Chapter 1

THE SAT PROBLEM OF SIGNED CNF FORMULAS

Bernhard Beckert
University of Karlsruhe
Institute for Logic, Complexity and Deduction Systems
D-76128 Karlsruhe, Germany
beckert@ira.uka.de

Reiner Hähnle
University of Karlsruhe
Institute for Logic, Complexity and Deduction Systems
D-76128 Karlsruhe, Germany
reiner@ira.uka.de

Felip Manyà
Universitat de Lleida
Jaume II, 69, E-25001 Lleida, Spain
felip@eup.udl.es

Abstract *Signed conjunctive normal form* (signed CNF) is a classical conjunctive clause form using a generalised notion of literal, called *signed literal*. A signed literal is an expression of the form $S : p$, where p is a classical atom and S, its *sign*, is a subset of a domain N. The informal meaning is "p takes one of the values in S". Signed formulas are a logical language for knowledge representation that lies in the intersection of the areas *constraint programming* (CP), *many-valued logic* (MVL), and *annotated logic programming* (ALP). This central rôle of signed CNF justifies a detailed study of its subclasses including algorithms for and complexities of associated satisfiability problems (SAT problems). Although signed logic is used since the 1960s, there are only few systematic investigations of its properties. In contrast to work done in ALP and MVL, our present work is a more fine-grained study for the case of propositional CNF. We highlight the most interesting lines of current research: (i) signed versions of some main proponents of classical deduction systems including non-trivial refinements having no classical counterpart; (ii) incomplete local search methods for satisfiability checking of signed formulas; (iii) phase transition phenomena as known, for example, from classical SAT and the influence of the cardinality of N on the crossover point; (iv) the complexity of the SAT problem for signed CNF and its subclasses.

Keywords: Many-valued logic, signed logic, resolution, Davis-Putnam-Loveland procedure, local search, phase transition.

1. INTRODUCTION

Signed formulas are a logical language for knowledge representation that lies in the intersection of the areas *constraint programming* (CP), *many-valued logic* (MVL), and *annotated logic programming* (ALP).

Signed conjunctive normal form (signed CNF) is a classical propositional or first-order conjunctive clause form using a generalised notion of literal, called *signed literal*. A signed literal is an expression of the form $S : p$, where p is a classical atom and S, its *sign*, is a subset of a domain N. The informal meaning is "p takes one of the values in S".

When N is considered to be a truth value set, signed CNF formulas turn out to be a generic representation for finite-valued logics [17]: The problem of deciding the satisfiability of formulas (SAT problem) of any finite-valued logic is in a natural way polynomially reducible to the problem of deciding satisfiability of formulas in signed CNF (signed SAT).

If N is equipped with an ordering, there is a natural notion of signed Horn formula (Definition 4). The particular case where N is lattice-ordered and S is an order filter is investigated in annotated logic programming [21] (there, S is called an *annotation*), therefore, annotated logic programs can be considered as particular signed logic formulas.

Third, $S : p$ can be interpreted as "p is constrained to the values in S" and, hence, as an instance of finite-domain constraint programming [20, 7].

Finally, it is also possible to embed signed formulas into classical monadic first-order logic by representing a signed literal $S : p$, where $S = \{i_1, \ldots, i_r\}$, as the classical formula

$$(\exists p)(s(p)) \wedge (\forall x)(s(x) \leftrightarrow (s(i_1) \vee \cdots \vee s(i_r)))$$

using a unary predicate symbol s.

Applications for deduction in signed logics derive from those of annotated logic programming (e.g., mediated deductive databases), constraint programming (e.g., scheduling), and many-valued logics (e.g., natural language processing). In addition, some problems usually denoted in classical clause logic can be formulated in a better or simply in a different way using signed logic: this comes from the disjunctive interpretation of signs that allows for a compact representation of certain finite-domain first-order properties; and there are additional dimensions along which one can calibrate, namely, the number and ordering of truth values as well as the form of the signs. This claim is supported by first experiments with combinatorial optimisation problems [6]. At the same time, computational complexity of signed logic is mostly comparable to classical logic (see Section 7.). Altogether, signed logic constitutes an interesting trade-off between expressivity and complexity.

The central rôle of signed CNF justifies a detailed study of its subclasses, including algorithms for and complexities of associated SAT problems. In contrast to surveys of ALP [21] and MVL [16, 19], the present chapter constitutes a more fine-grained study into signed formulas within the framework of propositional logic and conjunctive normal form. Although some of the results described here are not yet formally published, the following has the character of a survey, because, given the limited space, we decided to trade in formal proofs for examples and explanations. The reader is invited to consult the technical references given throughout.

In the following section, syntax and semantics of signed CNF are defined formally. Of the remaining sections each captures a specific line of research. Sections 3. and 4. discuss signed versions of some main proponents of classical deduction systems including non-trivial refinements having no classical counterpart. Section 5. focuses on incomplete local search methods for satisfiability checking of signed formulas. In Section 6. we look into the phase transition phenomena well-known from classical satisfiability testing (and other NP-complete problems) and investigate the influence of the cardinality of N on the crossover point. Finally, in Section 7., results proven so far on the complexity of checking satisfiability of formulas in signed CNF (signed SAT) and its subclasses are collected.

2. PRELIMINARIES

2.1 SYNTAX

We assume that a signature, i.e., a denumerable set of propositional variables is given. To form signed literals, the propositional variables (atoms) are adorned with a sign that consists of a finite set of (truth) values.

Definition 1 *A* truth value set *N is a finite set $\{i_1, i_2, \ldots, i_n\}$ where $n \in \mathbb{N}$. The cardinality of N is denoted by $|N|$. A partial order $\leq$ is associated with N, which may be the empty order.*

Definition 2 *A* sign *is a set $S \subseteq N$ of truth values. A* signed literal *is of the form $S : p$ where S is a sign and p is a propositional variable. The* complement *of a signed literal $S : p$, denoted by $\overline{S} : p$, is $(N \setminus S) : p$.*

$$\begin{tikzcd} & 4 \arrow[d, dash] & \\ & 3 \arrow[dl, dash] \arrow[dr, dash] & \\ 1 & & 2 \end{tikzcd}$$

Figure 1.1 A partially ordered truth value set, see Example 5.

A signed clause *is a finite set of signed literals. A signed clause containing exactly one literal is called a* signed unit clause*; and a signed clause containing exactly two literals is called a* signed binary clause. *The empty signed clause is denoted by* $\Box$.

A signed CNF formula *is a finite set of signed clauses. A signed CNF formula whose clauses are binary is called a* signed 2-CNF formula.

The clauses of a signed CNF formula are implicitly conjunctively connected; and the literals in a signed clause are implicitly disjunctively connected. In the following we use $S_1 : p_1 \vee \cdots \vee S_k : p_k$ to represent a signed clause $\{S_1 : p_1, \ldots , S_k : p_k\}$.

