MaxSAT resolution for regular propositional logic

Jordi Coll, Chu-Min Li, Felip Manyà, Elifnaz Yangin

International Journal of Approximate Reasoning 162 (2023) 109010

Contents lists available at ScienceDirect

International Journal of Approximate Reasoning

journal homepage: www.elsevier.com/locate/ijar

MaxSAT resolution for regular propositional logic

Jordi Coll [a], Chu-Min Li [b,c], Felip Manyà [a,*], Elifnaz Yangin [a,*]

[a] Artificial Intelligence Research Institute (IIIA, CSIC), Bellaterra, Spain
[b] MIS, Université de Picardie Jules Verne, Amiens, France
[c] Aix Marseille Univ, Université de Toulon, CNRS, LIS, Marseille, France

ARTICLE INFO

Article history:
Received 18 March 2023
Received in revised form 10 July 2023
Accepted 3 August 2023
Available online 9 August 2023

Keywords:
Multiple-valued logic
Maximum satisfiability
Signed CNF formulas
Regular CNF formulas
Resolution
Variable elimination

ABSTRACT

Proof systems for SAT are unsound for MaxSAT because they preserve satisfiability but fail to preserve the minimum number of unsatisfied clauses. Consequently, there has been a need to define cost-preserving resolution-style proof systems for MaxSAT. In this paper, we present the first MaxSAT resolution proof system specifically defined for regular propositional clausal forms and prove its soundness and completeness. The defined proof system provides an exact approach to solving Regular MaxSAT and Weighted Regular MaxSAT with variable elimination algorithms.

1. Introduction

Signed propositional logic is a logical formalism for knowledge representation that lies in the intersection of the areas of constraint programming, many-valued logics and annotated logic programming. Signed conjunctive normal forms (signed CNF formulas) can be seen as classical conjunctive normal forms equipped with a generalized notion of literal, called signed literal. A signed literal is an expression of the form $S : x$, where x is a propositional variable and S, its sign, is a subset of a domain N.

An important and well-investigated subclass of signed CNF formulas are regular CNF formulas: If N is equipped with an order $\leq$, a sign is regular if it is either of the form $\{j \in N \mid j \geq i\}$ or $\{j \in N \mid j \leq i\}$ for some $i \in N$. Regular CNF formulas are the signed CNF formulas whose literals have regular signs. Regular CNF formulas, and by extension signed CNF formulas, can be used to encode Boolean CNF formulas: If $N = \{0, 1\}$, a positive Boolean literal x is encoded as $x \geq 1$, and a negative literal $\neg x$ is encoded as $x \leq 0$. As in the Boolean case, the problem of determining the satisfiability of signed and regular CNF formulas is NP-complete [12,33].

However, regular CNF formulas offer distinct advantages over signed CNF formulas, including the distinction between positive and negative literals or the possibility of dealing with infinite domains.

The informal meaning of a signed literal $S : x$ is "x is constrained to the values of S". Thus, signed CNF formulas provide a suitable knowledge representation language for constraint programming, which has already been shown to be competitive when solving combinatorial problems with signed encodings [13,14]. From a problem solving perspective, the main advan-

* Corresponding authors.
E-mail addresses: felip@iiia.csic.es (F. Manyà), elifnaz.yangin@iiia.csic.es (E. Yangin).

https://doi.org/10.1016/j.ijar.2023.109010

tage is that the powerful and high-performing technology developed by the satisfiability testing community can be *easily* adapted to signed and regular CNF formulas. Compared to Boolean satisfiability testing, where problem-specific multiple-valued variables need to be encoded employing a set of Boolean variables, the domain information becomes explicit in a signed literal and can be exploited, for instance, to define domain-dependent variable selection heuristics [5,6].

When N is considered as a truth value set, signed and regular logics provide a framework for automated reasoning in multiple-valued logics [21,22]. The satisfiability problem (SAT) of any finitely-valued propositional logic, as well as of certain infinitely-valued logics, is polynomially reducible to the satisfiability problem of signed CNF formulas (Signed SAT) and regular CNF formulas (Regular SAT) [12].

We obtain annotated logic programming when we consider regular Horn CNF formulas and N is a lattice [25]. Annotated logic programming is an appropriate logical formalism to manage locally inconsistent, incomplete and uncertain databases.

In the area of satisfiability testing, one of the problems that has attracted more interest in recent years is the Maximum Satisfiability problem (MaxSAT) [9,28]. Whereas SAT is the problem of deciding if there exists a truth assignment for a given CNF formula that evaluates the formula to true, MaxSAT is the problem of finding a truth assignment that minimizes the number of unsatisfied clauses in a CNF formula.

In practice, SAT is used as a generic problem solving formalism for decision problems and MaxSAT for optimization problems. The development of highly competitive MaxSAT solvers (e.g. [9,32]) has enabled the application of MaxSAT to solve challenging optimization problems in various domains such as bioinformatics [35], circuit design and debugging [36], combinatorial testing [8], diagnosis [19], planning [38], scheduling [15] and team formation [34].

This paper specifically focuses on the MaxSAT problem for Regular CNF formulas. Our aim is to define a complete resolution-style proof system for Regular MaxSAT. Our work is motivated by the fact that the logic machinery defined for Regular SAT is not valid for Regular MaxSAT. The inference rules for Regular SAT are unsound in Regular MaxSAT because they preserve satisfiability but fail to preserve the minimum number of unsatisfied clauses between the premises and the conclusions.

In the Boolean case, new complete resolution and tableau-style proof systems for Boolean MaxSAT have had to be defined (see e.g. [18,20,26,31]). Presently, restrictions of Boolean MaxSAT resolution are routinely used in branch-and-bound MaxSAT solvers such as ahmaxsat [1] and MaxSatz [29,30]. Furthermore, Boolean MaxSAT resolution has been extended to signed CNF formulas [3]. The defined signed MaxSAT resolution rules are complete and provide a logical framework for weighted constraint satisfaction problems (WCSP), where some restrictions of the rules enforce existing local consistency properties for WCSPs [3,4]. Nevertheless, the restriction to Regular CNF formulas has not been investigated so far. This paper aims to bridge this gap.

It is widely known that there exists no polynomial-size resolution proof of the pigeon hole principle (PHP) [23]. Nevertheless, there exist polynomial-size MaxSAT resolution proofs of PHP if PHP is encoded as a MaxSAT instance using the dual rail encoding [24]. Indeed, the combination of the dual rail encoding and MaxSAT resolution is a stronger proof system than either general resolution or conflict-driven clause learning [16]. More recently, it has been shown that MaxSAT resolution, when combined with certain rules, also produces polynomial-size MaxSAT resolution proofs of PHP. For example, MaxSAT resolution with the split rule (replace clause c with $\overline{x} \vee c$ and $x \vee c$) produces polynomial-size proofs of PHP, and this does not happen if MaxSAT resolution is replaced with resolution [17,27]. These results suggest that MaxSAT resolution could be a key component in the development of faster and more robust SAT and MaxSAT solvers based on stronger proof systems. In particular, the use of bounded variable elimination, using MaxSAT resolution, as a preprocessing and inprocessing technique in SAT and MaxSAT solvers could significantly enhance the performance of these solvers in the Boolean and many-valued scenarios.

We believe that studying Regular MaxSAT resolution is worthwhile due to the utilization of regular signs. Apart from enabling to deal with infinite truth value sets, regular signs contribute to produce simpler and more efficient variable elimination algorithms for Regular MaxSAT. Thanks to the notion of polarity and the structure of regular signs, Regular MaxSAT resolution proofs are simpler than Signed MaxSAT resolution proofs because the inference rules exploit such features. Additionally, detecting intersections among signs does not require intricate operations; simple comparisons of numbers suffice. Regular signs also simplify the data structures as they can be represented by recording the lowest and greatest values of the sign.

