Inferring Clauses and Formulas in Max-SAT

Matthieu Py, Mohamed Sami Cherif, Djamal Habet

Inferring Clauses and Formulas in Max-SAT

Matthieu Py, Mohamed Sami Cherif and Djamal Habet

Aix-Marseille Université, Université de Toulon, CNRS, LIS, Marseille, France
{matthieu.py, mohamed-sami.cherif, djamal.habet}@univ-amu.fr

***Abstract*—In this paper, we are interested in proof systems for Max-SAT and particularly in the construction of Max-SAT equivalence-preserving transformations to infer information from a given formula. To this end, we introduce the notion of explainability and we provide a characterization for explainable clauses and formulas. Furthermore, we introduce a new proof system, called Explanation Calculus (ExC) and composed of two rules: symmetric cut and expansion. We study the relation between ExC and several existing proof systems. Then, we introduce a new algorithm, called explanation algorithm, able to construct an explanation in ExC for any clause or refute its explainability and we extend it for formula explanations. Finally, we use our results on explainability to provide proofs for the Max-SAT problem with a new bound on the number of inference steps in the proof, improving the bound obtained with the Max-SAT resolution calculus.**

***Index Terms*—Max-SAT, Proof Systems, Inference**

I. Introduction

Given a Boolean formula in Conjunctive Normal Form (CNF), the Max-SAT problem consists in determining the maximum number of clauses that it is possible to satisfy by an assignment of the variables. A well-known proof system for Max-SAT is Max-SAT resolution [8], [9], [17] which is an adaptation of the resolution rule [27] defined in the context of the SAT problem. The Max-SAT resolution calculus is sound and complete for the Max-SAT problem. Indeed, Max-SAT resolution can transform, through successive applications, any CNF formula ϕ to a satisfiable formula ϕ' and a multiset of empty clauses whose size is the optimum of ϕ. Lately, the Max-SAT resolution system has been extended using the split rule to improve its theoretical efficiency [20], [25], [7].

This paper also focuses on proof systems for Max-SAT. We introduce the notion and characterization of explainable clauses and formulas, i.e. clauses and formulas which can be deduced from a given formula following a finite sequence of transformations preserving Max-SAT equivalence. As Max-SAT resolution is not inferentially complete, we define a new proof system, called Explanation Calculus (ExC), composed of the symmetric cut rule and a new rule which we refer to as expansion. We also study the relation between ExC and other relative proof systems. Then, using the characterization of explainable clauses, we propose an algorithm, called the explanation algorithm, able to build an explanation using ExC rules for a given clause or refute its explainability. We prove that this can be achieved in $O(2^n)$ inference steps, where n is the number of variables in the formula. Then, using the characterization of explainable formulas, we extend the explanation algorithm to explain any formula or refute its explainability with a proof size of at most $O(m \times 2^n)$ inference steps, where m is the number of clauses in the formula. We also use it to construct proofs for the (plain) Max-SAT problem with the same upper bound. Consequently, we obtain a better upper bound for proof sizes than the current known result using the Max-SAT resolution calculus. Finally, we explain how to adapt this result to weighted partial formulas.

This paper is organized as follows. Section II includes some necessary definitions and notations and a brief overview of proof systems for Max-SAT. In Section III, the notion of explainable clauses and formulas is introduced and a characterization for such clauses and formulas is established. In Section IV, we introduce the Explanation Calculus (ExC) and we study its relation with other proof systems. In Section V, we introduce the explanation algorithm, able to construct an explanation for any given clause or refute its existence and we naturally extend it to explain formulas. In Section VI, we use our results on explainability to construct proofs for the Max-SAT problem and then we extend our results to weighted partial formulas in Section VII. Finally, we conclude and discuss future work in Section VIII.

