Resolution and Clause-Learning with Restarts for Signed CNF Formulas

David Mitchell
Simon Fraser University
mitchell@cs.sfu.ca

Abstract

Motivated by the question of how to efficiently do model finding or theorem proving for multi-valued logics, we study the relative reasoning power of resolution proofs and a natural family of model-finding algorithms for Signed CNF Formulas. The conflict-driven clause learning (CDCL) algorithm for SAT is the basis of model finding software systems (SAT solvers) that have impressive performance on many families of propositional formulas. CDCL with restarts (CDCL-R) has been shown to have essentially the same reasoning power as unrestricted propositional resolution. More precisely, they p-simulate each other. We show that this property generalizes to two families of Signed CNF formulas, those with unrestricted signs, and those where the truth value set is a lattice and all signs are regular. We show that a natural generalization of CDCL-R to these formulas has essentially the same reasoning power as natural generalizations of resolution found in the literature. Moreover, the algorithm efficiently simulates bounded width resolution in these systems. These families of signed formulas are possible reduction targets for a number of multi-valued logics, and thus this algorithm has potential as a basis for efficient implemented reasoning systems for many multi-valued logics.

1 Introduction

Multi-valued logics are among the most established and widely studied formalisms for reasoning with uncertainty. In this paper we consider Signed CNF formulas, as defined, for example, in [10, 6], and study the relative reasoning power of resolution proofs and a natural class of algorithms to decide their satisfiability.

The dominant algorithm in modern implemented model-finders for propositional satisfiability (SAT solvers) is the conflict-driven clause learning algorithm (CDCL)

This research was supported in part by a Natural Sciences and Engineering Research Council of Canada (NSERC) Discovery Grant.

introduced in [15], with restarts [13], here denoted CDCL-R. Good solvers based on CDCL-R have remarkable performance on many families of formulas. Consequently, many practical reasoning tasks are carried out by reduction to propositional CNF, or by adaptation of CDCL to other families of formulas. This leads us consider whether we could best obtain effective model finders or theorem provers for multi-valued logics by reduction to SAT or by adapting the CDCL algorithm to a multi-valued context. The goal of this paper is to contribute to our understanding of the potential of adaptations of CDCL-R to the multi-valued case.

Validity or satisfiability of many multi-valued and fuzzy logics (as well as annotated logics, and others) can be reduced naturally to satisfiability of signed CNF formulas [9]. Natural versions of resolution for signed CNF formulas have been presented in the literature, and it is possible to produce natural variants of the CDCL algorithm for these formulas as well. We study two particular families of signed formulas. One family, denoted here MV-CNF, is a very slight restriction on signed CNF formulas with unrestricted signs; the other family consists of Regular CNF formulas with a domain that is a lattice, here denoted Reg-CNF. Corresponding versions of binary resolution give us proof systems for each family, here denoted MV-RES and Reg-RES, respectively. We give an abstract signed version of the CDCL-R algorithm, denoted MV-CDCL-R, which can be instantiated for either family of formulas and many others.

Here, as is generally the case in proof complexity, we measure the reasoning power of a system in terms of minimum proof length. For an algorithm such as CDCL-R, the corresponding notion is minimum length of execution given optimum decision and restart policies. It has been shown that CDCL-R with unlimited restarts has, up to a small polynomial factor, the same reasoning power as unrestricted propositional resolution [18]. (It is not currently known whether restarts are essential or not. See the brief discussion in Section 7.) In [2] it was shown that that CDCL-R can efficiently refute CNF formulas that have bounded-width refutations.

The main purpose of this paper is to show that these properties generalize to the multi-valued systems in question. The main result is that MV-RES and the MV-CDCL-R algorithm for MV-CNF formulas are of essentially the same efficiency: For unsatisfiable MV-CDCL-R formulas, the minimum size proofs of unsatisfiability in the two systems differ in size by at most a small polynomial. Formally, the two systems are said to p-simulate each other, meaning there are polynomial-time functions mapping any proof in one system to a proof in the other. The same property is shown for Reg-RES and the corresponding Reg-CNF version of MV-CDCL-R.

The largest part of the proof consists of showing that the MV-CDCL-R algorithm can efficiently simulate arbitrary resolution refutations. This simulation yields the

following concrete bounds. A resolution refutation is of size r and width w if it has r clauses and the largest clause has at most w literals. We show that, if Γ is an unsatisfiable MV-CNF formula or Reg-CNF formula with s literals, over n atoms, and has a resolution refutation (in the corresponding resolution proof system) of size r and width w, then there is an execution of MV-CDCL-R that refutes Γ, with the following properties:

1. It implicitly generates a resolution refutation of size $O(wn^2r)$.
2. It can be executed in time $O(wn^2sr)$;

It follows that, for any fixed w, MV-CDCL-R can refute formulas that have width-w resolution refutations in time $O(n^{w+2}s)$, with an implicit resolution refutation constructed of size $O(n^{w+2})$.

Our proof is an adaptation of that in [18], with parts influenced by [2], although our presentation is distinct. Rather than proceed from a detailed examination of CDCL-R, we proceed from the key properties of resolution proofs, to a simplified algorithm that performs the main reasoning in CDCL-R, and then to a highly abstracted version of CDCL-R. This emphasizes the properties in the proofs and in the algorithm that are most relevant, in particular those involving "empowering clauses". Intuitively, a clause C is empowering for a set Γ of clauses if more can be proven by unit resolution from $\Gamma \cup \{C\}$ than from Γ alone. The key properties that are exploited by the proofs are that:

1. Non-trivial resolution refutations can be decomposed into a sequence of derivations of empowering clauses;
2. The clauses derived by the clause learning mechanism in the CDCL algorithm are empowering clauses.

Given a resolution refutation Π of clause set Γ, it is possible to use decision and restart policies to direct CDCL-R to efficiently construct a refutation of Γ that is not much longer than Π. This is done by repeatedly finding an empowering clause from Π, and then causing CDCL-R to generate one or more learned clauses that are "as good as" (in a sense to be made precise) that clause.

The main technical result in [18] is that, for any formula with a resolution refutation Π of length r, CDCL-R can implicitly generate a resolution refutation of length $O(n^4r)$. By slightly more careful counting, we obtain size $O(wn^2r)$, where w is the width of the given refutation Π. For general refutations, with no restriction on clause width, this gives us size $O(n^3r)$.

In [2] it is shown that CDCL-R, with sufficiently many random decisions and sufficiently frequent restarts, with high probability refutes any formula having a

width-w resolution refutation with using at most $O(n^{2w+2})$ conflicts, and therefore an implicit resolution refutation of size $O(n^{2w+3})$. For the restriction to formulas with refutations of width bounded by some fixed w, our deterministic bound gives a resolution refutation size of $O(n^2 r)$ which is $O(n^{w+2})$ because $r = O(n^w)$.

The organization of the paper is as follows. In Section 2 we define signed CNF, MV-CNF and Reg-CNF formulas, along with binary resolution rules for these formulas. The properties of resolution proofs that are central to the proof are defined and established in Section 3. In Section 4 we describe an algorithm that embodies the core reasoning in MV-CDCL-R (as well as standard CDCL-R), while Section 5 shows that repeated calls to this algorithm can refute formulas almost as efficiently as unrestricted resolution. In Section 6 we give our MV-CDCL-R algorithm, define p-simulation and give the main theorem. We briefly discuss applicability issues in Section 7, and conclude in Section 8.