Regular CNF formulas are also significant because they are the language behind the so-called order encoding [7,37], which has demonstrated to be more efficient than other encodings to model, for example, cardinality constraints.

The main contributions of this paper can be summarized as follows:

- Definition of the first resolution-style proof system for Regular MaxSAT and the corresponding proofs of soundness and completeness.
- Description of an exact variable elimination algorithm for Regular MaxSAT that incorporates a notion of saturation that exploits the structure of regular signs.
- Extension of the proposed proof system to handle CNF formulas with weighted clauses, resulting in a sound and complete proof system for Weighted Regular MaxSAT.

The paper is organized as follows. Section 2 defines the logic of signed and regular CNF formulas, as well as the Signed and Regular MaxSAT problems. Section 3 defines a resolution-style proof system for regular MaxSAT and proves its sound-

ness and completeness. Section 4 describes an exact variable elimination algorithm for Regular MaxSAT. Section 5 extends the proposed proof system to Weighted Regular MaxSAT. Finally, Section 6 presents some concluding remarks.

2. Syntax and semantics of signed and regular CNF formulas

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

Definition 1. A *truth value set* is a non-empty, finite set $N = \{i_1, i_2, \ldots, i_n\}$ where $n \in \mathbb{N}$. The cardinality of N is denoted by $|N|$. We assume that a total order $\leq$ is associated with N.

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

Definition 3. 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 $\square$. A signed clause c' *subsumes* a signed clause c iff, for every literal $S' : x \in c'$, there is a literal $S : x \in c$ such that $S' \subseteq S$. A clause c is a *tautology* iff there is a variable x in c such that $\bigcup_{S:x \in c} S = N$. A *signed CNF formula* is a finite multiset of signed clauses.

The clauses of a signed CNF formula are implicitly conjunctively connected and the literals in a signed clause are implicitly disjunctively connected. We often use $S_1 : x_1 \vee \cdots \vee S_k : x_k$ to represent a signed clause of the form $\{S_1 : x_1, \ldots, S_k : x_k\}$.

Example 1. Let $N = \{1, 2, 3\}$. The signed CNF formula

$$\{\{1, 3\} : x_1, \{2\} : x_1 \vee \{2, 3\} : x_2, \{1, 2\} : x_2 \vee \{1, 3\} : x_3\}$$

contains a unit signed clause and two binary signed clauses.

Definition 4. A weighted signed clause is a pair (c, w), where c is a signed clause and w, its weight, is a positive number. A weighted signed CNF formula is a finite multiset of weighted signed clauses.

Example 2. Let $N = \{1, 2, 3\}$. The formula

$$\{(\{1, 3\} : x_1, 8), (\{2\} : x_1 \vee \{2, 3\} : x_2, 3), (\{1, 2\} : x_2 \vee \{1, 3\} : x_3, 5)\}$$

is an example of weighted signed CNF formulas.

Note that weighted signed CNF formulas can contain multiple occurrences of the same signed clause, either with identical or different weights. However, it is possible to unfold a weighted signed clause into several occurrences of the clause with smaller weights, and vice versa. For example, consider a signed clause c. The following three weighted signed CNF formulas are equivalent: $\{(c, 3)\}$, $\{(c, 2), (c, 1)\}$, $\{(c, 1), (c, 1), (c, 1)\}$. In particular, by unfolding all weighted clauses into clauses of weight 1, we obtain an unweighted signed CNF formula.

Definition 5. Given a total order associated with N, for each element i of the truth value set N, let $\geq i$ denote the sign $\{j \in N \mid j \geq i\}$, and let $\leq i$ denote the sign $\{j \in N \mid j \leq i\}$. A sign S is *regular* if it is identical to $\geq i$ or $\leq i$ for some $i \in N$.

Note that we assume that N is totally ordered. Regular CNF formulas also include the case where N is partially ordered [10,11]. In the sequel, we always consider total orders because are the ones that have been used so far in satisfiability testing and constraint programming, which is our area of expertise. For the same reason, we exclusively consider finite truth value sets.

Definition 6. A literal $S : x$ is a *regular literal* if its sign S is regular. A regular literal has *positive polarity* if its sign is of the form $\geq i$ and has *negative polarity* if its sign is of the form $\leq i$. A signed clause (a signed CNF formula) is a *regular clause* (a *regular CNF formula*) if all its literals are regular.

Example 3. Let the truth value set $N = \{1, 2, 3, 4\}$ be ordered with the standard order on natural numbers. Then, the signs $\leq 2 = \{1, 2\}$ and $\geq 3 = \{3, 4\}$ are regular; and $\overline{\leq 2} = \geq 3 = \{3, 4\}$ and $\overline{\geq 3} = \leq 2 = \{1, 2\}$ are its complements, respectively. The signed CNF formula

$$\{\{2,3,4\}:x_1,\{1,2\}:x_1 \vee \{1,2,3\}:x_2,\{4\}:x_2 \vee \{3,4\}:x_3\}$$

is a regular CNF formula, which is represented with regular signs as follows:

$$\{\geq 2:x_1,\leq 2:x_1 \vee \leq 3:x_2,\geq 4:x_2 \vee \geq 3:x_3\}.$$

Next, we define the semantics of signed CNF formulas, and then the Signed MaxSAT problem. The semantics definition of regular CNF formulas is that of signed CNF formulas when their signs are regular.

Definition 7. An *assignment* is a mapping that assigns to every propositional variable an element of the truth value set. An assignment I *satisfies* a signed (regular) literal $S:p$ iff $I(p) \in S$. It *satisfies* a signed (regular) clause c iff it satisfies at least one of the literals in c, and it *satisfies* a signed (regular) CNF formula $\mathcal{C}$ iff it satisfies all the clauses in $\mathcal{C}$. A signed CNF formula is *satisfiable* iff it is satisfied by at least one assignment; otherwise, it is *unsatisfiable*. Signed (Regular) SAT for a signed (regular) CNF formula $\mathcal{C}$ is the problem of deciding if there is an assignment that satisfies $\mathcal{C}$.

By definition, the empty signed clause is unsatisfiable and the empty signed CNF formula is satisfiable. Notice that $S:x$ can be interpreted as "x is constrained to the values in S" and that the semantics is classical above the literal level.

Definition 8. Given a multiset of signed (regular) clauses $\mathcal{C}$, Signed (Regular) MaxSAT is the problem of finding an assignment for $\mathcal{C}$ that minimizes the number of unsatisfied clauses in $\mathcal{C}$. Given a multiset of weighted signed (regular) clauses $\mathcal{C}$, Weighted Signed (Regular) MaxSAT is the problem of finding an assignment for $\mathcal{C}$ that minimizes the sum of weights of unsatisfied clauses in $\mathcal{C}$.

3. Regular MaxSAT resolution

In this section, we define a resolution-style proof system for Regular MaxSAT and prove its soundness and completeness. Our calculus starts with a multiset of regular clauses $\mathcal{C}$, and applies a set of inference rules that remove two clauses (premises) from $\mathcal{C}$ and introduce a number of new clauses (conclusions) to $\mathcal{C}$.

We require all clauses to be in a normalized form before the application of a rule. Assuming that $N = \{1,\ldots,|N|\}$, the normalization of a clause can be achieved through a set of simple transformations, which are applied when a clause is initially introduced or when it is derived as a new conclusion:

- Tautologies are removed from $\mathcal{C}$. A regular clause is a tautology if it contains some regular literal of the form $\geq 1:x$ or of the form $\leq |N|:x$, or if it contains two regular literals of the form $\leq i:x$ and $\geq j:x$ such that $j \leq i+1$.
- Literals are always expressed without negation symbols, i.e. $\overline{\leq i:x}$ is replaced with $\geq i+1:x$, and $\overline{\geq i:x}$ is replaced with $\leq i-1:x$. The special cases $\overline{\leq |N|:x} = \overline{\geq 1:x} = \{\}:x$ are directly removed from the clause.
- If a clause contains multiple regular literals with positive polarity for a variable x, say $\geq i_1:x,\ldots,\geq i_k:x$, they are replaced with $\bigcup_{j=1}^{k} \geq i_j:x$, which also is a regular literal with positive polarity. Similarly, if it contains multiple regular literals with negative polarity for variable x, say $\leq i_1:x,\ldots,\leq i_{k'}:x$, they are replaced with $\bigcup_{j=1}^{k'} \leq i_j:x$, which also is a regular literal with negative polarity.

