refutation of Γ using the many-valued positive unit resolution (MVPUR) rule.*

Proof: (Soundness) Since the empty clause is unsatisfiable and is obtained after a finite number of applications of the MVPUR rule, it suffices to show that if there exists an interpretation that satisfies both $S:p$ and $S':p \vee D$, and $S \cap S' = \emptyset$, then this interpretation satisfies D as well. Assume that $S:p$ and $S':p \vee D$ are satisfiable. Let I be an interpretation that satisfies both $S:p$ and $S':p \vee D$. Then, it must be that $I(p) \in S$. Since $S \cap S' = \emptyset$, the interpretation I satisfies D as well.

(Completeness) Suppose that Γ is unsatisfiable. We must show that there exists a refutation of Γ using the MVPUR rule. We proceed by induction on l, where l is the length of Γ (i.e. the total number of literal occurrences in Γ). If $l = 0$, then it must be that $\Gamma = \{\Box\}$ because Γ is unsatisfiable. Thus, there exists a refutation of Γ.

Suppose now that there exists a refutation for every unsatisfiable many-valued Horn formula whose length is at most l, and suppose that the length of Γ is $l+1$. Since Γ is unsatisfiable, Γ must contain at least one unit clause with either a monosigned or a regular positive literal (otherwise, Γ is clearly satisfiable). Let $S:p$ be such a literal and let Γ' be the many-valued Horn formula obtained from Γ by applying the many-valued one-literal rule using the many-valued unit clause $\{S:p\}$ (cf. Proposition 2). Then, the length of Γ' is at most l and, by Proposition 2, Γ' is unsatisfiable. By the induction hypothesis, there exists a refutation $\mathcal{R}$ of Γ'. If all the clauses of Γ' appearing in $\mathcal{R}$ are also clauses of Γ, then $\mathcal{R}$

is also a refutation of Γ and we are done. Otherwise, let $C_1, \ldots, C_m$ be the many-valued clauses of Γ' appearing in $\mathcal{R}$ such that the literals $S_1 : p, \ldots, S_k : p$ were deleted from $C_1, \ldots, C_m$ during the application of the many-valued one-literal rule. We put back the eliminated literals $S_1 : p, \ldots, S_k : p$ at these clauses, and we have that the many-valued clauses so obtained belong to Γ. We now apply the MVPUR rule to each one of these clauses together with the many-valued unit clause $\{S : p\}$ used to create Γ'. Since by construction of Γ' we have that $S_1 \cap S = \emptyset, \ldots, S_k \cap S = \emptyset$, we obtain $C_1, \ldots, C_m$ again. Adding these resolution steps to $\mathcal{R}$ we get a refutation of Γ. ∎

Proposition 4 *The Many-Valued Horn SAT problem is polynomially solvable.*

Proof: Using the MVPUR rule, resolvents of clauses of a many-valued Horn instance have less literal occurrences than the parent clause with the biggest number of literal occurrences. Therefore, the number of different resolvents that can be derived is polynomial in the number of distinct literals occurring in the formula. Since the MVPUR rule is complete for many-valued Horn instances, the many-valued Horn problem is polynomially solvable. ∎

The complexity results for Many-Valued 2-SAT and Many-Valued Horn SAT, as well as the straightforward definition of many-valued binary resolution, show that the good properties of regular and monosigned CNF formulas for developing satisfiability solvers remain in many-valued CNF formulas. On the other hand, the use of arbitrary signed CNF formulas does not produce more compact encodings and complicate considerably the underlying logical calculi and the data structures of satisfiability solvers. The complexity and logical results of this paper, together with the much more compact encodings of combinatorial problems that produce the use of many-valued CNF formulas, are the reason for choosing many-valued CNF instead of signed CNF formulas in our project.

References

[1] M. Baaz and C. Fermüller. Resolution-based theorem proving for many-valued logics. *Journal of Symbolic Computation*, 19:353–391, 1995.

[2] R.J. Bayardo and R.C. Schrag. Using CSP look-back techniques to solve real-world SAT instances. In *Proc. of AAAI'97*, pages 203–208, 1997.

[3] B. Beckert, R. Hähnle, and F. Manyà. Transformations between signed and classical clause logic. In *Proceedings, IEEE 29th International Symposium on Multiple-Valued Logics*, pages 248–255, 1999.