A preliminary version of this paper appeared as [17]. The present version corrects notational problems, simplifies and clarifies the presentation, gives tighter simulations and adds the discussion on applicability.

2 Signed CNF Formulas and Resolution

Let T be a finite set of truth values, $\mathcal{P}$ a countably infinite set of multi-valued atoms and $\prec$ a partial order on T. We assume throughout that T is fixed, so the size of T is a constant in complexity analyses. Signed CNF formulas for T are constructed from literals of the form $p{\sqsubseteq}S$, where S is a non-empty proper subset of T and $p \in \mathcal{P}$. We will use S and R for sets of truth values in literals, and l, often with subscripts, for literals. Complementation of sets of truth values is taken with respect to T, so $\overline{S}$ denotes $\{a \in T \mid a \notin S\}$. If l is the literal $p{\sqsubseteq}S$, its complement $\bar{l}$ is the literal $p{\sqsubseteq}\overline{S}$. A clause is a disjunction of literals, and a formula is a conjunction of clauses. When convenient we identify clauses with sets of literals, and formulas with sets of clauses.

Remark 1. *In the literature on signed formulas, literals are typically written $S:p$, but we prefer the readability of a set-like notation such as $p{\sqsubseteq}S$. Formally S is a sequence of constant symbols denoting elements of T and enumerating the set of values p may take. Following usual practice in the literature, we overload S and use it both for the set of truth values and the string representing this set.*

An assignment τ for formula Γ and truth value set T is a function mapping the propositional atoms of Γ to T. Our assignments will often be partial. Assignment τ satisfies a literal $p{\sqsubseteq}S$ if $\tau(p) \in S$, satisfies a clause C if it satisfies at least one literal

in C, and satisfies CNF formula Γ if it satisfies every clause in Γ. If X is a literal, clause, or formula, we write $\tau \models X$ to indicate that τ satisfies X. A formula or set of formulas Φ entails a formula Γ, written $\Phi \models \Gamma$, if every assignment that satisfies each formula in Φ also satisfies Γ. In particular, if $S \subset R$, then every assignment that satisfies $p{\sqsubseteq}S$ also satisfies $p{\sqsubseteq}R$. We say $p{\sqsubseteq}S$ entails $p{\sqsubseteq}R$, and write $p{\sqsubseteq}S \models p{\sqsubseteq}R$.

The connectives in signed formulas are classical two-valued connectives. Signed CNF formulas are intended as a reduction target so, for example, to find a model of a multi-valued formula ϕ, we would transform it into an appropriate signed CNF formula Γ_ϕ, and then run a model finder for signed CNF. It is the responsibility of the transformation to correctly handle the semantics of non-classical connectives in the source logic. A general, linear-time transformation of formulas from an arbitrary finitely valued logic to Signed CNF is described in [9]

A number of variants of this basic logic have been studied. Typical variants restrict the allowed literals or impose structure on the truth value set T. We will explicitly examine two, although the method can be adapted to some others. (Unfortunately, terminology is not uniform in the literature, so our terms may correspond only roughly to those in some other papers.)

1. **Many-Valued CNF (MV-CNF):** Signed CNF formulas as just described, with the restriction that each atom p occurs in at most one literal in any clause. The order relation $\prec$ plays no role.

2. **Regular CNF over a lattice (Reg-CNF):** Let $\langle T, \prec\rangle$ be a lattice. We call a literal $p{\sqsubseteq}S$ regular for $\langle T, \prec\rangle$ iff S is either the upset $\uparrow a = \{b \in T \mid a \preceq b\}$, or the downset $\downarrow a = \{b \in T \mid b \preceq a\}$ of some $a \in T$. A formula Γ is regular if every literal in Γ is regular.

When making statements that apply to both families of formulas, we sometimes use the terms "signed CNF" or "signed formula", rather than explicitly saying "MV-CNF or Reg-CNF formula".

Example 1. *Consider the signed formulas with $T = \{0, 1\}$. The MV-CNF version is equivalent to the classical case, as is the Reg-CNF version with $0 \prec 1$.*

Example 2. *MV-CNF formulas for $\langle T, \prec\rangle$, with $T = \{0, \frac{1}{d-1}, \frac{2}{d-1} \ \ldots \ 1\}$, for some $d \in \mathbb{N}$, and $\prec$ the standard order on $\mathbb{Q}$, are a natural reduction target for commonly used multi-valued logics, including the finite-valued Łukasiewicz logics. In the analogous Reg-CNF version, literals are restricted to those equivalent to $p < a$ and $p > a$, for some $a \in S$. In MV-RES reasoning, the order on T is ignored.*

2.1 Signed Resolution

Let signed binary resolution be the following derivation rule (where A and B are arbitrary disjunctions).

$$\frac{(p{\sqsubseteq}S \ \vee \ A) \qquad (p{\sqsubseteq}R \ \vee \ B)}{(p{\sqsubseteq}(S \cap R) \ \vee \ A \ \vee \ B)} \tag{1}$$

We say the two antecedent (top) clauses in (1) were resolved on p to produce the resolvent clause. Two literals $p{\sqsubseteq}S$ and $p{\sqsubseteq}R$ clash if $S \neq R$. If $R \cap S$ is empty, we say the clash is annihilating, and otherwise we call the literal $p{\sqsubseteq}(R \cap S)$ the residue. A pair of clashing literals that are annihilating are inconsistent.

As in the classical case, a resolution derivation Π of clause C from a set Γ of clauses is a sequence of clauses $\langle C_1, \ldots, C_r \rangle$, where each C_i is either in Γ or derived from two earlier clauses in Π by the resolution rule, and $C_r = C$. The length, or size, of the derivation is the number of clauses r. A resolution refutation of Γ is a resolution derivation from Γ of the empty clause, denoted $\square$.

A resolution rule is sound and refutation complete for a family of formulas if, for every formula Γ in the family, Γ is unsatisfiable iff it has a refutation constructed using the rule. Rule (1) is the basis of resolution proof systems for our two families of formulas, but we need different variants to obtain a sound and complete proof system for each family.

Resolution for MV-CNF: We obtain a sound and refutation-complete proof system by providing rule (1) with implicit merging and annihilation [10]. That is:

1. (Merging) If two literals $p{\sqsubseteq}S$ and $p{\sqsubseteq}R$ with the same atom p occur in the resolvent, they are replaced by $p{\sqsubseteq}(S \cup R)$.
2. (Annihilation) Whenever $(S \cap R)$ is empty, the "false literal" $p{\sqsubseteq}\varnothing$ is omitted from the resolvent.

We denote the resulting system MV-RES.

Resolution for Reg-CNF: The following restricted signed resolution rule is sound and complete for regular formulas over a lattice [5].

$$\frac{(p{\sqsubseteq}{\uparrow}a \ \vee \ A) \qquad (p{\sqsubseteq}{\downarrow}b \ \vee \ B)}{(A \vee B)} \quad \text{provided} \quad a \not\preceq b. \tag{2}$$

We denote the resulting system by Reg-RES.