Definition 3 *The* length *of a signed clause* C*, denoted by* $|C|$*, is its cardinality. The length of a signed formula* Γ*, denoted by* $|\Gamma|$*, is the sum of the lengths of its signed clauses.*

Definition 4 *For all* $i \in N$*, let* $\uparrow i$ *denote the sign* $\{j \in N \mid j \geq i\}$ *and let* $\downarrow i$ *denote the sign* $\{j \in N \mid j \leq i\}$ *where* $\leq$ *is the partial order associated with* N*. A sign* S *is* regular *if it is identical to* $\uparrow i$ *or to* $\downarrow i$ *for some* $i \in N$.

A signed literal $S : p$ *is a* regular literal *if (a) its sign* S *is regular or (b) its sign* $S = \overline{S'}$ *is the complement of a regular sign* S'.

A signed clause (a signed CNF formula) is a regular clause *(a* regular CNF formula*) if all its literals are regular.*

Example 5 *Let the truth value set* $N = \{1, 2, 3, 4\}$ *be ordered as shown in Figure 1.1, i.e., we use the standard order on natural numbers except that* 1 *and* 2 *are incomparable. Then the signs* $\uparrow 1 = \{1, 3, 4\}$ *and* $\downarrow 1 = \{1\}$ *are regular; and* $\overline{\uparrow 1} = \{2\}$ *and* $\overline{\uparrow 3} = \{1, 2\}$ *are complements of regular signs. The signs* $\{3\}$ *and* $\{1, 4\}$ *are neither regular nor complements of regular signs.*

The complement $\overline{\uparrow 3}$ *of the regular sign* $\uparrow 3$ *is* not *regular as it cannot be represented as* $\uparrow i$ *or* $\downarrow i$ *for any* $i \in N$*. Thus, a regular literal can have a sign that is not regular (but is the* complement *of a regular sign only).*

Whenever the (partial) order on the truth value set is not empty, polarities can be assigned to signed literals in a meaningful way, which gives rise to a generalised notion of Horn clauses.

Definition 6 *A regular sign S is of* positive *(resp.* negative*)* polarity *if it is of the form $\uparrow i$ (resp. $\downarrow i$) for some $i \in N$. A regular literal is of* positive *(*negative*)* polarity *if its sign is of positive (negative) polarity.*

A regular clause is a regular Horn clause *if it contains at most one literal of positive polarity and the signs of all its other literals are complements of signs with positive polarity. A regular CNF formula is a* regular Horn formula *if all its clauses are regular Horn clauses.*

Our notion of regular Horn formula coincides with that of a propositional *annotated logic program* [21].

Example 7 *Using the truth value set N and the associated ordering from the previous example, the clauses (1) $\uparrow 1 : p$, (2) $\uparrow 2 : p \vee \overline{\uparrow 3} : q$, and (3) $\uparrow 4 : q$ are Horn clauses. The regular clause $\uparrow 1 : p \vee \uparrow 2 : q$ is* not *a Horn clause as it contains more than one literal of positive polarity. Since $\downarrow 1 = \overline{\uparrow 2}$ but $\downarrow 4 \neq \overline{\uparrow i}$ for all $i \in N$, the clause $\downarrow 1 : p$ is Horn whereas $\downarrow 4 : p$ is* not *Horn (both clauses are regular).*

Definition 8 *A literal $S : p$ is* monosigned *if its sign $S = \{i\}$ is a singleton. A signed clause (a signed CNF formula) is* monosigned *if all its literals are monosigned.*

Classical two-valued CNF formulas are a special case of monosigned CNF formulas (using a truth value set N with two elements). Monosigned CNF formulas are (trivially) regular w.r.t. the empty ordering.

2.2 SEMANTICS

Definition 9 *An* interpretation *is a mapping that assigns to every propositional variable an element of the truth value set.*

An interpretation I satisfies *a signed literal $S : p$ iff $I(p) \in S$. It* satisfies *a signed clause C iff it satisfies at least one of the signed literals in C; and it* satisfies *a signed CNF formula Γ iff it satisfies all clauses in Γ.*

A signed CNF formula (a signed clause) is satisfiable *iff it is satisfied by at least one interpretation; otherwise it is* unsatisfiable.

Two signed CNF formulas (signed clauses) are equivalent *if they are satisfied by the same interpretations. They are* satisfiability equivalent *iff they are either both satisfiable or both unsatisfiable.*

By definition, the empty signed clause is unsatisfiable and the empty signed CNF formula is satisfiable.

As in classical logic, a Horn formula $C = \overline{\uparrow i_1} : p_1 \vee \cdots \vee \overline{\uparrow i_k} : p_k \vee \uparrow j : q$ is equivalent to the implication $\uparrow i_1 : p_1 \wedge \cdots \wedge \uparrow i_k : p_k \rightarrow \uparrow j : q$, i.e., an interpretation I satisfies C iff it does not satisfy one of $\uparrow i_1 : p_1, \ldots, \uparrow i_k : p_k$ or it satisfies $\uparrow j : q$.

Proposition 10 *For all propositional variables p and all signs $S_1, \ldots, S_k \subseteq N$ ($k \in \mathbb{N}$), the signed clauses*

$$S_1 : p \vee \cdots \vee S_k : p \vee D \;\; \textit{and} \;\; (S_1 \cup \cdots \cup S_k) : p \vee D$$

are equivalent.

The simplification expressed in Proposition 10 is often but not always useful, as its application changes the structure of signs and can, for example, destroy the regularity of a clause.

2.3 CLAUSE FORM TRANSLATION

One of the prominent features of signed CNF formulas is that any formula of any finite-valued logic can be translated in polynomial time into a satisfiability equivalent *signed* CNF formula (the transformation is structure preserving [17]); thus, the SAT problem of a finite-valued logic is polynomially reducible to the signed SAT problem.

In addition, every signed CNF formula can be translated in polynomial time into a satisfiability equivalent *regular* CNF formula with an arbitrary total order on N by the following simple trick: a signed clause containing literals of the form $S : p$ is first transformed into a monosigned clause by replacing $S : p$ with $\bigvee_{i \in S} \{i\} : p$ (using Proposition 10). Then all monosigned literal occurrences are eliminated by replacing a clause $C = \{i\} : p \vee D$ with three clauses $C_1 = \uparrow i : p \vee S : q$, $C_2 = \downarrow i : p \vee S : q$, and $C_3 = D \vee \overline{S} : q$, where q is a new propositional variable not occurring anywhere else and S is an arbitrary regular sign (soundness of this transformation is a direct consequence of rule (1.1) below).

A *direct* polynomial time translation into satisfiability equivalent *regular* CNF formulas was given by Sofronie-Stokkermans [32] for the case that the set of truth values and its associated order form a distributive lattice; it exploits properties of distributive lattices and often produces much less clauses than the general method outlined above.