These transformations are sound since they do not alter the set of satisfying assignments of a clause. They can be applied in $O(nlog(n))$ time in the size of the clause. The following inference rules assume that the premises are normalized and require the same normalization to be enforced on the conclusions.

Definition 9. The Regular MaxSAT resolution calculus is formed by the following inference rules:

Rule 1

$$\begin{array}{lll} c_\alpha: & \geq j:x \vee A \\ c_\beta: & \leq k:x \vee B \\ \hline c_\cap: & A \vee B \\ c_\cup: & \leq k:x \vee \geq j:x \vee A \vee B \\ C_\alpha: & \geq j:x \vee A \vee \overline{B} \\ C_\beta: & \leq k:x \vee \overline{A} \vee B \end{array}$$

provided that $k < j$
and $A \vee B$ is not a tautology

Rule 2

$$\begin{array}{lll} c_\alpha: & \geq j:x \vee A \\ c_\beta: & \leq k:x \vee \geq l:x \vee B \\ \hline c_\cap: & \geq l:x \vee A \vee B \\ c_\cup: & \leq k:x \vee \geq j:x \vee A \vee B \\ C_\alpha: & \geq j:x \vee A \vee \overline{B} \\ C_\beta: & \leq k:x \vee \geq l:x \vee \overline{A} \vee B \end{array}$$

provided that $k < j < l$
and $A \vee B$ is not a tautology

$$
\begin{array}{ll}
c_\alpha: & \geq j : x \vee a_1 \vee \cdots \vee a_s \\
c_\beta: & \leq k : x \vee b_1 \vee \cdots \vee b_t \\
\hline
c_\cap: & a_1 \vee \cdots \vee a_s \vee b_1 \vee \cdots \vee b_t \\
c_\cup: & \geq j : x \vee \leq k : x \vee a_1 \vee \cdots \vee a_s \vee b_1 \vee \cdots \vee b_t \\
C_{\alpha,1}: & \geq j : x \vee a_1 \vee \cdots \vee a_s \vee \overline{b_1} \\
C_{\alpha,2}: & \geq j : x \vee a_1 \vee \cdots \vee a_s \vee b_1 \vee \overline{b_2} \\
\cdots & \\
C_{\alpha,t}: & \geq j : x \vee a_1 \vee \cdots \vee a_s \vee b_1 \vee \cdots \vee b_{t-1} \vee \overline{b_t} \\
C_{\beta,1}: & \leq k : x \vee b_1 \vee \cdots \vee b_t \vee \overline{a_1} \\
C_{\beta,2}: & \leq k : x \vee b_1 \vee \cdots \vee b_t \vee a_1 \vee \overline{a_2} \\
\cdots & \\
C_{\beta,s}: & \leq k : x \vee b_1 \vee \cdots \vee b_t \vee a_1 \vee \cdots \vee a_{s-1} \vee \overline{a_s}
\end{array}
$$

Fig. 1. Rule 1 in clausal form.

Rule 3

$$
\begin{array}{ll}
c_\alpha: & \leq i : x \vee \geq j : x \vee A \\
c_\beta: & \leq k : x \vee B \\
\hline
c_\cap: & \leq i : x \vee A \vee B \\
c_\cup: & \leq k : x \vee \geq j : x \vee A \vee B \\
C_\alpha: & \leq i : x \vee \geq j : x \vee A \vee \overline{B} \\
C_\beta: & \leq k : x \vee \overline{A} \vee B
\end{array}
$$

provided that $i < k < j$
and $A \vee B$ is not a tautology

Rule 4

$$
\begin{array}{ll}
c_\alpha: & \leq i : x \vee \geq j : x \vee A \\
c_\beta: & \leq k : x \vee \geq l : x \vee B \\
\hline
c_\cap: & \leq i : x \vee \geq l : x \vee A \vee B \\
c_\cup: & \leq k : x \vee \geq j : x \vee A \vee B \\
C_\alpha: & \leq i : x \vee \geq j : x \vee A \vee \overline{B} \\
C_\beta: & \leq k : x \vee \geq l : x \vee \overline{A} \vee B
\end{array}
$$

provided that $i < k < j < l$
and $A \vee B$ is not a tautology

where $A = a_1 \vee \cdots \vee a_s$ and $B = b_1 \vee \cdots \vee b_t$ are disjunctions of regular literals not containing variable x. The rules replace the premises with the clauses in the conclusion and we say that they resolve the premises on variable x. The tautologies concluded by the rule are omitted. Note that the literals of the two last conclusions are a superset of the literals of one of the premises.

Note that $\overline{A}$ and $\overline{B}$ are not in clausal form in the rules. To build clausal forms preserving the number of unsatisfied clauses, we use $\overline{A} = \{\overline{a_1}, a_1 \vee \overline{a_2}, \cdots, a_1 \vee \cdots \vee a_{s-1} \vee \overline{a_s}\}$ and $\overline{B} = \{\overline{b_1}, b_1 \vee \overline{b_2}, \cdots, b_1 \vee \cdots \vee b_{t-1} \vee \overline{b_t}\}$. Therefore, C_α and C_β are sets of clauses. Fig. 1 displays Rule 1 in clausal form.

Fig. 2 illustrates the application of Rules 1–4 to pairs of clauses that only contain variable x. Note that we establish an order between c_α and c_β in Rule 4 to avoid defining symmetric cases. Specifically, c_α is defined as the clause with the smallest negative literal. Similarly, from now on, when considering a pair of clauses c_α and c_β that both contain positive and negative literals for the same variable x, we always assume that c_α has the smallest negative literal. In case of a tie, c_α has the smallest (or equal) positive literal.

Example 4. If $N = \{1, 2, 3, 4, 5\}$, Rule 1 derives the multiset of regular clauses $\{\square, \leq 3 : x_1 \vee \geq 5 : x_1\}$ from the multiset $\{\leq 3 : x_1, \geq 5 : x_1\}$. Rule 2 derives the multiset of regular clauses $\{\geq 5 : x_1 \vee \geq 4 : x_2, \leq 2 : x_1 \vee \geq 4 : x_1 \vee \geq 4 : x_2\}$ from the multiset $\{\geq 4 : x_1 \vee \geq 4 : x_2, \leq 2 : x_1 \vee \geq 5 : x_1 \vee \geq 4 : x_2\}$. Rule 4 derives the multiset of regular clauses $\{\leq 1 : x_1 \vee \geq 5 : x_1 \vee \leq 2 : x_2, \leq 2 : x_1 \vee \geq 4 : x_1 \vee \leq 2 : x_2, \leq 2 : x_1 \vee \geq 5 : x_1 \vee \geq 3 : x_2\}$ from the multiset $\{\leq 1 : x_1 \vee \geq 4 : x_1 \vee \leq 2 : x_2, \leq 2 : x_1 \vee \geq 5 : x_1\}$.