The differences in the properties of the resolution rules of these two systems are the main thing to be delt with in adapting the CDCL-R algorithm and the proofs

from [18] to our setting. In MV-RES, literals have complements, and no atom appears in multiple literals of any clause, but resolution is not annihilating and we must allow for the residues. In Reg-RES, resolution is annihilating, but literals need not have complements (more precisely, the complement of a regular literal may not be a regular). Literal complementation and the annihilating property of classical resolution both play central roles in the classical CDCL-R algorithm.

3 Empowering and Absorbed Clauses

The efficient simulation of resolution by CDCL-R relies on a property of resolution refutations involving unit resolution. A unit clause is a clause with exactly one literal, and unit resolution is application of the resolution rule when at least one antecedent is a unit clause. We write $\Gamma \stackrel{u}{\vdash} (l)$, or simply $\Gamma \stackrel{u}{\vdash} l$, if (l) can be derived from Γ by unit resolution alone, and write $\Gamma \stackrel{u}{\vdash} \Box$ if there is a refutation of Γ using only unit resolution. As in the classical case, with appropriate data structures it is possible to check if $\Gamma \stackrel{u}{\vdash} l$ or $\Gamma \stackrel{u}{\vdash} \Box$ in linear time. (A proof of this is provided in Appendix A).

We will use certain sets of literals that are inconsistent with a given clause. For a set or sequence L of literals, we denote by $\widetilde{L}$ the set of literals which are the complements of literals in L. In particular, if C is a clause, then $\widetilde{C}$ is the set of complements of literals in C. Semantically, we view $\widetilde{C}$ as a conjunction of literals that is equivalent to $\neg C$. For Reg-CNF we must ensure all literals are regular, but even if C contains only regular literals $\widetilde{C}$ may not. We define a set $\widehat{L}$ as follows. If $L = l_1, l_2, \ldots l_k$ is a sequence of literals, $\widehat{L}$ denotes the set of all size-k sequences of the form $L' = l'_1, \ldots l'_k$, where each l'_i is a regular literal that is inconsistent with l_i. If L is empty then $\widehat{L}$ contains only the empty sequence.

If Γ is a set of clauses and L a set or sequence of literals, we may write Γ,L as an abbreviation for $\Gamma \cup \{(l) \mid l \in L\}$. If C is a clause of size k then Γ,$\widetilde{C}$ is the set of all clauses from Γ plus k unit clauses corresponding to the k literals in C. Thus Γ,$\widetilde{C} \stackrel{u}{\vdash} \Box$ indicates, intuitively, that the restriction of Γ obtained by setting all literals of C false can be refuted by unit resolution.

Definition 1 (Empowering and Absorbed Clauses)**.** *Let Γ be a set of clauses and C a clause with $\Gamma \models C$. For MV-CNF formula Γ we say C is l-empowering for Γ iff $C = (A \vee l)$ and*

1. Γ,$\widetilde{C} \stackrel{u}{\vdash} \Box$;
2. Γ,$\widetilde{A} \not\stackrel{u}{\vdash} \Box$;

3. For any literal l', if $l' \models l$, then $\Gamma,\widetilde{A} \not\vdash^{\mu} l'$.

For Γ a Reg-CNF formula, we say that C is l-empowering for Γ iff $C = (A \vee l)$ and

1a. $\Gamma,C' \vdash^{u} \square$, for some $C' \in \widehat{C}$;

2a. $\Gamma,A' \not\vdash^{\mu} \square$ for each $A' \in \widehat{A}$;

3a. $\Gamma,A' \not\vdash^{\mu} l'$ for each $A' \in \widehat{A}$, and each l' such that $l' \models l$;

Clause C is empowering for Γ if it is l-empowering for some $l \in C$, and is absorbed by Γ otherwise.

Lemma 1 (Existence of Empowering Clauses)**.** *Let Γ be a set of signed clauses for which $\Gamma \not\vdash^{y} \square$. If Π is a signed resolution refutation of Γ, then Π contains a clause that is empowering for Γ.*

Proof. Let C be the first clause in Π that does not satisfy condition 1 (or 1a, respectively) of Definition 1. Such a clause exists, because $\square$ suffices if no earlier clause does. C is the resolvent of two earlier clauses of Π, say $C_1 = (p \sqsubseteq S_1 \vee A_1)$, and $C_2 = (p \sqsubseteq S_2 \vee A_2)$, where $p \sqsubseteq S_1$ and $p \sqsubseteq S_2$ clash. One of C_1 or C_2 is empowering for Γ. To see this, first observe that both are logically implied by Γ, because they are in Π and signed resolution is sound, and both satisfy condition 1 (respectively, 1a) of Definition 1 by choice of C.

We complete the argument for MV-RES as follows. Both C_1 and C_2 satisfy condition 2 of Definition 1, because $C = (p \sqsubseteq (S_1 \cap S_2) \vee A_1 \vee A_2)$, so if $\Gamma,\widetilde{A_1} \vdash^{u} \square$ or $\Gamma,\widetilde{A_2} \vdash^{u} \square$ then $\Gamma,\widetilde{C} \vdash^{u} \square$, contradicting choice of C. Now, suppose both C_1 and C_2 fail condition 3 of Definition 1. That is, for some $R_1 \subset S_1$ and $R_2 \subset S_2$, we have $\Gamma,\widetilde{A_1} \vdash^{u} (p \sqsubseteq R_1)$ and $\Gamma,\widetilde{A_2} \vdash^{u} (p \sqsubseteq R_2)$. Resolving $(p \sqsubseteq R_2)$ and $(p \sqsubseteq R_1)$ produces the unit clause $(p \sqsubseteq (R_1 \cap R_2))$, where $(R_1 \cap R_2) \subset (S_1 \cap S_2)$. If follows that $\Gamma,\widetilde{C} \vdash^{u} \square$, because $p \sqsubseteq \overline{(S_1 \cap S_2)}$ is in $\widetilde{C}$ and we can resolve it with $p \sqsubseteq (R_1 \cap R_2)$ to obtain $\square$. This again contradicts choice of C, so at least one of C_1 or C_2 is empowering for Γ.

The completion for Reg-RES is just a variation. $C = (A_1 \vee A_2)$, because Reg-RES has no residuals, so if $\Gamma,A_1' \vdash^{u} \square$ for some $A_1' \in \widehat{A_1}$ then $\Gamma,C' \vdash^{u} \square$ for some $C' \in \widehat{C}$ (because $A_1 \subset C$) contradicting choice of C. By the symmetric argument, there is no $A_2' \in \widehat{A_2}$ for which $\Gamma,A_2' \vdash^{u} \square$. So, both C_1 and C_2 satisfy condition 2a. Now, suppose both C_1 and C_2 fail condition 3. Then, for some $A_1' \in \widehat{A_1}$, $A_2' \in \widehat{A_2}$, $R_1 \subset S_1$ and $R_2 \subset S_2$, we have that $\Gamma,A_1' \vdash^{u} (p \sqsubseteq R_1)$ and $\Gamma,A_2' \vdash^{u} (p \sqsubseteq R_2)$. Because Reg-RES has no residuals, we know $S_1 \cap S_2 = \varnothing$, so we also have that $R_1 \cap R_2 = \varnothing$ and $(p \sqsubseteq R_1)$ and $(p \sqsubseteq R_2)$ can be resolved to produce $\square$. So if $C' \in \widehat{C}$ then $\Gamma,C' \vdash^{u} \square$, again contradicting the choice of C. So either C_1 or C_2 is empowering for Γ. $\square$

4 Probing with Learning

The core of the CDCL algorithm can be viewed as a back-and-forth between two tightly related processes, one which guesses at partial assignments, and one which derives new clauses based on these guesses and what follows from them by unit propagation. We first consider an algorithm, which we call Probe-and-Learn, that embodies one stage of this interaction. Describing the algorithm requires some terminology.