3. RESOLUTION

In this section we review in a uniform way resolution style calculi for signed CNF formulas and their subclasses that appeared in the literature [15, 17, 18, 28, 29, 25, 32, 33, 3]. The perhaps most straightforward, refutation complete version is formed by the rules below [28].

$$\underset{\text{signed binary resolution}}{\frac{S_1 : p \vee D_1 \qquad S_2 : p \vee D_2}{(S_1 \cap S_2) : p \vee D_1 \vee D_2}} \qquad \underset{\text{simplification}}{\frac{\emptyset : p \vee D}{D}} \tag{1.1}$$

Note that, unlike classical resolution, the literal resolved upon does not necessarily vanish and a so-called *residue* remains. The following parallel resolution rule [15, 28] avoids building residues. Both versions (1.1) and (1.2) were originally thought to require the *merging rule* embodied in Proposition 10 for completeness; however, one can show that it is not necessary [18].

$$\frac{S_1\!:\!p \vee D_1 \quad \cdots \quad S_m\!:\!p \vee D_m}{D_1 \vee \cdots \vee D_m} \quad \text{if } S_1 \cap \cdots \cap S_m = \emptyset \tag{1.2}$$

signed parallel resolution

In the case of monosigned and regular CNF formulas over a totally ordered truth value set, completeness of signed binary resolution is preserved if rule applications generating a residue are not allowed; hence (1.1) can be simplified:

$$\frac{S_1\!:\!p \vee D_1 \qquad S_2\!:\!p \vee D_2}{D_1 \vee D_2} \quad \text{if } S_1 \cap S_2 = \emptyset \tag{1.3}$$

monosigned/regular binary resolution

Completeness of binary resolution, as well as of ordered resolution and hyperresolution, for monosigned CNF formulas is proved by Baaz and Fermüller [1]. If N is totally ordered, one obtains the hyperresolution-like refinements (1.4) and (1.5) of regular binary resolution by combining several applications of rule (1.3) into one [17, 18].

$$\frac{\uparrow i_1 : p \vee D_1 \quad \cdots \quad \uparrow i_m : p \vee D_m \qquad \downarrow j : p \vee D}{D_1 \vee \cdots \vee D_m \vee D} \quad \text{if } (\max_{1 \le k \le m} i_k) > j \tag{1.4}$$

regular resolution

Using the *maximal* i_k in the rule above is not strictly necessary: admitting *any* $i_k > j$ yields a sound rule, but may lead to longer proofs. For regular formulas, (1.4) with $m = 1$ is the same as (1.3).

Example 11 *Let the truth value set be* $N = \{1, 2, 3\}$ *(with the natural order), and let* Γ *be the following regular CNF formula:*

$$\{\downarrow 1 : p_1 \vee \downarrow 2 : p_2,\ \uparrow 2 : p_1 \vee \downarrow 1 : p_2,\ \downarrow 1 : p_1 \vee \uparrow 3 : p_3,$$
$$\uparrow 3 : p_2 \vee \uparrow 2 : p_3,\ \uparrow 3 : p_2 \vee \downarrow 1 : p_3\}$$

The last three clauses resolve to $\downarrow 1 : p_1 \vee \uparrow 3 : p_2$ *by rule (1.4), which in turn resolves to* $\downarrow 1 : p_1$ *with the first clause (by either rule (1.4) or (1.3)). From* *there, one obtains $\downarrow 1 : p_2$ with the second clause. In three more steps the empty clause can be derived.*

$$\begin{array}{c}\dfrac{\downarrow i_1 : p_1 \vee D_1 \quad \cdots \quad \downarrow i_m : p_m \vee D_m \qquad \uparrow j_1 : p_1 \vee \cdots \vee \uparrow j_m : p_m \vee E}{D_1 \vee \cdots \vee D_m \vee E} \\ \text{provided } m \geq 1, i_l < j_l \text{ for all } 1 \leq l \leq m, \\ D_1, \ldots, D_m, E \text{ contain only negative literals} \\ \\ \text{regular negative hyperresolution}\end{array} \tag{1.5}$$

Sofronie-Stokkermans [32, 33] proved that, when clauses only contain positive regular literals or their complements and N is a distributive lattice, an analogue of rule (1.5) is complete where all negative literals of the form $\downarrow i : p$ are replaced with complements of positive literals, i.e., literals of the form $\overline{\uparrow i} : p$. When N is a lattice, the following calculus is complete [3]:

$$\begin{array}{cc}\dfrac{\begin{array}{c}\uparrow i : p \vee D_1 \\ \overline{\uparrow j} : p \vee D_2\end{array}}{D_1 \vee D_2} & \dfrac{\begin{array}{c}\uparrow i : p \vee D_1 \\ \uparrow j : p \vee D_2\end{array}}{\uparrow (i \sqcup j) : p \vee D_1 \vee D_2} \\ \text{if } i \geq j & \text{if neither } i \geq j \text{ nor } j \geq i \\ \text{lattice-regular binary resolution} & \text{lattice-regular reduction}\end{array} \tag{1.6}$$

Note that, when N is totally ordered, the left rule of (1.6) is the same as (1.3) for regular formulas.

Refinements of regular binary resolution being complete for regular Horn formulas over a totally ordered truth value set are regular unit resolution [17] (this corresponds to the case $D_1 = \Box$ in rule (1.3)) and regular positive unit resolution [25] (where, in addition, the unit input clause must be a positive literal). Recently, we proved [3] that the rules below are complete for regular Horn formulas in case N forms an upper semi-lattice.

$$\begin{array}{cc}\dfrac{\begin{array}{c}\uparrow i : p \\ \overline{\uparrow j} : p \vee C\end{array}}{C} & \dfrac{\begin{array}{c}\uparrow i : p \\ \uparrow j : p\end{array}}{\uparrow (i \sqcup j) : p} \\ \text{if } i \geq j & \text{if neither } i \geq j \text{ nor } j \geq i \\ \text{lattice-regular positive unit resolution} & \text{lattice-regular reduction}\end{array} \tag{1.7}$$

Example 12 *Using the upper semi-lattice ordering and regular Horn clauses from Example 7, one may derive $\uparrow 2 : p$ from clauses (2) and (3) by lattice-regular positive unit resolution. The resolvent together with clause (1) gives $\uparrow 3 : p$ by lattice-regular reduction.*

Recall that lattice-based regular Horn formulas are propositional annotated logic programs. As a consequence, the various SLD-style resolution procedures developed for ALP [21, 23, 22] can be used as well. Note, however, that SLD resolution is optimised for first-order logic and is not very efficient on the propositional level.

We close this section with a brief remark on the techniques one can employ to prove completeness of the mentioned resolution calculi. It turns out that semantic tree arguments retain much of their clarity. The most straightforward approach is to use $|N|$-ary semantic trees [17]. Just as in classical resolution theory, more complex refinements are often better handled by inductive construction of a proof, where the number of atoms or atom occurrences in a formula supplies the induction parameter [18].

4. DAVIS-PUTNAM-LOVELAND PROCEDURES

In classical logic, among the most competitive propositional satisfiability solvers are variants of the Davis-Putnam-Loveland procedure (DPL) [9]. In this section we describe the extensions of DPL that have been proposed for signed and regular CNF formulas. They are complete proof procedures for testing the satisfiability of this kind of formulas and seem to be good candidates to implement signed satisfiability solvers.