Theorem 1. *The regular MaxSAT resolution calculus is sound.*

Proof. Given an assignment I, we prove that the number of unsatisfied clauses in the premises is the same as the number of unsatisfied clauses in the conclusion of the rule. We distinguish the following cases:

1. I unsatisfies the two premises: Since $I(A) = I(B) = I(\leq i : x) = I(\geq j : x) = I(\leq k : x) = I(\geq l : x) = false$, I unsatisfies the first two conclusions. It satisfies the other conclusions because either A or B appear negated in them. This holds for the four rules.
2. I satisfies the two premises: this case holds since the conjunction of the premises implies each one of the conclusions. More precisely, for all the rules, I satisfies the two last conclusions because their literals are a superset of the literals of some premise. This also holds for the second conclusion of Rules 1–3. For the second conclusion of Rule 4 we have three cases: (i) I satisfies A or B: I satisfies the second conclusion because it contains A and B; (ii) I unsatisfies A and B and satisfies $\geq j : x$: I satisfies the second conclusion because it contains $\geq j : x$; and (iii) I unsatisfies A and B and satisfies $\leq i : x$: Since $i < l$, I unsatisfies $\geq l : x$ and, since the second premise is satisfied, I satisfies $\leq k : x$. We now prove separately that I satisfies the first conclusion for each rule. For Rule 1, since $k < j$, I must satisfy A or B and, therefore, it satisfies the first conclusion. For Rule 2, we distinguish two cases: (i) I satisfies $\geq l : x$: It satisfies the first

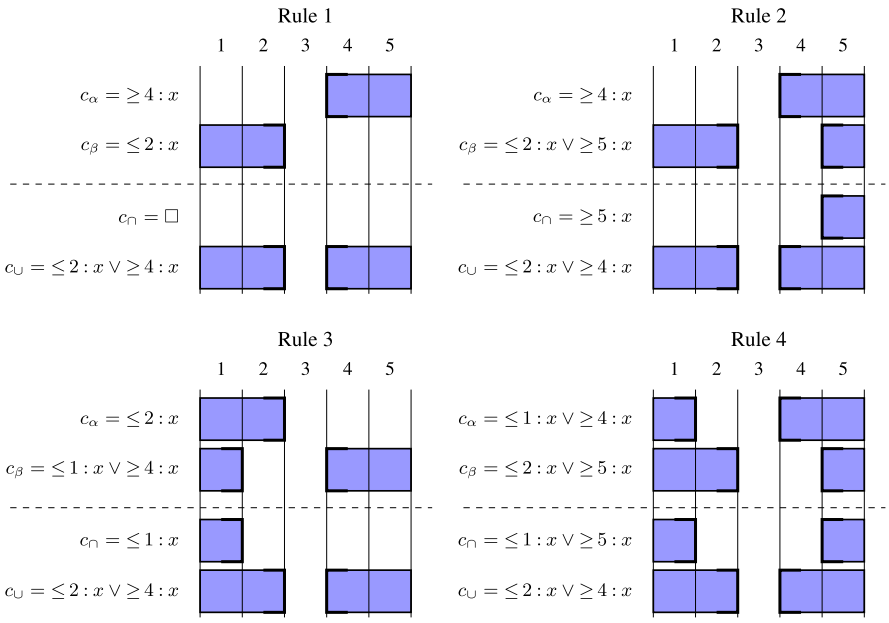

Fig. 2. Example of applications of Rule 1, Rule 2, Rule 3 and Rule 4 with clauses only containing variable x, with $N = \{1, \ldots, 5\}$. The values of x that each clause allows are painted in blue. (For interpretation of the colors in the figures, the reader is referred to the web version of this article.)

conclusion because it contains $\geq l : x$; and (ii) I unsatisfies $\geq l : x$: since $k < j$, I must satisfy A or B and, therefore, it satisfies the first conclusion. For Rule 3, we distinguish two cases: (i) I satisfies $\leq i : x$: It satisfies the first conclusion because it contains $\leq i : x$; and (ii) I unsatisfies $\leq i : x$: since $k < j$, I must satisfy A or B and, therefore, it satisfies the first conclusion. For Rule 4, we distinguish two cases: (i) I satisfies $\leq i : x$ or $\geq l : x$: It satisfies the first conclusion because it contains both literals; and (ii) I unsatisfies $\leq i : x$ and $\geq l : x$: since $k < j$, I must satisfy A or B and, therefore, it satisfies the first conclusion.

3. I satisfies the first premise and unsatisfies the second premise: For Rules 1–2, the second and third conclusions are implied by the first premise. Then, we distinguish two cases: (i) I satisfies A: the first conclusion is satisfied because it contains A and the last conclusion is unsatisfied because we assumed that all its literals are unsatisfied; and (ii) I unsatisfies A: the first conclusion is unsatisfied because we assumed that all its literals are unsatisfied and the last conclusion is satisfied because it contains $\overline{A}$. For Rules 3–4, we distinguish two cases: (i) I satisfies A: the three first conclusions are satisfied because they contain A and the last conclusion is clearly unsatisfied; and (ii) I unsatisfies A: $\leq i : x$ is unsatisfied because $\leq k : x$ is unsatisfied and $i < k$ and, therefore, $\geq j : x$ is satisfied. Thus, the first conclusion is unsatisfied and the rest are satisfied.
4. I unsatisfies the first premise and satisfies the second premise. For Rule 1 and Rule 3, the second and fourth conclusions are implied by the second premise. Then, we distinguish two cases: (i) I satisfies B: the first conclusion is satisfied because it contains B and the third conclusion is unsatisfied because we assumed that all its literals are unsatisfied; and (ii) I unsatisfies B: the first conclusion is unsatisfied because we assumed that all its literals are unsatisfied and the third conclusion is satisfied because it contains $\overline{B}$. For Rule 2 and Rule 4, we distinguish two cases: (i) I satisfies B: the first, second and fourth conclusions are satisfied because they contain B and the third conclusion is clearly unsatisfied; and (ii) I unsatisfies B: $\geq l : x$ is unsatisfied because $\geq j : x$ is unsatisfied and $j < l$ and, therefore, $\leq k : x$ is satisfied. Thus, the first conclusion is unsatisfied and the rest are satisfied. □

Definition 10. A multiset of regular clauses $\mathcal{C}$ is saturated w.r.t. a variable x if, for every pair of regular clauses c_α and c_β containing x, none of the rules of the calculus (Rule 1, Rule 2, Rule 3 and Rule 4) can be applied.

Fig. 3 shows an example of a saturated multiset of regular clauses w.r.t. variable x.

Lemma 1. *Let $\mathcal{C} = E \cup D$ be a saturated multiset of regular clauses w.r.t. a variable x, let E be the subset of clauses of $\mathcal{C}$ not containing x, and let D be the subset of clauses of $\mathcal{C}$ containing x. Assume that E is satisfiable. Then, any partial assignment I that satisfies E and does not assign any value to x can be extended to an assignment that satisfies $\mathcal{C}$.*

Proof. We have to extend I so that I satisfies $\mathcal{C}$. We partition D into the multiset D' and D'', where D' is not satisfied by I and D'' is satisfied by I. D' contains clauses of one of the following forms: $\leq i : x \vee \geq j : x \vee A$, $\leq i : x \vee A$, or $\geq j : x \vee A$. Since A is falsified in D', we assume that D' contains clauses without A in the remainder of the proof. Now, we define the