II. Preliminaries

A. Definitions and Notations

Let X be the set of propositional variables and $n = |X|$ the number of variables. A literal l is a variable $x \in X$ or its negation $\overline{x}$. A clause c is a disjunction of literals $(l_1 \vee l_2 \vee \cdots \vee l_k)$. A formula in Conjunctive Normal Form (CNF) ϕ is a conjunction of clauses $\phi = c_1 \wedge c_2 \wedge \cdots \wedge c_m$. An assignment of variables $I : X \rightarrow \{true, false\}$ maps each variable to a boolean value. A literal l is satisfied (resp. falsified) by an assignment I if $l \in I$ (resp. $\bar{l} \in I$). A clause c is satisfied by an assignment I if at least one of its literals is satisfied by I, otherwise it is falsified by I. The empty clause $\square$ contains zero literals and is always falsified. A clause c subsumes a clause c' if each literal of c is a literal of c', i.e. $\forall l \in c,\ l \in c'$. A clause c opposes a clause c' if c contains a literal whose negation is in c', i.e. $\exists l \in c,\ \bar{l} \in c'$. We denote $var(c)$ and $var(\phi)$ the variables appearing respectively in the clause c and the formula ϕ. A CNF formula ϕ is satisfied by an assignment I, that we call model of ϕ, if each clause $c \in \phi$ is satisfied by I, otherwise it is falsified by I. The cost of an assignment I, denoted $cost_I(\phi)$, is the number of clauses falsified by I. For a given CNF formula ϕ, solving the Max-SAT problem consists

in determining the minimum number of falsified clauses in ϕ, which is the optimum of ϕ denoted $opt(\phi) = \min_I cost_I(\phi)$. We define the equivalence in the sense of Max-SAT as follows:

Definition 1 (Max-SAT Equivalence)**.** *We say that two CNF formulas ϕ and ϕ' are Max-SAT-equivalent and we denote $\phi \equiv \phi'$ if, for any assignment $I : var(\phi) \cup var(\phi') \rightarrow \{true, false\}$, we have $cost_I(\phi) = cost_I(\phi')$.*

B. Proof Systems for Max-SAT

In the last fifteen years, the study of inference rules for Max-SAT has led to major results in Max-SAT theory and solving. In particular, one of the first proof systems for Max-SAT is based on an inference rule called Max-SAT resolution [8], [9], [17], which is an extension of the resolution rule [27] introduced in the context of the satisfiability problem. Max-SAT resolution was shown to be sound for Max-SAT, i.e. it preserves Max-SAT equivalence. Furthermore, it is complete for the Max-SAT problem, i.e. it is sufficient to prove the optimum cost of a given CNF formula. Indeed, using Max-SAT resolution, we can exhibit a sequence of transformations from a given formula ϕ to an equivalent formula containing a satisfiable sub-formula and a multiset of empty clauses whose size is equal to the optimum of ϕ.

Definition 2 (Max-SAT resolution rule [8], [9], [17])**.** *Given two clauses $c_1 = x \vee a_1 \vee \cdots \vee a_s$ and $c_2 = \overline{x} \vee b_1 \vee \cdots \vee b_t$ with $A = a_1 \vee \cdots \vee a_s$ and $B = b_1 \vee \cdots \vee b_t$, the Max-SAT resolution rule replaces c_1 and c_2 by a set of new clauses where c_3 is the resolvent clause and $cc_1, \ldots, cc_{t+s}$ are compensation clauses:*

$$\frac{c_1 = x \vee A \qquad c_2 = \overline{x} \vee B}{\begin{array}{c} c_3 = A \vee B \\ cc_1 = x \vee A \vee \overline{b_1} \\ cc_2 = x \vee A \vee b_1 \vee \overline{b_2} \\ \cdots \\ cc_t = x \vee A \vee b_1 \vee \cdots \vee b_{t-1} \vee \overline{b_t} \\ cc_{t+1} = \overline{x} \vee B \vee \overline{a_1} \\ cc_{t+2} = \overline{x} \vee B \vee a_1 \vee \overline{a_2} \\ \cdots \\ cc_{t+s} = \overline{x} \vee B \vee a_1 \vee \cdots \vee a_{s-1} \vee \overline{a_s} \end{array}}$$

To construct complete proofs for the Max-SAT problem using Max-SAT resolution, it is possible to use the saturation algorithm [9]. The idea of this algorithm is to iteratively choose a variable and apply Max-SAT resolution on every pair of clauses $c_1 = x \vee A$ and $c_2 = \overline{x} \vee B$ until they are all opposed on at least one variable which is different from x. It was proved that the saturation algorithms allows to generate proofs for the Max-SAT problem in less than $n \times m \times 2^n$ inference steps. The saturation algorithm is mainly interesting for its theoretical result but it is also marginally used in Max-SAT solvers [4].