For signed formulas, the guesses (called "decisions" in the SAT literature) involve restrictions on assignments, rather than absolute assignments. We will make use of certain sequences of restrictions.

Definition 2 (Proper Restriction Sequence)**.** *A proper restriction sequence (or just "restriction sequence") δ for T and Γ is a sequence $\delta = \langle l_1, l_2, \ldots l_k\rangle$ of distinct literals for T such that:*

1. *If δ contains a literal $p{\sqsubseteq}S$ then Γ has a literal with the atom p;*
2. *The set of literals in δ is satisfiable;*
3. *If l_i is $p{\sqsubseteq}S$, then $\bigcap\{R \mid j < i \text{ and } l_j = p{\sqsubseteq}R\} \cap \overline{S} \neq \varnothing$*

For any non-empty restriction sequence $\delta = \langle l_1, \ldots l_{k-1}, l_k\rangle$, we denote by δ^- the maximal proper prefix $\delta^- = \langle l_1, \ldots l_{k-1}\rangle$.

We will use δ and α for decision sequences. Condition 3 of the definition ensures that each successive literal in the restriction sequence further restricts the allowed assignments.

We say that assignment τ is consistent with restriction sequence δ if no literal in δ is inconsistent with a literal in τ. For literal l and restriction sequence δ, we may say that "δ makes l false" if l is not satisfied by any assignment consistent with δ, and that "δ makes l true" if every assignment consistent with δ satisfies l.

Unit propagation is a key component of CDCL algorithms, and in particular of the "back-and-forth" process embodied in Probe-and-Learn. We will define unit propagation for signed formulas, as used in our algorithm, in terms of restriction sequences.

Definition 3 ($\mathrm{UP}(\Gamma, \delta)$)**.** *For any clause set Γ and restriction sequence δ for Γ, we denote by* $\mathrm{UP}(\Gamma, \delta)$ the restriction sequence δ' defined by the fixpoint of the following operation:

> If Γ contains a clause $C = (l \vee B)$ where δ makes every literal of B false, but does not make l either true or false, extend δ with l.

Algorithm 1: Probe-and-Learn

Input: Clause set Γ; truth value set $\langle T, \prec \rangle$; restriction sequence δ.
Output: Clause set Γ'; restriction sequence δ'.

```
α ← a minimal extension of δ such that either UP(Γ, α) ⊨ Γ or Γ,α ⊢^u □
while Γ,α ⊢^u □ and α ≠ ⟨⟩ do
    α,C ← Handle-Conflict(Γ, α)
    Γ ← Γ ∪ {C}
end
return Γ,α
```

Unit propagation corresponds to unit resolution, in a context where we are interested in collecting implied restrictions on truth assignments rather than derived unit clauses. In particular, the restriction sequence $\mathrm{UP}(\Gamma, \delta)$ makes a clause of Γ false if and only if $\Gamma,\delta \stackrel{u}{\vdash} \square$.

A second process, closely related to unit propagation, involves derivation of clauses called "asserting clauses". CDCL-R proofs of unsatisfiability are constructed by a sequence of asserting clause derivations.

Definition 4 (Asserting Clause; Conflict Clause)**.** *Clause C is an asserting clause for signed CNF formula Γ and restriction sequence δ iff*

1. $\Gamma,\widetilde{C} \stackrel{u}{\vdash} \square$ *if* Γ *is* MV-CNF; $\Gamma, A \stackrel{u}{\vdash} \square$ for each $A \in \widehat{C}$ if Γ is Reg-CNF;
2. For each literal $l \in C$, there is a literal l' with $l' \models \bar{l}$ and $\Gamma,\delta \stackrel{u}{\vdash} l'$;
3. For exactly one literal $l \in C$, there is no literal l' with $l' \models \bar{l}$ and $\Gamma,\delta^- \stackrel{u}{\vdash} l'$.

C is a conflict clause for Γ and δ if it satisfies conditions 1 and 2.

The Probe-and-Learn algorithm is presented in Algorithm 1. It is parameterized by T and $\prec$ so we don't need to present distinct versions for MV-CNF and Reg-CNF. The differences only affect low level details involving operations on literals. In analyses, we take the parameter $\langle T, \prec \rangle$ to be fixed, and allow the other two arguments, the clause set Γ and restriction sequence δ, to vary.

Probe-and-Learn extends δ to a minimal extension α of δ for which unit propagation either produces a satisfying assignment or makes a clause false. In the former case, Γ and the satisfying assignment α are returned. In the latter case Handle-Conflict is executed. Handle-conflict returns a proper prefix of α (which becomes the new value of α) and a clause C which is added to Γ ("learned").

We require Handle-Conflict to satisfy the following correctness property. If the call Handle-Conflict(Γ, α) returns α',C then

1. α' is a proper prefix of α;
2. If α' is empty, then C is $\square$;
3. If α' is not empty, then C is an asserting clause for Γ and UP(Γ,α').

If C is an asserting clause for Γ and α', then UP($\Gamma \cup \{C\}$),α') will be a proper extension of UP(Γ,α), and it is possible for this propagation to reach a new conflict, after which a new asserting clause may be derived. The loop on lines 2-5 of Probe-and-Learn repeats this process until unit propagation no longer produces a conflict (or Γ is proven unsatisfiable). Probe-and-Learn returns the resulting clause set and restriction sequence.

For correctness, there is no restriction on the method by which Handle-Conflict generates C and α. For the p-simulation results of Section 6.1, Handle-Conflict must run in polynomial time. For the concrete simulation bounds of Section 5 and Section 6, it must run in time $O(ns)$, where n is the number of atoms and s is the total number of literal occurrences in the clause set, and there must be a corresponding (possibly implicit) resolution derivation of each asserting clause.

For simplicity of the remainder of the presentation, we will assume that Handle-Conflict is implemented by an algorithm analogous to the standard method that forms the basis of conflict clause derivation in almost all CDCL SAT solvers. We now describe this algorithm, and show that it does run in the require time bound and construct an appropriate resolution derivation.

4.1 Asserting Clause Derivation

An execution of Handle-Conflict(Γ,α) must, except in the case it finds a satisfying assignment, return an asserting clause for a proper prefix of α. The standard methods for this in CDCL SAT solvers involve a resolution derivation closely connected to the unit propagation sequence that establishes a conflict. (The method may either be implemented based on resolution, as we describe below, or the "implication graph" [15].) A generalized version of this process can be used in our many-valued Handle-Conflict procedure. We describe a particular version, the so-called "1UIP asserting clause" derivation. Most CDCL SAT solvers use a refined version of this.