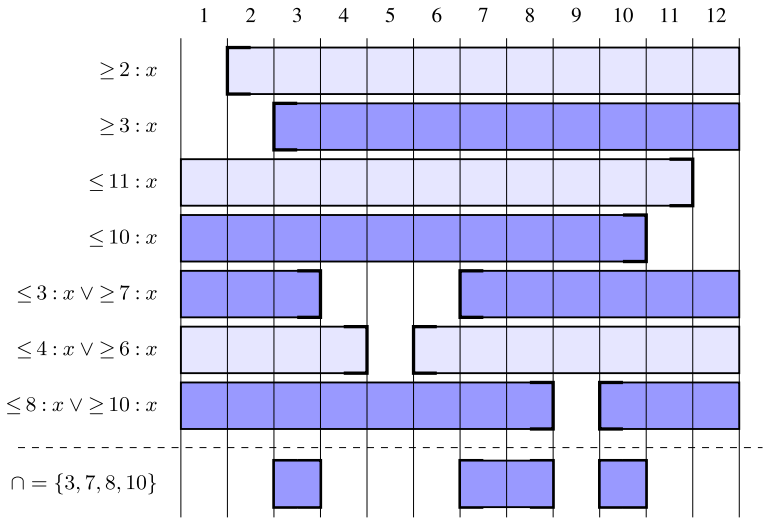

Fig. 3. Example of saturated formula with clauses that only contain variable x, with $N = \{1, \ldots, 12\}$. The values of x that each clause allows are painted in blue. The subsumed clauses are painted in light blue. The bottom row shows the intersection of the values allowed by the clauses.

set of values that x can take according to I, and show that it is never empty. If D' is empty, I can assign to x any value in N. From the set of unit clauses whose literal has negative polarity (i.e., of the form $\leq i : x$), we only need to consider the clause with the smallest i because the other negative literals are subsumed. Similarly, from the set of unit clauses whose literal has positive polarity (i.e., of the form $\geq j : x$), we only need to consider the clause with the greatest j. Therefore, the set of values that x can take is contained in the interval $[a, b]$, where the most restrictive unit clauses are $\geq a : x$ and $\leq b : x$. Note that $a \leq b$, otherwise we could apply Rule 1. In case no unit clause $\geq a : x$ exists (resp. no unit clause $\leq b : x$ exists), a is the lowest (resp. b is the greatest) value of N. Then, we consider three cases:

1. If no binary clause exists, I can assign to x any value in the set $[a, b]$.
2. If there is exactly one binary clause of the form $\leq i : x \vee \geq j : x$, then $a \leq i$ and $b \geq j$ by definition of saturation. Therefore, I can assign to x any value in the non-empty set $[a, i] \cup [j, b]$.
3. If there is more than one binary clause, as the formula is saturated w.r.t. x, any two binary clauses in D' of the form $c_s = \leq i : x \vee \geq j : x$ and $c_t = \leq k : x \vee \geq l : x$ satisfy one of the two following restrictions: (i) $i \leq k < l \leq j$ or (ii) $i < j \leq k < l$. As we do not have tautological clauses, it always holds that $i < j$ and $k < l$. Restriction (i) implies that clause c_s subsumes c_t. From now on, we consider that all subsumed clauses are removed, and hence only restriction (ii) can hold between two binary clauses. Therefore, we can establish a total ordering $c_1 \prec \cdots \prec c_n$ on the binary clauses in D' such that, for any two clauses c_s and c_{s+1} of the form $\leq i : x \vee \geq j : x \vee A$ and $\leq k : x \vee \geq l : x \vee B$, it holds that $j \leq k$. By transitivity, this relation also happens between any two clauses c_s and c_t such that $s < t$. Note also that $c_1 = \leq i : x \vee \geq j : x$ satisfies that $a \leq i$; otherwise, we could apply Rule 2. Similarly, $c_n = \leq i : x \vee \geq j : x$ satisfies that $b \geq j$; otherwise, we could apply Rule 3. Let c_s and c_{s+1} be any two consecutive clauses of the sequence. Clearly, these two clauses allow to assign to x a non-empty set of values. Specifically, x can take any value on the interval $[j, k] \subset [a, b]$. Note also that any clause to the left of c_s also admits the values in $[j, k]$, and the same happens with any clause to the right of c_{s+1}. Therefore, letting clause c_s be of the form $\leq i_s : x \vee \geq j_s : x$, the values that I can assign to x are:

$$[a, i_1] \cup \bigcup_{1 \leq s \leq n-1} [j_s, i_{s+1}] \cup [j_n, b]$$

Thus, if we extend I by assigning to x any value of the indicated non-empty set, I satisfies D' and $\mathcal{C}$. □

In the proof, we made use of an abuse of notation. Whenever we write an interval $[i, j]$, we are actually referring to the set $[i, j] \cap N$. Note that the clauses in the example of Fig. 3 serve as an example of the set D' in the proof of Lemma 1.

Next, we proceed to prove the completeness of the Regular MaxSAT resolution calculus, as stated in Theorem 2. Before that, we introduce some necessary notation and auxiliary lemmas.

Definition 11. The scope of a clause c, denoted by $scope(c)$, is the set of variables appearing in c. The scope of a multiset of clauses $\mathcal{C}$ is the set of variables appearing in $\mathcal{C}$.

Definition 12. Given a clause c and a set of variables X such that $X \supseteq scope(c)$, $score(c, X)$ denotes the set of assignments over X that unsatisfy c. Given a multiset of clauses $\mathcal{C}$, $score(\mathcal{C}, X)$ denotes the sum of the scores of the clauses in $\mathcal{C}$.

When clear from the context, such as when X corresponds to $scope(\mathcal{C})$ given a CNF formula $\mathcal{C}$, we write $score(\mathcal{C})$.

Lemma 2. *Let $\mathcal{C}$ be a multiset of regular clauses. The iterative application of Rules 1–4 for resolving on variable x always terminates regardless of the order of application; i.e., it reaches a state where no rule can be applied to any two clauses of $\mathcal{C}$. Hence, the result is a multiset of regular clauses which is saturated w.r.t. x.*

Proof. In order to unify the proof for the four rules we use signed literal notation and follow the proof in [2]. In particular, considering a truth value set $N = \{1, \ldots, |N|\}$, a disjunction of regular literals $\leq i : x \vee \geq j : x$ can be denoted by $\{1, \ldots, i, j, \ldots, |N|\}$:$x$. The overall idea of the proof is that, given a formula $\mathcal{C} = F \cup \{c_\alpha, c_\beta\}$ that is not saturated w.r.t. variable x, the application of any of the Rules 1–4 over c_α, c_β results in a formula $\mathcal{C}' = F \cup \{c_\cap, c_\cup\} \cup C_a \cup C_b$ that is closer to a saturation w.r.t. variable x. Let us partition the clauses of $\mathcal{C}$ into a sequence of $|N|$ multisets $\mathcal{B}$=$\mathcal{B}_0, \mathcal{B}_1, \ldots, \mathcal{B}_{|N|-1}$, such that the multiset $\mathcal{B}_i$ contains all clauses from $\mathcal{C}$ of the form S:$x \vee A$ such that $|S|$=i. $\mathcal{B}_0$ is the multiset of clauses not containing variable x. If M=$score(\mathcal{C})$, then $score(\mathcal{B}_i) \leq M$ for $i \in 0, \ldots, |N|-1$. Let us look at $score(\mathcal{B}_0) score(\mathcal{B}_1) \ldots score(\mathcal{B}_{|N|-1})$ as a word of length $|N|$ and base $M+1$, where $score(\mathcal{B}_0)$ is the most significant digit. By Theorem 1, the preservation of the number of unsatisfied clauses guarantees that $score(\mathcal{C})$=$score(\mathcal{C}')$, and therefore that no digit reaches a value greater than M after an application of a rule. However, we will show that the value of the $|N|$-digit word always increases when we apply any of the rules, which completes the proof since this word has a finite maximum value. When we apply one of Rules 1–4, we remove clauses c_α=S_α:$x \vee A$ and c_β=S_β:$x \vee B$ from their corresponding multisets $\mathcal{B}_{|S_\alpha|}$, $\mathcal{B}_{|S_\beta|}$. We also add a number of new clauses to some multisets of $\mathcal{B}$, including clause $c_\cap$=$S_\cap$:$x \vee A \vee B$. By definition of the rules, $S_\cap \subset S_\alpha, S_\beta$ (see also Fig. 2), hence $|S_\cap| < |S_\alpha|, |S_\beta|$ and also $score(c_\alpha) > 0$. Therefore, $c_\cap$ will be inserted into a digit of $\mathcal{B}$ more significant than the digits of the removed clauses c_α, c_β, thus increasing the value of the word. □

Theorem 2. *For any multiset of regular clauses $\mathcal{C}$, we have that*

$$\mathcal{C} \vdash \underbrace{\square, \ldots, \square}_{m}, \mathcal{D}$$

where $\mathcal{D}$ is a satisfiable multiset of regular clauses, m is the minimum number of unsatisfied clauses of $\mathcal{C}$, and $\mathcal{C} \vdash \square, \ldots, \square, \mathcal{D}$ denotes that the multiset of clauses $\{\square, \ldots, \square, \mathcal{D}\}$ can be obtained from the multiset $\mathcal{C}$ by applying the Regular MaxSAT resolution rule a finite number of times.