4.1 THE SIGNED DPL PROCEDURE

The signed DPL procedure (Signed-DPL) is based on the following rules:

Signed one-literal rule: Given a signed CNF formula Γ that contains a signed unit clause $\{S : p\}$,

1. remove all clauses containing a literal $S' : p$ such that $S \subseteq S'$;
2. delete all occurrences of literals $S'' : p$ such that $S \cap S'' = \emptyset$;
3. replace all occurrences of literals $S''' : p$ with $(S''' \cap S) : p$.

Signed branching rule: Reduce the problem of determining whether a signed CNF formula Γ (that contains the propositional variable p) is satisfiable to the problem of determining whether there is an $i \in N$ such that $\Gamma \cup \{\{i\} : p\}$ is satisfiable.

Definition 13 *Given a signed CNF formula Γ that contains a unit clause $\{S : p\}$, let $simplify(\Gamma, S : p)$ denote the result of applying the signed one-literal rule to Γ using the unit clause $\{S : p\}$.*

```
procedure Signed-DPL
Input: a signed CNF formula $\Gamma$ and a truth value set $N = \{i_1, \dots, i_n\}$
Output: “satisfiable” or “unsatisfiable”

begin
 /* signed one-literal rule */
 while $\Gamma$ contains a unit clause $\{S : p\}$ do
   $\Gamma := \mathit{simplify}(\Gamma, S : p)$
 od;
 if $\Gamma = \emptyset$ then return “satisfiable” fi;
 if $\Box \in \Gamma$ then return “unsatisfiable” fi;

 /* signed branching rule */
 let $p$ be a propositional variable occurring in $\Gamma$;
 for $j = 1$ to $n$ do
   if Signed-DPL$(\Gamma \cup \{i_j : p\})$ = “satisfiable” then
     return “satisfiable” fi
 od;
 return “unsatisfiable”
end
```

Figure 1.2 The Signed Davis-Putnam-Loveland procedure (Signed-DPL).

The Signed-DPL procedure is shown in Figure 1.2. It first repeatedly applies the signed one-literal rule. Once the formula cannot be further simplified, it then applies the branching rule and recursively tries to solve each of $|N|$ sub-problems. As these sub-problems by construction contain a signed unit clause, the signed one-literal rule can be applied again. The procedure terminates when either a satisfiable sub-problem is found or all sub-problems have been shown to be unsatisfiable.

Intuitively, Signed-DPL constructs a proof tree using a depth-first strategy. The root node of that tree is labelled with the input formula; the other nodes are labelled with the formulas that result from a single application of the signed one-literal or the signed branching rule to the formula of their parent node. If all the leaves of the tree contain the signed empty clause, the input formula is unsatisfiable; otherwise, if at least one leaf is labelled with the empty signed CNF formula, the input formula is satisfiable.

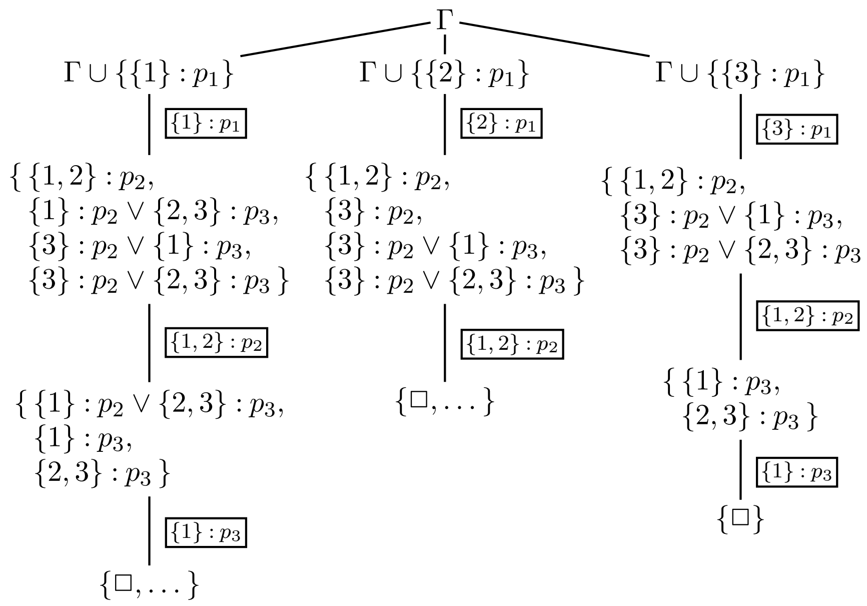
Γ
$\Gamma \cup \{\{1\}:p_1\}$
$\Gamma \cup \{\{2\}:p_1\}$
$\Gamma \cup \{\{3\}:p_1\}$
$\{1\}:p_1$
$\{2\}:p_1$
$\{3\}:p_1$
$\{\{1,2\}:p_2,\ \{1\}:p_2 \vee \{2,3\}:p_3,\ \{3\}:p_2 \vee \{1\}:p_3,\ \{3\}:p_2 \vee \{2,3\}:p_3\}$
$\{\{1,2\}:p_2,\ \{3\}:p_2,\ \{3\}:p_2 \vee \{1\}:p_3,\ \{3\}:p_2 \vee \{2,3\}:p_3\}$
$\{\{1,2\}:p_2,\ \{3\}:p_2 \vee \{1\}:p_3,\ \{3\}:p_2 \vee \{2,3\}:p_3\}$
$\{1,2\}:p_2$
$\{1,2\}:p_2$
$\{1,2\}:p_2$
$\{\{1\}:p_2 \vee \{2,3\}:p_3,\ \{1\}:p_3,\ \{2,3\}:p_3\}$
$\{\Box,\dots\}$
$\{\{1\}:p_3,\ \{2,3\}:p_3\}$
$\{1\}:p_3$
$\{1\}:p_3$
$\{\Box,\dots\}$
$\{\Box\}$

Figure 1.3 A proof tree created by Signed-DPL.

Example 14 *Let the truth value set be $N = \{1,2,3\}$ with an arbitrary order; and let the signed CNF formula Γ consist of the following six clauses:*

$$\begin{array}{ll} \{3\}:p_1 \vee \{1,2\}:p_2 & \{2,3\}:p_1 \vee \{1\}:p_2 \vee \{2,3\}:p_3 \\ \{1,3\}:p_1 \vee \{3\}:p_2 & \{2\}:p_1 \vee \{1,2\}:p_2 \\ \{3\}:p_2 \vee \{1\}:p_3 & \{3\}:p_2 \vee \{2,3\}:p_3 \end{array}$$

Figure 1.3 shows the proof tree created by Signed-DPL for input Γ. Edges corresponding to an application of the signed one-literal rule are labelled with the literal that is used for simplification.

4.2 AN IMPROVED BRANCHING RULE FOR SIGNED-DPL

An application of the branching rule of Signed-DPL from the previous section always creates $|N|$ new sub-branches. In this section, we present an improved branching rule for Signed-DPL that in many cases creates less sub-branches [25].