We make the assumption, consistent with standard practice, that UP(Γ, δ) is computed incrementally according to the order of literals in δ. That is, if $\delta = l_1, l_2, l_3...$, we first extend δ by computing UP(Γ,l_1), then extend UP(Γ,l_1) by any additional literals in UP(Γ,$\langle l_1, l_2 \rangle$), etc., so that the last literals in UP(Γ,δ) are those that are not also in UP(Γ,δ^-). The elements of δ in UP(Γ,δ) are called "decisions" (they are the guesses), and the others are there because they are implied.

We obtain the desired asserting clause by means of a resolution derivation constructed as follows. Let $L = l_1, l_2, \ldots, l_d, \ldots, l_k$ be the literals of UP(Γ,δ) in order, and l_d be the last literal of L that is in δ (i.e., the last decision literal in L). We associate to each literal l_i in $l_{d+1}, \ldots, l_r$ a pair of clauses B_i and a C_i. For each i in $d+1, \ldots, r$, let B_i be a clause of Γ that, when restricted by $l_1, \ldots l_{i-1}$, is the unit clause (l_i). Such a B_i must exist, since l_i was obtained by unit propagation.

We define the C_i by induction in reverse order as follows. Let C_{r+1} be a clause of Γ that is made false by L. For each i in $r \ldots d+1$ (proceeding in that order) let C_i be the resolvent of B_i and C_{i+1} if they are resolvable, and C_{i+1} if they are not. It is clear that each clause C_i in the sequence can be derived from Γ by resolution, and that the number of derived clauses in this derivation is at most the length of the sequence L. Let j be the largest index in $d+1, \ldots r$ for which C_j contains only one literal that is inconsistent with a literal in the sequence $l_j, \ldots l_r$. (Certainly $j = d+1$ will work, if no larger value does.) The clause C_j, which is known in the SAT literature as the 1UIP clause (for "first unique implication point"), is the clause to be returned by Handle-Conflict. The restriction sequence to be returned by Handle-Conflict is the least prefix δ' of δ such that UP(Γ,δ') makes C_j a unit clause.

Lemma 2. *Each derived clause in the derivation of the 1UIP clause is a conflict clause for δ and Γ, and the clause returned by HandleConflict is an asserting clause for δ and Γ.*

Proof. We give a proof for MV-RES. The proof for Reg-RES is almost identical. C_{r+1} satisfies property 1 of Definition 4, because $C_{r+1} \in \Gamma$ and trivially for any C we have $\{C\}$,$\widetilde{C} \vdash^u \square$, so Γ,$\widetilde{C_{r+1}} \vdash^u \square$. Also, by choice C_{r+1} is made false by UP(Γ,δ), so it satisfies property 2. So C_{r+1} is a conflict clause. Now, assume that some C_{i+1} is a conflict clause. $\delta, \widetilde{C_i} \vdash^u \square$ because C_i contains every literal of $B_i \cup C_{i+1}$ except for possibly l_i and some literal l that clashes with it. So either $\widetilde{C_i}$ makes C_{i+1} false or it makes C_{i+1} and B_i clashing unit clauses. UP(Γ,δ) makes C_i false because each of its literals is either in C_{i+1} or in $B_i \backslash l_i$, both of which are made false by UP(Γ,δ). Thus, all the C_i are conflict clauses. C_j satisfies property 3 of Definition 4 by choice, so is an asserting clause. $\square$

4.2 Complexity of Probe-and-Learn

In all complexity analyses, given a formula Γ, n will be the number of distinct atoms, the size s the number of literal occurrences, and $|\Gamma|$ the number of clauses.

Lemma 3. *Let Γ be a formula of size s over n atoms. If δ is a restriction sequence for Γ s.t. Γ,$\delta \vdash^u \square$, and* Probe-and-Learn(Γ, δ) *returns Γ', δ', then*

1. *The number of conflicts generated during execution is* $|\Gamma'| - |\Gamma| < |\delta|$*;*
2. *The number of clauses derived in conflict clause derivation is at most* $|\delta|O(n)$*;*
3. *The execution can be carried out in time* $|\delta|O(ns)$*.*

Proof. Each iteration of the body of the loop performs unit propagation and executes Handle-Conflict, which performs the asserting clause derivation. On each iteration of the loop, except possibly the terminating iteration in the case that a satisfying assignment is found, δ is set to a proper prefix of its previous value. Therefore, the number of iterations, the number of conflicts, and the number of asserting clauses added to Γ, are all at most $|\delta|$. The number of derivation steps during one asserting clause derivation is at most the maximum number of literals in a restriction sequence, which is $|T|n = O(n)$. Each resolution derivation step can be carried out in time $O(n)$. The time spent by Handle-Conflict is the time to do one unit propagation and up to n resolution steps, so is $O(s + n^2) = O(ns)$. So the total time for Probe-and-Learn is $|\delta|O(ns) = O(n^2 s)$. □

5 Simulating Resolution with Probe-and-Learn

We now show that, if Π is a resolution refutation of MV-CNF or Reg-CNF formula Γ, there is a sequence of calls to Probe-and-Learn that refutes Γ in time polynomial in the combined size of Π and Γ. We begin by showing that any empowering clause can be absorbed by a sequence of calls to Probe-and-Learn. Througout, n is the number of distinct atoms and s the number of literal occurrences, of formula Γ.

Lemma 4. *Suppose* $C = (A \vee l)$ *is* l*-empowering for clause set* Γ*. Then there is a sequence of calls to Probe-and-Learn that generates a superset* Γ' *of* Γ *such that* C *is not* l*-empowering for* Γ'*, and the following properties hold.*

1. *The number of calls to Probe-and-Learn is* $O(n)$*;*
2. *The size of the underlying resolution derivation is* $O(n^2)$
3. *The entire computation can be carried out in time* $O(n^2 s)$*;*

Proof. We state the proof for Reg-RES; that for MV-RES is similar. Let δ be a restriction sequence consisting of the literals of some element of $\widehat{A}$, in any order, followed by a literal that is inconsistent with l. We construct a sequence $\Gamma_0, \Gamma_1, \ldots \Gamma_r$ of increasing (by set inclusion) clause sets with $\Gamma = \Gamma_0$, such that C is not l-empowering for Γ_r, and set $\Gamma' = \Gamma_r$. We obtain Γ_{i+1} by setting $\Gamma_{i+1},\alpha = \text{Probe-and-Learn}(\Gamma_i, \delta)$,

and ignoring α. Each execution of Probe-and-Learn involves one or more executions of Handle-Conflict(Γ_i,δ'), where δ' is a prefix (not necessarily proper) of δ. We consider the sequence of all calls to Handle-Conflict, over however many calls to Probe-and-Learn are made. Each execution of Handle-Conflict returns a pair $\langle \alpha, B \rangle$, where α is a proper prefix of δ and B is an asserting clause for UP(Γ_i,α) and Γ_i. The "learned clause" B is added to Γ_i. Clause B, when restricted by UP(Γ_i, δ^-) is a unit clause. By construction, for each call to Handle-Conflict this implied unit clause will contain a distinct literal on the same atom as l. Therefore, after $O(n)$ calls to Handle-Conflict, Γ_i must contain either $\Box$ a clause B for which the implied unit clause entails l. In either case, C is not l-empowering for Γ_i. Each call to Handle-Conflict requires time at most $O(ns)$, and derives $O(n)$ clauses. □

Lemma 5. *If C is empowering for Γ, and C is of width (size) w, then there is a sequence of calls to Probe-and-Learn that generates a superset Γ' of Γ which absorbs C, and such that the following hold.*

1. *The number of calls to Probe-and-Learn is $O(wn)$;*
2. *The size of the underlying resolution derivation is $O(wn^2)$;*
3. *The entire computation can be executed in time $O(wn^2s)$.*

Proof. Apply Lemma 4 for each literal $l \in C$. □

To see that an appropriate sequence of calls to Probe-and-Learn can refute Γ in a number of steps not much greater than the size of any given refutation, we identify a sequence of empowering clauses, and absorb each. By "refutes Γ", we mean that it produces a set Γ' of clauses, each of which is implied by Γ, and containing $\Box$.