Proof. Let $x_1, \ldots, x_n$ be any list of the variables occurring in $\mathcal{C}$. We construct two sequences of multisets $C_0, \ldots, C_n$ and $D_1, \ldots, D_n$ so that $\mathcal{C} = C_0$; for $i = 1, \ldots, n$, $C_i \cup D_i$ is a saturation of C_{i-1} w.r.t. x_i; and, for $i = 1, \ldots, n$, C_i is the multiset of clauses not containing variables $x_1, \ldots, x_i$ and D_i is the multiset of clauses containing variable x_i. These sequences can be computed in a finite number of steps because C_{i+1} contains one variable less than C_i, and by Lemma 2 the saturation of C_i can be computed in a finite number of steps. For $i = 1, \ldots, n$, we saturate C_{i-1} w.r.t. x_i, and then partition the resulting multiset into a subset D_i containing x_i, and another subset C_i not containing x_i. Since C_n contains no variables, it is either the empty multiset or it just contains a finite number of empty clauses $\{\square, \ldots, \square\}$.

We will define a complete satisfying assignment for the multiset $D = \bigcup_{i=1}^{n} D_i$ to prove that it is satisfiable. For $i = 1, \ldots, n$, let $d_i = D_i \cup \cdots \cup D_n$ and let $d_{n+1} = \emptyset$. Notice that d_i only contains the variables in $\{x_i, \ldots, x_n\}$, $d_i = D_i \cup d_{i+1}$ and d_i is saturated w.r.t. x_i because d_{i+1} does not contain x_i and $C_i \cup D_i$ is saturated w.r.t. x_i.

We now define a sequence of assignments $I_1, \ldots, I_{n+1}$, where I_{n+1} is the empty assignment and, therefore, satisfies d_{n+1}. Assignment I_i is defined from assignment I_{i+1} as follows: Assume by induction hypothesis that I_{i+1} satisfies d_{i+1}. Since d_i is saturated w.r.t. x_i and $d_i = D_i \cup d_{i+1}$, by Lemma 1, we can extend I_{i+1} with a value of x_i that satisfies d_i. Iterating, we have that I_1 satisfies $d_1 = D = \bigcup_{i=1}^{n} D_i$.

Since the soundness theorem ensures that the regular MaxSAT resolution rules preserve the number of unsatisfied clauses for every assignment, the number of empty clauses in C_n ($|C_n| = m$) is the minimum number of clauses of $\mathcal{C}$ that can be unsatisfied. In particular, assignment I_1 falsifies exactly m clauses of $\mathcal{C}$. □

4. An exact variable elimination algorithm for Regular MaxSAT

Algorithm 1 shows the pseudo-code of a variable elimination algorithm derived from the proof of Theorem 2: Given an input multiset of regular clauses $\mathcal{C}$ with n different variables, the algorithm returns the minimum number m of clauses of $\mathcal{C}$ that will be unsatisfied for any assignment, and a total optimal assignment I that falsifies exactly m clauses of $\mathcal{C}$.

The variable elimination algorithm has the following functions:

- Function $saturation(C, x)$ computes a saturation of C w.r.t. x. By Lemma 2, this can be implemented by systematically trying to apply Rules 1–4 to any pair of clauses containing x, until no application can be done.

Algorithm 1: An exact variable elimination algorithm for MaxSAT.

Input: $\mathcal{C}$: a multiset of regular clauses.
Output: (m, I): an assignment I that unsatisfies the minimum number of clauses (m).
$C_0 \leftarrow \mathcal{C}$;
for $i \leftarrow 1$ **to** n **do**
  $C' \leftarrow saturation(C_{i-1}, x_i)$;
  $\langle C_i, D_i \rangle \leftarrow partition(C', x_i)$;
$m \leftarrow |C_n|$;
$I \leftarrow \emptyset$;
for $i \leftarrow n$ **downto** 1 **do**
  $I \leftarrow I \cup [x_i \mapsto max\_extension(x_i, I, D_i)]$;
return (m, I)

- Function $partition(C, x_i)$ partitions C into two multisets, C_i and D_i so that C_i contains the regular clauses without occurrences of variable x_i, and D_i contains the regular clauses with occurrences of x_i.
- Function $max\_extension(x_i, I, D_i)$ computes a truth assignment for x_i as follows: if I satisfies all the clauses in D_i, including the case in which $D_i = \{\}$, then the function returns the greatest truth value. Otherwise, by Lemma 1, I assigns to x_i one value of the non-empty set $[a, i_1] \cup \bigcup_{1 \leq s \leq n-1} [j_s, i_{s+1}] \cup [j_n, b]$.

The algorithm has two parts. In the first part, the algorithm successively saturates w.r.t. all the variables occurring in the input multiset. Once the current multiset is saturated w.r.t. a variable x_i, it partitions the resulting multiset into two multisets: C_i and D_i. The multiset C_i contains the clauses without occurrences of x_i, and the multiset D_i contains the clauses with occurrences of x_i. The algorithm continues saturating C_i w.r.t. one of the remaining variables, and ignores D_i. This process continues until all the variables are eliminated. At the end, C_n does not contain any variable, and the number of empty clauses in C_n is the returned minimum number of unsatisfied clauses. In the second part, the algorithm builds an optimal assignment as function $max\_extension(x_i, I, D_i)$ states.

Example 5. Let $N = \{1, 2, 3, 4, 5\}$ and let $C_0 = \{\leq 1 : x_3, \geq 3 : x_1 \vee \geq 2 : x_2, \leq 1 : x_1 \vee \geq 2 : x_2, \leq 2 : x_2 \vee \geq 2 : x_3, \geq 1 : x_2 \vee \leq 4 : x_3, \geq 3 : x_3\}$. Saturating C_0 w.r.t. x_1, we get $C_1 = \{\leq 1 : x_3, \geq 2 : x_2, \leq 2 : x_2 \vee \geq 2 : x_3, \geq 1 : x_2 \vee \geq 4 : x_3, \geq 3 : x_3\}$ and $D_1 = \{\leq 1 : x_1 \vee \geq 3 : x_1 \vee \geq 2 : x_2\}$. Saturating C_1 w.r.t. x_2, we get $C_2 = \{\leq 1 : x_3, \geq 3 : x_3\}$ and $D_2 = \{\geq 2 : x_2, \leq 2 : x_2 \vee \geq 2 : x_3, \geq 1 : x_2 \vee \geq 4 : x_3\}$. Saturating C_2 w.r.t. x_3, we get $C_3 = \{\square\}$ and $D_3 = \{\leq 1 : x_3 \vee \geq 3 : x_3\}$. Hence, the minimum number of unsatisfied clauses is 1, and $x_3 \mapsto 5, x_2 \mapsto 5, x_1 \mapsto 5$ is an optimal assignment.

5. Weighted Regular MaxSAT resolution

For ease of presentation, we have presented the inference rules for (unweighted) Regular MaxSAT. The weighted version of the inference rules of the proposed Regular MaxSAT proof systems are shown in Fig. 4. Conclusions with weight zero are omitted.