Definition 15 *Let Γ be a signed CNF formula, and let p be a propositional variable occurring in Γ. Then, the set $N_\Gamma^p \subseteq N$ consists of those truth values that appear in Γ in literals of the form $S : p$.*

Truth values $i, j \in N_\Gamma^p$ are equivalent, denoted by $i \approx_p j$, if, for all literals of the form $S : p$ in Γ, $i \in S$ iff $j \in S$.

The partial order $\preceq_p$ on equivalence classes of N^p_Γ w.r.t. $\approx_p$ is defined by: $\overline{i} \preceq_p \overline{j}$ if, for all literals of the form $S : p$ in Γ, $\overline{i} \subseteq S$ implies $\overline{j} \subseteq S$.

The elements of maximal classes w.r.t. $\preceq_p$ are called maximal truth values *of p in Γ. A set $M = \{i_1, \dots , i_m\} \subseteq N$ is called a* maximal truth value set *of p in Γ if it contains one element of each of the classes $\{\overline{i_1}, \dots , \overline{i_m}\}$ that are maximal w.r.t. $\preceq_p$.*

It can happen that some truth values $i, j \in N$ occur in a formula Γ exactly in the same signs of literals of the form $S{:}p$, i.e., $i \approx_p j$ where $\approx_p$ is the equivalence relation from Definition 15. In that case, it suffices that the branching rule considers only one of the truth values i and j.

In addition, if the truth values of the equivalence class $\overline{j}$ occur (among other signs) in all signs in which the truth values of the equivalence class $\overline{i}$ occur, i.e., if $\overline{i} \preceq_p \overline{j}$, then the truth values in $\overline{i}$ can be ignored by the branching rule of Signed-DPL. This simplification is justified because, if an interpretation I satisfies Γ and $I(p) \in \overline{i}$, then Γ is as well satisfied by every interpretation I' that assigns a truth value from $\overline{j}$ to p and is identical to I for the other propositional variables. Consequently, it is sufficient if the branching rule of Signed-DPL considers only the elements of a maximal truth value set.

Example 16 *Let the set of truth values be $N = \{1, 2, \dots , 7\}$, and let*

$$\begin{array}{lll} \{2,7\} : p \vee D_1 & \{1,3,6\} : p \vee D_2 & \{1,3,4\} : p \vee D_3 \\ \{1,2,3,7\} : p \vee D_4 & \{2,5,7\} : p \vee D_5 & \end{array}$$

be the clauses in the signed CNF formula Γ in which literals of the form $S : p$ occur. Then, the equivalence classes of N^p_Γ w.r.t. $\approx_p$ are $\{2, 7\}$, $\{1, 3\}$, $\{4\}$, $\{5\}$, and $\{6\}$. The maximal elements w.r.t. the order relation $\preceq_p$ are $\{2, 7\}$ and $\{1, 3\}$, because $\{4\} \preceq_p \{1, 3\}$, $\{5\} \preceq_p \{2, 7\}$, and $\{6\} \preceq_p \{1, 3\}$. A maximal truth value set of p in Γ is $\{2, 3\}$.

Proposition 17 *Let Γ be a signed CNF formula, let p be a propositional variable occurring in Γ, and let $\{i_1, \dots , i_m\}$ be a maximal truth value set of p in Γ. Then, Γ is satisfiable iff there is a $k \in \{1, \dots , m\}$ such that $\Gamma \cup \{i_k : p\}$ is satisfiable.*

The branching rule of Proposition 17 can reduce the size of a Signed-DPL proof tree considerably. Consider, for example, the formula Γ from Example 14; $\{2, 3\}$ is a maximal truth value set of p_1 in Γ. Therefore, the leftmost branch of the proof tree for Γ shown in Figure 1.3 is actually redundant and is not constructed if the improved branching rule from Proposition 17 is used. When the branching rule is applied to the formula from Example 16, the number of new sub-branches is reduced from seven to two.

4.3 THE REGULAR DPL PROCEDURE

Signed-DPL is, of course, suitable for the (sub-)class of *regular* CNF formulas as well. However, there are some refinements and special techniques that can be applied. In this section we describe the regular Davis-Putnam-Loveland procedure (Regular-DPL) defined by Hähnle [18] for regular CNF formulas over a totally ordered truth value set. Regular-DPL was the first many-valued DPL-style procedure published and inspired some of the work reported in this survey.

The regular one-literal rule consists of only the first two parts of the signed one-literal rule from Section 4.1, which preserve regularity of the formula; the third part is not needed. The regular branching rule reduces the problem of checking whether a regular CNF formula Γ is satisfiable to the problem of checking whether one of the formulas $\Gamma \cup \{S : p\}$ and $\Gamma \cup \{\overline{S} : p\}$ is satisfiable where $S : p$ is a regular literal occurring in Γ. The branching factor is at most two when the *regular* branching rule is applied, but not all literals containing p are necessarily removed. In contrast to that, the *signed* branching rule removes all occurrences of p, but the branching factor can be as large as the cardinality of the truth value set.

Example 18 *Figure 1.4 shows the proof tree constructed by Regular-DPL for the regular CNF formula Γ from Example 11. As in Figure 1.3, edges corresponding to an application of the one-literal rule are labelled with the literal that is used for simplification.*

The performance of Regular-DPL depends (I) on the data structures used to represent formulas and (II) on the heuristic for selecting the next literal to which the branching rule is applied. Manyà et al. [26] describe an implementation of Regular-DPL that uses suitable data structures and incorporates the regular two-sided Jeroslow-Wang heuristic defined by Hähnle [18]. It is the only DPL-style procedure implemented so far in the framework of signed CNF formulas.

5. LOCAL SEARCH ALGORITHMS

Local search algorithms (LSAs) outperform deductive decision procedures for checking satisfiability of CNF formulas on some problem classes. In particular, this holds for satisfiable hard random 3-SAT instances, which the fastest implementations of DPL cannot solve within a reasonable time limit [30]. In this section, we describe the first LSA that deals with signed CNF formulas and we report some experimental results.

Regular-GSAT [4], an extension of GSAT [31] whose pseudo-code is shown in Figure 1.5, tries to find a satisfying interpretation for a regular CNF formula Γ (with a total order on truth values) performing a greedy local search through the space of interpretations. It starts with a randomly generated interpretation I. If