Lemma 6. *Let Γ be a set of signed clauses and Π a signed resolution refutation of Γ of size at most r clauses, in which no clause has width greater than w. Then there is a sequence of calls to Probe-and-Learn that refutes Γ, such that the following hold.*

1. *The number of calls to Probe-and-Learn in the is $O(wnr)$;*
2. *The size of the underlying resolution refutation is $O(wn^2r)$;*
3. *The entire computation can be executed in time $O(wn^2sr)$.*

Proof. We generate a sequence $\Gamma_0 \ldots \Gamma_k$ of supersets of Γ, where $\Gamma_0 = \Gamma$, $k \leq r$, and $\Box \in \Gamma_k$, as follows. If $\Gamma_i \stackrel{u}{\vdash} \Box$ we are done. Otherwise, let C be the first clause in Π that is empowering for Γ_i. Lemma 1 ensures such a clause exists. By Lemma 5 there is a sequence of calls to Probe-and-Learn that generates a superset of Γ_i for which C is absorbed. Let this set be Γ_{i+1}. The claims follow from Lemma 5 and the fact that the number of clauses to be absorbed is at most r. □

Algorithm 2: Multi-Valued CDCL with Restarts (MV-CDCL-R)

```
Input: Finite set Φ of signed clauses; truth value set ⟨T, ≺⟩
. Output: SAT or UNSAT
. Γ ← Φ   // Clause set, initialized to the input clauses
δ ← ⟨⟩   // Restriction sequence, initialized to empty
repeat
    Γ, δ ← Probe-and-Learn(Γ, δ, ⟨T, ≺⟩)
    if UP(Γ, δ) ⊨ Γ then
        return SAT
    if δ = ⟨⟩ then
        return UNSAT
    if Time to Restart then
        δ ← ⟨⟩
end
```

6 CDCL with Restarts

We assume the reader is familiar with the standard CDCL-R algorithm. (A self-contained description of CDCL and its relationship to resolution can be found in [16], among other places. The reader may also want to refer to [18] and [2] for distinct presentations of the algorithm, as well as the proofs we work from, and [3] where a careful examination of the relation between resolution and the implication graph method for obtaining conflict clauses appears.)

CDCL-R can be described in terms of a sequence of calls to Probe-and-Learn, as illustrated in Algorithm 2. While many details have been abstracted away, Algorithm 2 captures the core algorithm implemented by CDCL-R-based solvers. For simplicity, let us assume the given formula is unsatisfiable. The algorithm begins with the empty restriction sequence. In the first call to Probe-and-Learn, the restriction sequence is extended until a clause is made false, after which clause learning and back-jumping are carried out (by Handle-Conflict, within Probe-and-Learn). In subsequent executions of the loop body, the restriction sequence resulting from the most recent Handle-Conflict is extended until Probe-and-Learn again finds a conflict. Each asserting clause derived by Handle-Conflict is new, so each call to Probe-and-Learn extends the clause set with at least one new implied clause. This is repeated, until the derived conflict clause is the empty clause.

At this level of abstraction, the signed versions and classical version are not

distinguishable, except for the input parameter $\langle T, \prec \rangle$, which affects only the low-level steps in Probe-and-Learn and Handle-Conflict. We make $\langle T, \prec \rangle$ a parameter to make explicit the fact that Probe-and-Learn (and in particular Handle-Conflict), must be appropriate to the order relation and the class of formulas in question. If T is of size 2, then with appropriate choice for Probe-and-Learn, this algorithm is equivalent to the classical CDCL-R.

Since CDCL-R can be viewed simply as a repeated application of Probe-and-Learn, it is straightforward to see that MV-CDCL-R can be guided to refute any formula that has a resolution refutation Π in time polynomial in the size of Π. We need one more kind of object to complete the argument.

Definition 5 (Extended Restriction Sequence for MV-CDCL-R)**.** *An extended restriction sequence for MV-CDCL-R on input* Γ *and* $\langle T, \prec \rangle$ *is a finite sequence of symbols satisfying:*

1. *Each symbol is either a literal for* $\langle T, \prec \rangle$ *or the distinguished symbol* `R`*;*

2. *Each maximal sub-sequence with no* `R` *is a restriction sequence for* Γ*.*

We may take two views of an extended restriction sequence. On one view, we may take it as a record or witness of an actual execution of MV-CDCL-R. On the other, we may view it as a string to control an intended execution of MV-CDCL-R.

Lemma 7. *Let* Γ *be an MV-CNF or Reg-CNF formula of size* s *over* n *atoms that has a resolution refutation of size* r *and width* w*. Then there is an execution of MV-CDCL-R that refutes* Γ *in time* $O(wn^2sr)$ *and with an underlying resolution refutation of size* $O(wn^2r)$*.*

Proof. It is sufficient to show there is an extended restriction sequence that produces such an execution. Since MV-CDCL-R is repeated execution of Probe-and-Learn, we need only to take the sequence of calls to Probe-and-Learn implied by Lemma 6, produce a restriction sequence corresponding to each call, and concatenate all the restriction sequences with an `R` separating each adjacent pair. □

6.1 Proof Complexity and p-Simulation

Propositional proof complexity is the study of the relative power of proof systems for propositional logic, measured by minimum length of proofs for tautological formulas. The abstract definition of propositional proof system introduced in the seminal paper of Cook and Reckow [8] can be trivially adapted to refutation proofs for unsatisfiable signed CNF formulas (or indeed any co-NP complete set).

Definition 6. *A refutation proof system for a set S of unsatisfiable signed CNF formulas is a set of strings L with a poly-time onto function V_L from strings over the alphabet of L to $S \cup \bot$, such that $V_L(x) = \Gamma$ if x is an L-proof that Γ is unsatisfiable, and $V_L(x) = \bot$ otherwise.*

Intuitively, L is the proofs of the system and V_L is an efficiently computable function that verifies their correctness.

Proof system $\mathcal{A}$ p-simulates proof system $\mathcal{B}$ if there exists a polynomial function $p()$ such that, for every unsatisfiable formula Γ and every $\mathcal{B}$-proof Π_B of Γ, there is an $\mathcal{A}$-proof Π_A of Γ with $|\Pi_A| \leq p(|\Pi_B|)$.

As a simplifying convention, we require that the minimum size of a proof of Γ is $|\Gamma|$. This is not standard in the proof complexity literature, but is necessary for relevance to practical satisfiability algorithms, and is followed also in, e.g., [3, 7, 11, 18]. This is because a formula may be large but have a tiny proof, and any reasonable satisfiability solver begins by reading the entire formula. Moreover, any reasonable CDCL-R-based solver begins by executing unit propagation, which may visit every clause of the formula.