From a conceptual viewpoint, a clause with weight w is equivalent to having w copies of that clause. Thus, the application of a weighted rule collapses $\min(u, w)$ applications of the unweighted rule. Since the premises can have different weights, we add again the premises with weights $u - \min(u, w)$ and $w - \min(u, w)$. One of such weights is zero, and the other allows to use the copies of the premise that have not been consumed in other inference steps. So, the results of soundness and completeness for the weighted case follow directly from the results in the unweighted case. Nevertheless, in practice, it is much more efficient to apply the weighted rules when we have weighted clauses.

6. Conclusions

We have defined the first resolution-style proof system for Regular MaxSAT and proved its soundness and completeness, described the first exact variable elimination for Regular MaxSAT that incorporates a notion of saturation that exploits the structure of regular signs, and extended the proposed proof system to solve Weighted Regular MaxSAT.

Finally, we want to point out several future research directions:

- A first extension of our work is to consider that the truth value set is infinite. In this case, we should consider four types of regular signs: $\geq i$, $> i$, $\leq i$ or $< i$ in order to deal with negations of regular literals.
- A second extension of our work is to associate a partial order with the truth value set instead of a total order. The existing works that use lattices as truth value sets are a good starting point for this research direction.
- A third extension is to study the advantages of regular encodings for representing combinatorial optimization problems.
- A fourth extension is to implement the proposed variable elimination algorithm for Regular MaxSAT and compare its performance with a Signed MaxSAT algorithm. Our intuition is that, for the same problem, Regular MaxSAT will generally be more efficient than Signed MaxSAT because it better exploits the structure of the CNF formula.

Rule 1

$$\frac{\begin{array}{l} c_\alpha : \ (\geq j : x \vee A, u) \\ c_\beta : \ (\leq k : x \vee B, w) \end{array}}{\begin{array}{l} c_\cap : \ (A \vee B, m) \\ c_\cup : \ (\leq k : x \vee \geq j : x \vee A \vee B, m) \\ C_\alpha : \ (\geq j : x \vee A \vee \overline{B}, m) \\ C_\beta : \ (\leq k : x \vee \overline{A} \vee B, m) \\ c'_\alpha : \ (\geq j : x \vee A, u - m) \\ c'_\beta : \ (\leq k : x \vee B, w - m) \end{array}}$$

provided that $k < j$,
$A \vee B$ is not a tautology
and $m = \min(u, w)$

Rule 2

$$\frac{\begin{array}{l} c_\alpha : \ (\geq j : x \vee A, u) \\ c_\beta : \ (\leq k : x \vee \geq l : x \vee B, w) \end{array}}{\begin{array}{l} c_\cap : \ (\geq l : x \vee A \vee B, :) \\ c_\cup : \ (\leq k : x \vee \geq j : x \vee A \vee B, m) \\ C_\alpha : \ (\geq j : x \vee A \vee \overline{B}, m) \\ C_\beta : \ (\leq k : x \vee \geq l : x \vee \overline{A} \vee B, m) \\ c'_\alpha : \ (\geq j : x \vee A, u - m) \\ c'_\beta : \ (\leq k : x \vee \geq l : x \vee B, w - m) \end{array}}$$

provided that $k < j < l$
$A \vee B$ is not a tautology
and $m = \min(u, w)$

Rule 3

$$\frac{\begin{array}{l} c_\alpha : \ (\leq i : x \vee \geq j : x \vee A, u) \\ c_\beta : \ (\leq k : x \vee B, w) \end{array}}{\begin{array}{l} c_\cap : \ (\leq i : x \vee A \vee B, :) \\ c_\cup : \ (\leq k : x \vee \geq j : x \vee A \vee B, m) \\ C_\alpha : \ (\leq i : x \vee \geq j : x \vee A \vee \overline{B}, m) \\ C_\beta : \ (\leq k : x \vee \overline{A} \vee B, m) \\ c'_\alpha : \ (\leq i : x \vee \geq j : x \vee A, u - m) \\ c'_\beta : \ (\leq k : x \vee B, w - m) \end{array}}$$

provided that $i < k < j$
$A \vee B$ is not a tautology
and $m = \min(u, w)$

Rule 4

$$\frac{\begin{array}{l} c_\alpha : \ (\leq i : x \vee \geq j : x \vee A, u) \\ c_\beta : \ (\leq k : x \vee \geq l : x \vee B, w) \end{array}}{\begin{array}{l} c_\cap : \ (\leq i : x \vee \geq l : x \vee A \vee B, m) \\ c_\cup : \ (\leq k : x \vee \geq j : x \vee A \vee B, m) \\ C_\alpha : \ (\leq i : x \vee \geq j : x \vee A \vee \overline{B}, m) \\ C_\beta : \ (\leq k : x \vee \geq l : x \vee \overline{A} \vee B, m) \\ c'_\alpha : \ (\leq i : x \vee \geq j : x \vee A, u - m) \\ c'_\beta : \ (\leq k : x \vee \geq l : x \vee B, w - m) \end{array}}$$

provided that $i < k < j < l$
$A \vee B$ is not a tautology
and $m = \min(u, w)$

Fig. 4. Proof system for Weighted Regular MaxSAT.

Declaration of competing interest

The authors declare the following financial interests/personal relationships which may be considered as potential competing interests:

Jordi Coll, Felip Manyà and Elifnaz Yangin report financial support was provided by Spain Ministry of Science and Innovation. Chu-Min Li reports financial support was provided by French National Research Agency.

Data availability

No data was used for the research described in the article.

Acknowledgements

This work has been supported by the French Agence Nationale de la Recherche, reference ANR-19-CHIA-0013-01, and grants PID2019-111544GB-C21 and TED2021-129319B-I00 funded by MCIN/AEI/10.13039/501100011033.

References

[1] André Abramé, Djamal Habet Ahmaxsat, Description and evaluation of a branch and bound Max-SAT solver, J. Satisf. Boolean Model. Comput. 9 (2014) 89–128.
[2] Carlos Ansótegui, María Luisa Bonet, Jordi Levy, Felip Manyà, A complete resolution calculus for Signed Max-SAT, in: Proceedings of the 37th International Symposium on Multiple-Valued Logic (ISMVL), Oslo, Norway, IEEE CS Press, 2007, pp. 22–27.
[3] Carlos Ansótegui, María Luisa Bonet, Jordi Levy, Felip Manyà, The logic behind weighted CSP, in: Proceedings of the 20th International Joint Conference on Artificial Intelligence, IJCAI-2007, Hyderabad, India, 2007, pp. 32–37.
[4] Carlos Ansótegui, Maria Luisa Bonet, Jordi Levy, Felip Manyà, Resolution procedures for multiple-valued optimization, Inf. Sci. 227 (2013) 43–59.
[5] Carlos Ansótegui, Jose Larrubia, Chu Min Li, Felip Manyà, Exploiting multivalued knowledge in variable selection heuristics for SAT solvers, Ann. Math. Artif. Intell. 49 (1–4) (2007) 191–205.
[6] Carlos Ansótegui, Jose Larrubia, Felip Manyà, Boosting Chaff's performance by incorporating CSP heuristics, in: 9th International Conference on Principles and Practice of Constraint Programming, CP-2003, Kinsale, Ireland, in: LNCS, vol. 2833, Springer, 2003, pp. 96–107.
[7] Carlos Ansótegui, Felip Manyà, Mapping problems with finite-domain variables into problems with Boolean variables, in: Proceedings of the 7th International Conference on Theory and Applications of Satisfiability Testing (Revised Selected Papers), SAT-2004, Vancouver, Canada, in: LNCS, vol. 3542, Springer, 2004, pp. 1–15.
[8] Carlos Ansótegui, Felip Manyà, Jesus Ojeda, Josep M. Salvia, Eduard Torres, Incomplete MaxSAT approaches for combinatorial testing, J. Heuristics 28 (4) (2022) 377–431.
[9] Fahiem Bacchus, Matti Järvisalo, Martins Ruben, Maximum satisfiability, in: Armin Biere, Marijn Heule, Hans van Maaren, Toby Walsh (Eds.), Handbook of Satisfiability, IOS Press, 2021, pp. 929–991.