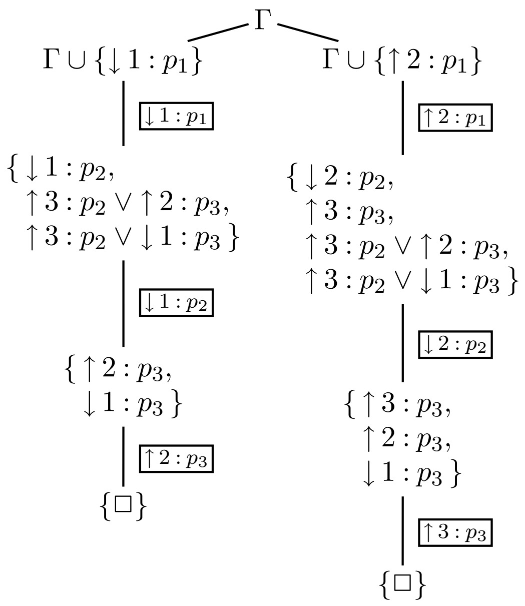
Γ
$\Gamma \cup \{\downarrow 1 : p_1\}$
$\Gamma \cup \{\uparrow 2 : p_1\}$
$\boxed{\downarrow 1 : p_1}$
$\boxed{\uparrow 2 : p_1}$
$\{\downarrow 1 : p_2, \uparrow 3 : p_2 \vee \uparrow 2 : p_3, \uparrow 3 : p_2 \vee \downarrow 1 : p_3\}$
$\{\downarrow 2 : p_2, \uparrow 3 : p_3, \uparrow 3 : p_2 \vee \uparrow 2 : p_3, \uparrow 3 : p_2 \vee \downarrow 1 : p_3\}$
$\boxed{\downarrow 1 : p_2}$
$\boxed{\downarrow 2 : p_2}$
$\{\uparrow 2 : p_3, \downarrow 1 : p_3\}$
$\{\uparrow 3 : p_3, \uparrow 2 : p_3, \downarrow 1 : p_3\}$
$\boxed{\uparrow 2 : p_3}$
$\boxed{\uparrow 3 : p_3}$
$\{\Box\}$
$\{\Box\}$

Figure 1.4 A proof tree created by Regular-DPL.

Table 1.1 Comparison of running times for Regular-GSAT and Regular-DPL.

V	C	*Regular-GSAT*				*Regular-DPL*	
		MaxTries	*MaxChanges*	*time (secs.)*		*time (secs.)*	
				μ	σ	μ	σ
80	487	100	1000	0.68	0.63	1.10	0.85
120	720	200	2800	6.35	5.33	19.45	16.47
160	972	260	6200	25.98	21.48	290.16	325.40
200	1230	400	12000	99.97	88.49	3242.58	3000.16

I does not satisfy Γ, then it creates a set $\mathcal{S}$, formed by those variable-value pairs (p, k) that give rise to a maximal decrease (possibly zero or negative) in the total number of unsatisfied clauses of Γ when the truth value of I at p is changed to k. Next, a propositional variable p' appearing in $\mathcal{S}$ is randomly chosen. Then a truth value k' from $\{k \mid (p', k) \in \mathcal{S}\}$ is randomly chosen. Finally, I is updated to k' at p'. Such changes are repeated until either a satisfying interpretation is found or a pre-set maximum number of changes (MaxChanges) is reached. The whole process is repeated up to MaxTries times, if no satisfying interpretation is found before.

```
procedure Regular-GSAT
Input: a regular CNF formula $\Gamma$, MaxChanges, and MaxTries
Output: either a model of $\Gamma$, or “no satisfying interpretation found”

begin
 for $i = 1$ to MaxTries do
   $I$ := a randomly generated interpretation for $\Gamma$;
   for $j = 1$ to MaxChanges do
     if $I$ satisfies $\Gamma$ then return $I$ fi;
     $S := \{(p, k) \mid$ decrease in number of unsatisfied clauses of $\Gamma$
                maximal, when $I$ changed to $k$ at $p\}$;
     select randomly $p' \in \{p \mid (p, k) \in S\}$;
     select randomly $k' \in \{k \mid (p', k) \in S\}$;
     $I(p') := k'$
   od
 od;
 return “no satisfying interpretation found”
end
```

Figure 1.5 The procedure Regular-GSAT.

Table 1.1 summarises an experiment performed in order to compare the performance of Regular-DPL and Regular-GSAT on satisfiable random regular (signed) 3-SAT instances of the hard region of the phase transition (see Section 6.) with a different number of propositional variables and $|N| = 3$ [4]. Both procedures were applied to 100 satisfiable instances with 80, 120, 160 and 200 propositional variables. In order to obtain more accurate results, each instance was run 50 times with Regular-GSAT. The first column contains the number V of propositional variables and the second the number C of clauses of the instances tested. The remaining columns display the settings of MaxTries and MaxChanges employed, the average μ and the standard deviation σ of the time needed to solve the sets of instances considered. The run time of each instance solved with Regular-DPL corresponds to the time needed to solve that instance, whereas the run time of each instance solved with Regular-GSAT is the average run time over the 50 runs on that instance.

It is clear that local search algorithms for solving regular SAT problems scale better than Regular-DPL when the number of variables in the problem instances increases. This result suggests that local search algorithms, just as their classical counterparts, are good candidates for solving difficult satisfiable problems. First experiments with scheduling problems support this conjecture [6].

Local search algorithms are incomplete and cannot prove unsatisfiability. Recently, some impressive results were obtained by combining deterministic and complete satisfiability procedures (such as DPL) with randomisation to cope with the so-called “heavy-tailed” distribution phenomenon [14]. We expect this to generalise to signed logic as well.

6. PHASE TRANSITIONS

The phase transition phenomenon for the 3-SAT problem consists of two observations: (I) There is a sharp increase (phase transition) of the percentage of unsatisfiable random 3-SAT instances around a certain point when the ratio $\frac{C}{V}$ between the number C of clauses and the number V of variables is varied (at lower ratios, most instances are under-constrained and thus satisfiable, at higher ratios, most instances are over-constrained and thus unsatisfiable). (II) There is an easy-hard-easy pattern in the computational difficulty of solving problem instances as $\frac{C}{V}$ is varied; the hard instances tend to be found near the crossover point.

Phase transitions occur, among other NP-hard problems, in classical [27] and random regular 3-SAT problems [26]. In the present context, our interest in them is twofold: (i) The hard instances described below provide a first testbed to evaluate and compare satisfiability solvers for signed CNF formulas, and (ii) with an eye on knowledge representation with signed CNF formulas, it would be valuable to know what impact the cardinality of N has on the crossover point.

Before we describe the phase transition phenomena in the signed case, we explain how random regular 3-SAT instances are generated. Given a fixed number C of clauses, a number V of propositional variables, and a totally ordered truth value set N, for one problem instance C non-tautological regular clauses are generated. Each regular clause is produced by uniformly choosing three literals with different propositional variables from the set of possible regular literals.

Manyà et al. [26] report on experiments performed on random regular 3-SAT instances with Regular-DPL (see Section 4.3). They observed both aspects (I) and (II) of phase transition. Figure 1.6 visualises this for the random regular 3-SAT problem, where $|N| = 7$ and $V = 60$. Along the vertical axis is the average number of nodes in the proof tree needed to solve a problem instance with Regular-DPL. Along the horizontal axis is the ratio $\frac{C}{V}$ in the test problems. One observes clearly the easy-hard-easy pattern as $\frac{C}{V}$ is varied. The dashed line indicates the percentage of instances found to be satisfiable (the 100 % mark is scaled to the maximum of the curve indicating hardness of problems).