To view a satisfiability algorithm as a proof system, we may take any trace of the algorithm on an unsatisfiable clause set Γ as a proof of the unsatisfiability of Γ, provided that the trace reflects the running time of the algorithm, and that we can efficiently verify that the trace corresponds to an execution of the algorithm that reports "unsatisfiable". For present purposes, we may use extended restriction sequences as MV-CDCL-R proofs.

Theorem 1. *MV-CDCL-R p-simulates MV-RES and Reg-RES.*

Proof. To show that CDCL-R p-simulates resolution, it is sufficient to show that for any resolution refutation Π of clause set Γ there is an extended restriction sequence δ such that, when CDCL-R is executed in accordance with δ on input Γ, it runs in time polynomial in the length of Π, and reports UNSAT. This follows from Lemma 7. □

Corollary 1 (Pipatsrisawat & Darwiche)**.** *CDCL-R p-simulates resolution.*

To see that this is indeed a corollary, it is enough to observe that MV-CNF and MV-RES for $|T| = 2$ are equivalent to the classical case.

Theorem 2. *MV-RES and Reg-RES p-simulate MV-CDCL-R for MV-CNF and Reg-CNF formulas respectively.*

Proof. Consider an execution of MV-CDCL-R that halts and outputs "UNSAT", and let σ be the extended restriction sequence corresponding to this execution. The implicit underlying resolution refutation of Γ consists of the sequence of asserting

clauses returned by Handle-Conflict, and all the clauses implicitly used in their derivation. It is clear that this sequence of clauses is of length polynomial in the length of σ. □

7 Application Issues

Our stated motivation was to better understand the question of whether reasoning in multi-valued logics might be effectively carried out by reduction to multi-valued CNF formulas followed by execution of a CDCL-R based algorithm adapted to the multi-valued setting. Our simulation results provide one way of understanding the power of MV-CDCL-R. To the extent that the theorems of [18, 2] are related to positive performance characteristics of solvers based on CDCL-R, we now expect the same to hold for solvers based on MV-CDCL-R. This provides some evidence that reduction to multi-valued CNF may be a fruitful approach to multi-valued logic reasoning.

The MV-CDCL-R algorithm seems quite reasonable to implement. Most of the work is easily inherited from existing CDCL SAT solvers. For $|T|$ up to 64, the sign of a literal can be implemented in a single word on modern CPUs, and key operations such as the check for clashing, computing the residual of two clashing literals, or merging two non-clashing literals, can be done in a single CPU instruction. Even for significantly larger T, most modern CPUs provide support to do these operations in hardware.

A number of algorithms and solvers for signed or multi-valued formulas, or special cases of them, have been described in the literature. Most of these are essentially backtracking or tableau-based. Thus, they correspond to tree-like versions of resolution and we can expect that, as in the classical case, there will be formulas for which they are exponentially less efficient than unrestricted resolution. We are aware of two algorithms that seem closely related to our MV-CDCL-R, namely those of of [14] and [12]. These do not use restarts, which are essential to our work here, but otherwise seem very closely related to the instantiation of MV-CDCL-R for MV-CNF. The interested reader will find a useful discussion of implementation issues in both of those reports. In light of the importance of restarts in high-performance SAT solvers, and of our results here, it would be very interesting to see the effect on performance of modifying these solvers to execute good restart policies.

An alternate reasoning strategy is to reduce the multi-valued reasoning problem to SAT, and then execute a classical (CDCL-R based) SAT solver. The advantage of this approach is that SAT solvers are easily available and subject to constant improvement efforts. We will not address reductions directly from multi-valued

logics to SAT, but restrict our attention to the following observation. We can easily reduce Signed CNF satisfiability to SAT as follows. The set of propositional atoms will contain atoms we write as $p = a$, one for each signed atom p and each truth value $a \in T$. Each literal $p{\sqsubseteq}\{a, b, c, \ldots\}$ then maps to the disjunction $(p = a \vee p = b \vee p = c \vee \ldots)$. Each clause of a Signed CNF formula maps to a single propositional clause. We add to the resulting clauses the set of binary clauses of the form $(\overline{p = a} \vee \overline{p = c})$, for each atom p and each pair $a, c \in T$ with $a \neq c$.

We may simulate Signed resolution using classical resolution on these clauses, but this does not seem of potential use in practice. To simulate the single resolution step

$$\frac{p{\sqsubseteq}S \ \vee \ A \qquad p{\sqsubseteq}R \ \vee \ B}{p{\sqsubseteq}(S \cap R) \ \vee \ A \ \vee \ B} \tag{3}$$

with classical resolution seems to require a number of 2-valued resolution steps that is quadratic in $|T|$. Since T is fixed, this is only a constant blow-up, but constants can be important in solver design. Moreover, this constant applies to optimal proofs, adding a large amount of non-determinism to the proof simulations. In a practical algorithm, expecting to make $|T|^2$ correct decisions in order to simulate a single multi-valued resolution step seems unreasonably optimistic.

Recently, it was been shown that for any CDCL-R solver S and any unsatisfiable CNF formula F, it is easy to generate a CDCL solver S' (with no restarts) and a formula F' (consisting of a conjunction of F with some new clauses) such that S' generates exactly the same resolution refutation of F that S' does, and with only a small polynomial slow-down [4]. Thus, theoretically, restarts are not really required. However, it is far from clear that this fact can be used to make implemented solvers without restarts that are as fast in practice as solvers that use restarts.

The potential of using resolution-based methods for signed formulas to solve combinatorial optimization problems has been examined in [1]. It would be interesting to relate that direction of work to the present one.

8 Conclusion

We have presented a natural generalization of the SAT algorithm known as CDCL with restarts to signed CNF formulas, in particular to Multi-valued CNF formulas, and to Regular formulas when the truth value set is a lattice. Adapting the proofs from [18, 2] we showed that the algorithm p-simulates natural forms of binary resolution for these formulas, and vice versa. The simulation of resolution by the algorithm is quite efficient, both in terms of length and width. Explaining the impressive performance of SAT solvers in practice in light of their worst-case per-

formance and the NP-completeness of SAT is an area of active interest. We do not know if the theorems of [18, 2] are significant in such an explanation, but it is plausible and consistent with empirical observation. Moreover, there is a good deal of evidence that frequent restarts are important in practice. To the extent that this is so, we have shown that similar properties hold for the generalization to multi-valued CNF formulas. We consider the algorithm we have described easily implementable and of potential use in developing practical model finders and theorem provers for multi-valued logics.

References

[1] Carlos Ansótegui, María Luisa Bonet, Jordi Levy, and Felip Manyà. Resolution procedures for multiple-valued optimization. *Information Sciences*, 227:43 – 59, 2013.

[2] Albert Atserias, Johannes Klaus Fichte, and Marc Thurley. Clause-learning algorithms with many restarts and bounded-width resolution. *J. Artif. Intell. Res. (JAIR)*, 40:353–373, 2011.

[3] Paul Beame, Henry Kautz, and Ashish Sabharwal. Towards understanding and harnessing the potential of clause learning. *Journal of Artificial Intelligence Research*, 22:319–351, 2004.