[10] Bernhard Beckert, Reiner Hähnle, Felip Manyà, Transformations between signed and classical clause logic, in: Proceedings, 29th International Symposium on Multiple-Valued Logics (ISMVL), Freiburg, Germany, IEEE Press, Los Alamitos, 1999, pp. 248–255.
[11] Bernhard Beckert, Reiner Hähnle, Felip Manyà, The 2-SAT problem of regular signed CNF formulas, in: Proceedings, 30th International Symposium on Multiple-Valued Logics (ISMVL), Portland/OR, USA, IEEE CS Press, Los Alamitos, 2000, pp. 331–336.
[12] Bernhard Beckert, Reiner Hähnle, Felip Manyà, The SAT problem of signed CNF formulas, in: David Basin, Marcello D'Agostino, Dov Gabbay, Seán Matthews, Luca Viganò (Eds.), Labelled Deduction, in: Applied Logic Series, vol. 17, Kluwer, Dordrecht, 2000, pp. 61–82.
[13] R. Béjar, F. Manyà, Solving combinatorial problems with regular local search algorithms, in: Proceedings of the 6th International Conference on Logic for Programming and Automated Reasoning, LPAR'99, Tbilisi, Republic of Georgia, in: LNAI, vol. 1705, Springer, 1999, pp. 33–43.
[14] Ramón Béjar, Alba Cabiscol, C. Fernández, Felip Manyà, Carla P. Gomes, Capturing structure with satisfiability, in: 7th International Conference on Principles and Practice of Constraint Programming, CP-2001, Paphos, Cyprus, in: LNCS, vol. 2239, Springer, 2001, pp. 137–152.
[15] Miquel Bofill, Jordi Coll, Marc Garcia, Jesús Giráldez-Cru, Gilles Pesant, Josep Suy, Mateu Villaret, Constraint solving approaches to the business-to-business meeting scheduling problem, J. Artif. Intell. Res. 74 (2022) 263–301.
[16] Maria Luisa Bonet, Sam Buss, Alexey Ignatiev, João Marques-Silva, António Morgado, MaxSAT resolution with the dual rail encoding, in: Proceedings of the 32nd AAAI Conference on Artificial Intelligence, AAAI, New Orleans, Louisiana, USA, 2018, pp. 6565–6572.
[17] Maria Luisa Bonet, Jordi Levy, Equivalence between systems stronger than resolution, in: Proceedings of the 23rd International Conference on Theory and Applications of Satisfiability Testing, SAT-2020, Alghero, Italy, in: LNCS, vol. 12178, Springer, 2020, pp. 166–181.
[18] María Luisa Bonet, Jordi Levy, Felip Manyà, Resolution for Max-SAT, Artif. Intell. 171 (8–9) (2007) 240–251.
[19] Dominique D'Almeida, Éric Grégoire, Model-based diagnosis with default information implemented through MAX-SAT technology, in: Proceedings of the IEEE 13th International Conference on Information Reuse & Integration, IRI, Las Vegas, NV, USA, 2012, pp. 33–36.
[20] Guido Fiorino, New tableau characterizations for non-clausal MaxSAT problem, Log. J. IGPL 30 (3) (2022) 422–436.
[21] Reiner Hähnle, Automated Deduction in Multiple-Valued Logics, International Series of Monographs in Computer Science, vol. 10, Oxford University Press, 1994.
[22] Reiner Hähnle, Short conjunctive normal forms in finitely-valued logics, J. Log. Comput. 4 (6) (1994) 905–927.
[23] Armin Haken, The intractability of resolution, Theor. Comput. Sci. 39 (1985) 297–308.
[24] Alexey Ignatiev, António Morgado, João Marques-Silva, On tackling the limits of resolution in SAT solving, in: Proceedings of the 20th International Conference on Theory and Applications of Satisfiability Testing, SAT, Melbourne, Australia, in: LNCS, vol. 10491, Springer, 2017, pp. 164–183.
[25] Michael Kifer, V.S. Subrahmanian, Theory of generalized annotated logic programming and its applications, J. Log. Program. 12 (1992) 335–367.
[26] Javier Larrosa, Federico Heras, Simon de Givry, A logical approach to efficient Max-SAT solving, Artif. Intell. 172 (2–3) (2008) 204–233.
[27] Javier Larrosa, Emma Rollon, Towards a better understanding of (partial weighted) MaxSAT proof systems, in: Proceedings of the 23rd International Conference on Theory and Applications of Satisfiability Testing, SAT-2020, Alghero, Italy, in: LNCS, vol. 12178, Springer, 2020, pp. 218–232.
[28] Chu Min Li, F. Manyà, MaxSAT, hard and soft constraints, in: Armin Biere, Marijn Heule, Hans van Maaren, Toby Walsh (Eds.), Handbook of Satisfiability, IOS Press, 2021, pp. 903–927.
[29] Chu Min Li, Felip Manyà, Nouredine Ould Mohamedou, Jordi Planes, Exploiting cycle structures in Max-SAT, in: Proceedings of the 12th International Conference on Theory and Applications of Satisfiability Testing, SAT-2009, Swansea, UK, in: Lect. Notes Comput. Sci., vol. 5584, Springer, 2009, pp. 467–480.
[30] Chu Min Li, Felip Manyà, Jordi Planes, New inference rules for Max-SAT, J. Artif. Intell. Res. 30 (2007) 321–359.
[31] Chu Min Li, Felip Manyà, Joan Ramon Soler, A tableau calculus for non-clausal maximum satisfiability, in: Proceedings of the 28th International Conference on Automated Reasoning with Analytic Tableaux and Related Methods, TABLEAUX, London, UK, in: LNCS, vol. 11714, Springer, 2019, pp. 58–73.
[32] Chu-Min Li, Zhenxing Xu, Jordi Coll, Felip Manyà, Djamal Habet, Kun He, Combining clause learning and branch and bound for MaxSAT, in: Proceedings of the 27th International Conference on Principles and Practice of Constraint Programming, CP, Montpellier, France, in: LIPIcs, vol. 210, Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2021, pp. 38:1–38:18.
[33] Felip Manyà, The 2-SAT problem in signed CNF formulas, Mult.-Valued Log. Int. J. 5 (4) (2000) 307–325.
[34] Felip Manyà, Santiago Negrete, Carme Roig, Joan Ramon Soler, Solving the team composition problem in a classroom, Fundam. Inform. 174 (1) (2020) 83–101.
[35] João Marques-Silva, Josep Argelich, Ana Graça, Inês Lynce, Boolean lexicographic optimization: algorithms & applications, Ann. Math. Artif. Intell. 62 (3–4) (2011) 317–343.
[36] Sean Safarpour, Hratch Mangassarian, Andreas G. Veneris, Mark H. Liffiton, Karem A. Sakallah, Improved design debugging using maximum satisfiability, in: Proceedings of 7th International Conference on Formal Methods in Computer-Aided Design, FMCAD, Austin, Texas, USA, 2007, pp. 13–19.
[37] Naoyuki Tamura, Akiko Taga, Satoshi Kitagawa, Mutsunori Banbara, Compiling finite linear CSP into SAT, Constraints 14 (2009) 254–272.
[38] Lei Zhang, Fahiem Bacchus, MAXSAT heuristics for cost optimal planning, in: Proceedings of the 26th AAAI Conference on Artificial Intelligence, Toronto, Ontario, Canada, 2012, pp. 1846–1852.