Recent experiments indicate that the location of the crossover point increases logarithmically as a function of the cardinality of the truth value set [5]. Table 1.2 shows the location of the crossover point for different cardinalities of N. The following equation was derived from the experimental crossover points by

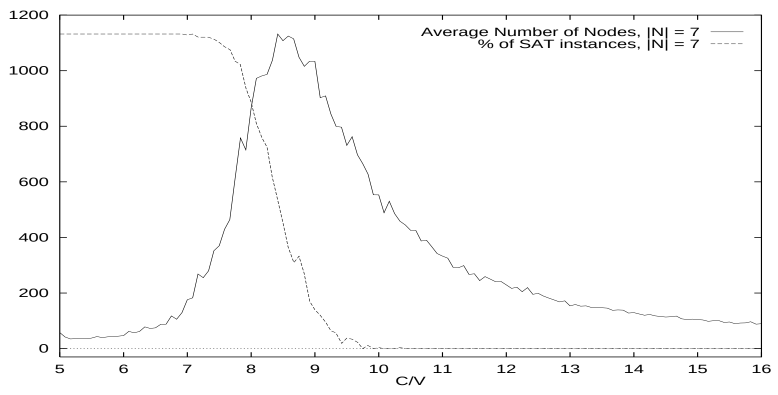

Figure 1.6 Phase transition in the random regular 3-SAT problem.

Table 1.2 Location of the crossover point for different cardinalities of N.

$\|N\|$	*crossover point*	$\|N\|$	*crossover point*	$\|N\|$	*crossover point*
2	4.25	10	9.08	30	10.16
3	6.08	15	9.50	40	10.33
4	7.08	20	9.75	50	10.41
5	7.75	25	10.00	60	10.50

using the Levenberg-Marquardt method for obtaining a non-linear regression model [5]:

$$L\big(|N|\big) = 6.30544\ \ln^{0.391434}\big(|N|\big)$$

7. COMPLEXITY OF THE SIGNED SAT PROBLEM

7.1 OVERVIEW

It is well-known that the classical SAT problem is NP-complete [8]. It is, however, polynomially solvable under certain restrictions. For example, there are linear-time algorithms for solving the classical SAT problem in case all clauses of the formula have at most one positive literal (Horn SAT) [10] and in case all clauses of the formula have at most two literals (2-SAT) [13].

Similar to the classical case, the *signed* SAT problem is NP-complete, but some of its sub-classes are polynomially solvable. In recent years, complexity results for the signed 2-SAT and signed Horn SAT problems have been established. These problems have the truth value set N (resp. $(N, \geq)$) as a second input parameter (besides the formula Γ to be tested for satisfiability). Thus,

Table 1.3 Known complexity results for signed SAT problems.

	SAT	*2-SAT*	*Horn SAT*
classical	NP-compl.	linear [13]	linear [10]
mono-signed	NP-compl.	linear [24]	—
regular, N totally ord.	NP-compl.	polynomial [25]	$\lvert\Gamma\rvert \log \lvert\Gamma\rvert$ [18, 25]
regular, N a distr. lattice, signs of form $\uparrow i$ and $\overline{\uparrow i}$	NP-compl.	NP-compl. [2]	$\lvert\Gamma\rvert\lvert N\rvert^2$ [33]
regular, N a lattice, signs of form $\uparrow i$ and $\overline{\uparrow i}$	NP-compl.	NP-compl.	polynomial [3]
regular, N a lattice, signs of form $\uparrow i$ and $\downarrow i$	NP-compl.	polynomial [2]	—
regular (arbitrary)	NP-compl.	NP-compl.	—
signed (arbitrary)	NP-compl.	NP-compl. [25, 3]	—

signed SAT is the problem of deciding for an arbitrary formula Γ over an arbitrary truth value set N, whether there is an interpretation over N satisfying Γ. One also considers decision problems where N is not an input parameter but fixed, which is denoted by attaching the fixed truth value set N as an index to the name of the decision problem. For example, given a fixed truth value set N, *signed SAT$_N$* is the problem of deciding for an arbitrary formula Γ over N whether there is an interpretation over N satisfying Γ.

NP-containment of the most general problem, signed SAT, is straightforward to show. The classical SAT problem is trivially reducible to signed SAT$_{\{0,1\}}$; therefore, the latter and signed SAT are both NP-complete. Further results are summarised in Table 1.3 and are discussed in Sections 7.2 and 7.3 below.

7.2 THE SIGNED 2-SAT PROBLEM

The signed 2-SAT$_N$ problem for $|N| \geq 3$ and, therefore, the signed 2-SAT problem was proven to be NP-complete by Manyà [25] (as compared to the classical 2-SAT problem that can be solved in linear time); an alternative proof of NP-hardness of signed 2-SAT was later given by Beckert et al. [3]. Manyà [25] reduces the 3-colourability problem of graphs to signed 2-SAT$_N$ to show its NP-hardness, whereas the NP-hard problem Beckert et al. [3] reduce to signed 2-SAT is classical SAT.

The *regular* 2-SAT problem is NP-complete as well; this can be shown by reducing the (general) signed 2-SAT problem to regular 2-SAT [2]. Under certain restrictions, however, satisfiability of regular 2-CNF formulas can be checked in polynomial time. This problem was first considered by Manyà [25] with the additional assumption that N is a totally ordered set. In that case, a refinement of Regular-DPL yields a quadratic-time procedure. A generalisation of this result was proved by Beckert et al. [2]: If N is a lattice and all occurring signs are of the form $\uparrow i$ or the form $\downarrow i$, then regular 2-SAT is polynomially solvable.

A further special case of the regular 2-SAT problem that can be solved in polynomial time is the monosigned 2-SAT problem. By examining the rules of monosigned binary resolution one can check that the number of possible resolvents for a given monosigned 2-CNF formula is polynomial in the number of distinct literals it contains. A quadratic-time procedure for solving monosigned 2-SAT was described by Manyà [25]. He later refined the result by showing that monosigned 2-SAT is solvable in time linear in the length of the formula using a reduction to classical 2-SAT [24].

7.3 THE REGULAR HORN SAT PROBLEM

A Horn fragment is naturally defined if (and only if) the truth value set N is totally ordered or at least a finite lattice.

If N is totally ordered, the problem of deciding whether a regular Horn formula Γ is satisfiable can be solved in time linear in $n = |\Gamma|$ in case $|N|$ is fixed, and in time linear in $n \log n$ otherwise [18]. Algorithms with the same complexity were described in [25]. An algorithm for a particular subclass of regular Horn formulas appeared before [11]; related results can be found in a paper by Escalada-Imaz and Manyà [12].

If N is a finite lattice, regular Horn SAT is solvable in time linear in the length of the formula and polynomial in the cardinality of N via a reduction to the classical Horn SAT problem [3]. For distributive lattices, the more precise bound $n \cdot |N|^2$ was found independently [33], which contains also some results on decidable first-order fragments of regular CNF formulas.

A closer inspection of the proofs in the cited papers yields immediately that all defined regular Horn SAT$_N$ problems have linear complexity.

Acknowledgments

This work was partially supported by EC COST Action 15 *Many-valued Logics for Computer Science Applications.* The third author was partially supported by project TIC96-1038-C04-03 funded by the CICYT and his contribution was carried out during a visit at the University of Karlsruhe with a postdoctoral fellowship of the "Comissionat per a Universitats i Recerca" (1997BEAI400138).