[4] Paul Beame and Ashish Sabharwal. Non-restarting SAT solvers with simple preprocessing can efficiently simulate resolution. In Carla E. Brodley and Peter Stone, editors, *Proceedings of the Twenty-Eighth AAAI Conference on Artificial Intelligence, July 27 -31, 2014, Québec City, Québec, Canada.*, pages 2608–2615. AAAI Press, 2014.

[5] Bernhard Beckert, Reiner Hähnle, and Felip Manyà. The 2-sat problem of regular signed cnf formulas. In *Proc. 30th IEEE International Symposium on Multi-Valued Logic (ISMVL 2000)*, pages 331–336, 2000.

[6] Bernhard Beckert, Reiner Hahnle, and Filip Manya. The SAT problem of signed CNF formulas. In Basin, D'Agostino, Gabbay, Matthews, and Viganò, editors, *Labelled Deduction*, volume 17 of *Applied Logic Series*, chapter 3, pages 59–80. Kluwer Academic, 2000.

[7] Samuel Buss, Jan Hoffmann, and Jan Johannsen. Resolution trees with lemmas: Resolution refinements that characterize DLL algorithms with clause learning. *Logical Methods in Computer Science*, 4(4:13), 2008.

[8] Stephen Cook and Robert Reckhow. The relative efficiency of propositional proof systems. *J. Symbolic Logic*, 44(1):23–46, 1979.

[9] Reiner Hähnle. Short conjunctive normal forms in finitely valued logics. *J. Log. Comput.*, 4(6):905–927, 1994.

[10] Reiner Hähnle. Exploiting data dependencies in many-valued logics. *Journal of Applied Non-Classical Logics*, 6(1):49–69, 1996.

[11] Philipp Hertel, Fahiem Bacchus, Toniann Pitassi, and Allen Van Gelder. Clause learning can effectively p-simulate general propositional resolution. In *Proc., 23rd National*

Conference on Artificial Intelligence (AAAI-08), Volume 1, pages 283–290, 2008.

[12] Siddhartha Jain, Eoin O'Mahony, and Meinolf Sellmann. A complete multi-valued SAT solver. In *Principles and Practice of Constraint Programming – 16th International Conference (CP 2010), St. Andrews, Scotland, UK, September 6-10, 2010. Proceedings*, pages 281–296, 2010.

[13] Henry Kautz, Eric Horvitz, Yongshao Ruan, Carla Gomes, and Bart Selman. Dynamic restart policies. In *Proc., 19th National Conference on Artificial Intelligence (AAAI-2002)*, pages 674–681, 2002.

[14] Cong Liu, Andreas Kuehlmann, and Matthew W. Moskewicz. Cama: A multi-valued satisfiability solver. In *Proc., 2003 Int'l. Conf. on Computer Aided Design (ICCAD-2003)*, pages 326–333, 2003.

[15] João Marques-Silva and Karem Sakallah. Grasp: A search algorithm for propositional satisfiability. *IEEE Transactions on Computers*, 48(5):506–521, May 1999.

[16] David Mitchell. A SAT solver primer. *EATCS Bulletin*, 85:112–133, February 2005. The Logic in Computer Science Column.

[17] David Mitchell. Resolution and clause learning for multi-valued CNF formulas. In Thomas Lukasiewicz, Rafael Peñaloza, and Anni-Yasmin Turhan, editors, *Proceedings of the First Workshop on Logics for Reasoning about Preferences, Uncertainty, and Vagueness, PRUV 2014, co-located with 7th International Joint Conference on Automated Reasoning (IJCAR 2014), Vienna, Austria, July 23-24, 2014.*, volume 1205 of *CEUR Workshop Proceedings*, pages 141–154. CEUR-WS.org, 2014.

[18] Knot Pipatsrisawat and Adnan Darwiche. On the power of clause-learning SAT solvers as resolution engines. *Artificial Intelligence*, 175(2):512–525, 2011.

[19] Hantao Zhang and Mark E. Stickel. An efficient algorithm for unit propagation. In *Proceedings of the Fourth Int'l Symposium on Artificial Intelligence and Mathematis (AI-MATH'96), Fort Lauderdale Florida USA*, pages 166–169, 1996.

A Linear-time Unit Propagation

We will show that unit propagation can be executed in linear time on MV-CNF and REG-CNF formulas. The method uses a single watched literal in each clause, which is a simplification of the method with head and tail watches of [19].

Lemma 8. *Let Γ be a set of multi-valued clauses with s literals in total, over a domain T of size t. Then* $\mathrm{UP}(\Gamma)$*, or* $\mathrm{UP}(\Gamma,\delta)$*, can be computed in time $O(st)$.*

Proof. We assume a data structure for clauses that supports viewing each clause as a list of literals, and each literal $p{\sqsubseteq}S$ in a clause as a list of atom-value pairs $\langle p, a\rangle$, one for each $a \in S$, in no particular order. (This exact representation is not important, but makes explanation simple.) We further assume that accessing the next literal in a clause or the next value in S takes constant time. For each clause

C we maintain a pointer or index to a single pair $\langle p, a\rangle$ for a single literal $p{\sqsubseteq}S$, and call this the "watched value" for C. Initially, the watched value for each clause is the first value of the first literal. As we execute the procedure, we keep track of a set of assignments of values to each atom that remain possible. We will maintain the following invariant:

> If $\langle p, a\rangle$ in the representation of literal $p{\sqsubseteq}S$ is the watched value for clause C, then every $\langle p, b\rangle$ that precedes $\langle p, a\rangle$ in the representation of $p{\sqsubseteq}S$ is known to be impossible, and for every literal that precedes $p{\sqsubseteq}S$ in the representation of C, every value is known to be impossible.

We maintain a queue Q of pairs $\langle p, a\rangle$ of atoms and values which we have determined to be impossible, but for which we have not propagated the effects of that fact. We initialize Q by inserting the set of pairs

$$\{\langle p, a\rangle \mid C = (p{\sqsubseteq}S) \text{ is a unit clause of } \Gamma \text{ and } a \notin S\}$$

For each pair $\langle p, a\rangle$ consisting of an atom p and value a, we construct a list of watched occurrences of the pair. Now, until Q is empty, remove one pair $\langle p, a\rangle$ from Q and handle it as follows. Traverse the list of watched occurrences of $\langle p, a\rangle$. For each occurrence, scan the remaining values in the literal with $\langle p, a\rangle$ looking for a value that is not known to be impossible, to be used as a new watch. If none is found, search for one in the subsequent literals of the clause. If none is found, this clause is now effectively an empty clause, so we are done with it and proceed to the next watched occurrence of $\langle p, a\rangle$. If one was found but it is in the last literal of the clause, then this clause is effectively unit, say $(q{\sqsubseteq}R)$. Add the set of pairs $\{\langle q, b\rangle \mid b \notin R$ and b is not already known to be impossible$\}$ to Q. If a new value to watch was found and it is not in the last literal of the clause, then add the corresponding pair to the appropriate watch list, and get the next pair from Q.

This algorithm visits each literal l in Γ at most once, and uses constant work for each such visit, so runs in time $O(st)$. □

Since t is a constant in our analyses, this establishes the linear time unit propagation we need. This algorithm works for both MV-CNF and REG-CNF, and essentially amounts to reducing them to the two-valued case and performing standard watched-literal unit propagation.

Received 27 July 2015