[4] B. Beckert, R. Hähnle, and F. Manyà. The 2-SAT problem of regular signed CNF formulas. In *Proceedings, IEEE 30th International Symposium on Multiple-Valued Logics*, pages 331–336, 2000.

[5] B. Beckert, R. Hähnle, and F. Manyà. The SAT problem of signed CNF formulas. In David Basin, Marcello D'Agostino, Dov Gabbay, Seán Matthews, and Luca Viganò, editors, *Labelled Deduction*, volume 17 of *Applied Logic Series*, pages 61–82. Kluwer, Dordrecht, 2000.

[6] R. Béjar. *Systematic and Local Search Algorithms for Regular-SAT*. PhD thesis, Universitat Autònoma de Barcelona, 2000.

[7] R. Béjar and F. Manyà. A comparison of systematic and local search algorithms for regular CNF formulas. In *Proc. of ECSQARU'99*, pages 22–31. Springer LNAI 1638, 1999.

[8] R. Béjar and F. Manyà. Phase transitions in the regular random 3-SAT problem. In *Proc. of ISMIS'99*, pages 292–300. Springer LNAI 1609, 1999.

[9] R. Béjar and F. Manyà. Solving combinatorial problems with regular local search algorithms. In *Proc. of LPAR'99*, pages 33–43. Springer LNAI 1705, 1999.

[10] R. Béjar and F. Manyà. Solving the round robin problem using propositional logic. In *Proc. of AAAI-2000*, pages 262–266, 2000.

[11] R. Béjar, A. Cabiscol, C. Fernández, F. Manyà, and C.P. Gomes. Capturing structure with satisfiability. In *Proc. of CP-2001*, pages 137–152. Springer LNCS 2239, 2001.

[12] J.M. Crawford and A.B. Baker. Experimental results on the application of satisfiability algorithms to scheduling problems. In *Proc. of AAAI'94*, pages 1092–1097, 1994.

[13] M. Davis, G. Logemann, and D. Loveland. A machine program for theorem-proving. *Communications of the ACM*, 5:394–397, 1962.

[14] G. Escalada-Imaz and F. Manyà. The satisfiability problem for multiple-valued Horn formulæ. In *Proceedings, IEEE International Symposium on Multiple-Valued Logics*, pages 250–256, 1994.

[15] A.M. Frisch and T.J. Peugniez. Solving non-boolean satisfiability problems with stochastic local search. In *Proc. of IJCAI-2001*, pages 282–288, 2001.

[16] R. Hähnle. Short conjunctive normal forms in finitely-valued logics. *Journal of Logic and Computation*, 4(6):905–927, 1994.

[17] R. Hähnle. Exploiting data dependencies in many-valued logics. *Journal of Applied Non-Classical Logics*, 6:49–69, 1996.

[18] H.A. Kautz and B. Selman. Unifying SAT-based and graph-based planning. In *Proc. of IJCAI'99*, pages 318–325, 1999.

[19] Chu Min Li. Integrating equivalence reasoning into Davis-Putnam procedure. In *Proc. of AAAI-2000*, pages 291–296, 2000.

[20] Chu Min Li and Anbulagan. Look-ahead versus look-back for satisfiability problems. In *Proc. of CP'97*, pages 341–355. Springer LNCS 1330, 1997.

[21] F. Manyà, R. Béjar, and G. Escalada-Imaz. The satisfiability problem in regular CNF-formulas. *Soft Computing: A Fusion of Foundations, Methodologies and Applications*, 2(3):116–123, 1998.

[22] F. Manyà. The 2-SAT problem in signed CNF formulas. *Multiple-Valued Logic*, 5(4):307–325, 2000.

[23] N.V. Murray and E. Rosenthal. Signed formulas: A liftable meta logic for multiple-valued logics. In *Proc. of the International Symposium on Methodologies for Intelligent Systems, ISMIS'93, Trondheim, Norway*, pages 275–284. Springer LNAI 689, 1993.

[24] B. Selman, H.A. Kautz, and B. Cohen. Noise strategies for improving local search. In *Proc. of AAAI'94*, pages 337–343, 1994.

[25] B. Selman, H. Levesque, and D. Mitchell. A new method for solving hard satisfiability problems. In *Proc. of AAAI'92*, pages 440–446, 1992.

[26] J. P. Marques Silva and K. A. Sakallah. GRASP: A search algorithm for propositional satisfiability. *IEEE Transactions on Computers*, 48(5):506–521, 1999.