References

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

[2] Bernhard Beckert, Reiner Hähnle, and Felip Manyà. On the regular 2-SAT problem. University of Karlsruhe, Dept. of Computer Science. Available at `ftp://sonja.ira.uka.de/pub/beckert/Regular_2SAT.ps.gz`, 1999.

[3] Bernhard Beckert, Reiner Hähnle, and Felip Manyà. Transformations between signed and classical clause logic. In *Proceedings, 29th International Symposium on Multiple-Valued Logics (ISMVL), Freiburg, Germany*, pages 248–255. IEEE Press, Los Alamitos, 1999.

[4] Ramon Béjar and Felip Manyà. A comparison of systematic and local search algorithms for regular CNF formulas. In *Proceedings, 5th European Conference on Symbolic and Quantitative Approaches to Reasoning with Uncertainty (ECSQARU), London, England*, LNCS 1638, pages 22–31. Springer, 1999.

[5] Ramon Béjar and Felip Manyà. Phase transitions in the regular random 3-SAT problem. In *Proceedings, International Symposium on Methodologies for Intelligent Systems (ISMIS), Warsaw, Poland*, LNCS 1609, pages 292–300. Springer, 1999.

[6] Ramon Béjar and Felip Manyà. Solving combinatorial problems with regular local search algorithms. In *Proceedings, 6th International Conference on Logic for Programming and Automated Reasoning (LPAR), Tbilisi, Georgia*, LNCS 1705, pages 33–43. Springer, 1999.

[7] Thierry Castell and Hélène Fargier. Between SAT and CSP: Propositional satisfaction problems and clausal CSPs. In *Proceedings, European Conference on Artificial Intelligence (ECAI), Brighton, UK*, pages 214–218. John Wiley & Sons, 1998.

[8] Stephen Cook. The complexity of theorem-proving procedures. In *Proceedings, 3rd Annual ACM Symposium on Theory of Computing (STOC), Shaker Heights, USA*, pages 151–158. ACM Press, 1971.

[9] Martin Davis, George Logemann, and Donald Loveland. A machine program for theorem-proving. *Communications of the ACM*, 5(7):394–397, 1962.

[10] William Dowling and Jean Gallier. Linear-time algorithms for testing the satisfiability of propositional Horn formulæ. *Journal of Logic Programming*, 1(3):267–284, 1984.

[11] Gonzalo Escalada-Imaz and Felip Manyà. The satisfiability problem for multiple-valued Horn formulæ. In *Proceedings, International Symposium*

on Multiple-Valued Logics (ISMVL), Boston, USA, pages 250–256. IEEE Press, 1994.

[12] Gonzalo Escalada-Imaz and Felip Manyà. Efficient interpretation of propositional multiple-valued logic programs. In *Advances in Intelligent Computing*, LNCS 945, pages 428–439. Springer, 1995.

[13] S. Even, A. Itai, and A. Shamir. On the complexity of timetable and multicommodity flow problems. *SIAM Journal of Computing*, 5(4):691–703, 1976.

[14] Carla P. Gomes, Bart Selman, and Henry Kautz. Boosting combinatorial search through randomization. In *Proceedings, 15th National Conference on Artificial Intelligence (AAAI), Madison/WI, USA*, pages 431–437. AAAI Press, Menlo Park, 1998.

[15] Reiner Hähnle. Short CNF in finitely-valued logics. In *Proceedings, International Symposium on Methodologies for Intelligent Systems (ISMIS), Trondheim, Norway*, LNCS 689, pages 49–58. Springer, 1993.

[16] Reiner Hähnle. *Automated Deduction in Multiple-Valued Logics*, volume 10 of *International Series of Monographs in Computer Science*. Oxford University Press, 1994.

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

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

[19] Reiner Hähnle and Gonzalo Escalada-Imaz. Deduction in many-valued logics: A survey. *Mathware and Soft Computing*, 4(2):69–97, 1997.

[20] Reiner Hähnle, Ryuzo Hasegawa, and Yasuyuki Shirai. Model generation theorem proving with interval constraints. In F. Benhamou, W. Older, M. van Emden, and P. van Hentenryck, editors, *Proceedings, ILPS Post-Conference Workshop on Interval Constraints, Portland, USA*, 1995.

[21] Michael Kifer and V. S. Subrahmanian. Theory of generalized annotated logic programming and its applications. *Journal of Logic Programming*, 12:335–367, 1992.

[22] Sonia M. Leach and James J. Lu. Query processing in annotationed logic programming: Theory and implementation. *Journal of Intelligent Information Systems*, 6(1):33–58, 1996.

[23] James J. Lu. Logic programming with signs and annotations. *Journal of Logic and Computation*, 6(6):755–778, 1996.

[24] Felip Manyà. The 2-SAT problem in signed CNF formulas. *Multiple-Valued Logic. An International Journal*, 1999. To appear.

[25] Felip Manyà. *Proof Procedures for Multiple-Valued Propositional Logics*. Number 9 in Monografies de l'Institut d'Investigació en Intel.ligència Artificial. IIIA-CSIC, Bellaterra (Barcelona), 1999.

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

[27] David Mitchell, Bart Selman, and Hector Levesque. Hard and easy distributions of SAT problems. In *Proceedings, 10th National Conference on Artificial Intelligence (AAAI), San Jose, USA*, pages 459–465. MIT Press, 1992.

[28] Neil V. Murray and Erik Rosenthal. Signed formulas: A liftable meta logic for multiple-valued logics. In *Proceedings, International Symposium on Methodologies for Intelligent Systems (ISMIS), Trondheim, Norway*, LNCS 689, pages 275–284. Springer, 1993.

[29] Neil V. Murray and Erik Rosenthal. Adapting classical inference techniques to multiple-valued logics using signed formulas. *Fundamenta Informaticae*, 21(3):237–253, 1994.

[30] Bart Selman, Henry A. Kautz, and Bram Cohen. Noise strategies for local search. In *Proceedings, 12th National Conference on Artificial Intelligence (AAAI), Seattle, USA*, pages 337–343. AAAI Press, Menlo Park, 1994.

[31] Bart Selman, Hector Levesque, and David Mitchell. A new method for solving hard satisfiability problems. In *Proceedings, 10th National Conference on Artificial Intelligence (AAAI), San Jose, USA*, pages 440–446. MIT Press, 1992.

[32] Viorica Sofronie-Stokkermans. *Fibered Structures and Applications to Automated Theorem Proving in Certain Classes of Finitely-Valued Logics and to Modeling Interacting Systems*. PhD thesis, Johannes Kepler Universität Linz, Forschungsinstitut für symbolisches Rechnen, 1997.

[33] Viorica Sofronie-Stokkermans. On translation of finitely-valued logics to classical first-order logic. In *Proceedings, 13th European Conference on Artificial Intelligence (ECAI), Brighton, UK*, pages 410–411. John Wiley & Sons, 1998.