Example 1. *Given the formula $\phi = (\overline{x_1} \vee x_3) \wedge (x_1) \wedge (\overline{x_1} \vee x_2) \wedge (\overline{x_2} \vee \overline{x_3})$, the application of the saturation algorithm in the lexicographic order proves that $\phi \equiv \square \wedge (x_1 \vee \overline{x_2} \vee \overline{x_3}) \wedge (\overline{x_1} \vee x_2 \vee x_3)$ and is represented in Figure 1.*

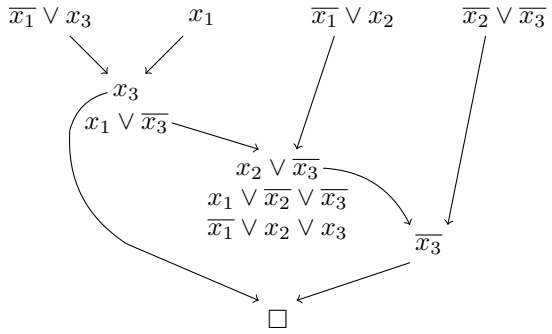

Fig. 1. Max-SAT resolution proof

Since its introduction, Max-SAT resolution has played a major role in Max-SAT theory and solving [6]. In particular, it is extensively used in the context of Branch and Bound algorithms (e.g. AHMAXSAT [2], MiniMaxSAT [13], AKMAXSAT [16], MAXSATZ [23], [21]) to transform inconsistent cores, mainly in the form of patterns [1], [18], [21]. Furthermore, the power of Max-SAT resolution and its impact on unit propagation were also extensively studied in this context [3], [10], [11]. Max-SAT resolution is also used in Core-Guided algorithms to transform cores returned by SAT solvers [24], [14]. In recent work, Max-SAT resolution was augmented with other rules such as the split rule defined below or the extension rule [19]. It was shown that the addition of such rules to Max-SAT resolution can improve its efficiency to generate shorter proofs for the Pigeonhole problem [20], [19] or allow, given a resolution refutation for SAT, to generate a Max-SAT refutation [25], [12], [26]. We must also mention that other Max-SAT proof systems exist like the Clause Tableau Calculus [22].

Definition 3 (Split)**.** *Given a clause $c_1 = (A)$ where A is a disjunction of literals and a variable x, the split rule replaces c_1 by two clauses $c_2 = (x \vee A)$ and $c_3 = (\overline{x} \vee A)$:*

$$\frac{c_1 = (A)}{c_2 = (x \vee A) \quad c_3 = (\overline{x} \vee A)}$$

In our work, we will mainly use a new proof system composed of the symmetric cut (also called the Almost Common Clause rule [5]) and the expansion rule (defined in Section V). Since these rules are strongly related to the split rule and Max-SAT resolution, we will mainly establish results for this proof system and provide insights for other proof systems using Max-SAT resolution instead of symmetric cut and/or using split instead of expansion. The symmetric cut that we define bellow is a special case of the Max-SAT resolution rule.

Definition 4 (Symmetric cut)**.** *Given two clauses $c_1 = x \vee A$ and $c_2 = \overline{x} \vee A$ where A is a disjunction of literals, the symmetric cut rule replaces c_1 and c_2 by $c_3 = (A)$:*

$$\frac{c_1 = (x \vee A) \quad c_2 = (\overline{x} \vee A)}{c_3 = (A)}$$

III. On the Explainability of Clauses and Formulas

In this section, we define the notion of explainable clauses and formulas. A clause is explainable if there exists a sequence of equivalence-preserving transformations, called explanation, that ensures the deduction of the given clause from an initial CNF formula. We start by giving the formal definition of an explainable clause. Then, we present two results in Propositions 1 and 2. The first establishes that subsumed clauses are explainable while the second stipulates that clauses which oppose every clause in the formula are unexplainable. Furthermore, we establish a characterization of explainable clauses in Theorem 1, a characterization that we will use later to propose a construction algorithm for such clauses. Finally, we generalize the notion of explainability on formulas and we characterize such formulas in Theorem 2.

Definition 5 (Explainable clause)**.** *Let ϕ be a CNF formula and c be a non-tautological clause, we say that c is explainable in ϕ if there exists a CNF formula ϕ' such that $\phi \equiv c \wedge \phi'$ otherwise we say that c is unexplainable in ϕ.*

Intuitively, explainable clauses represent information that can be deduced from a formula. If we exhibit an equivalence-preserving transformation T from ϕ to $c \wedge \phi'$, we say that T is an explanation of c in ϕ. Such a transformation represents the proof that the wanted information, i.e. a clause, can be inferred soundly from the formula. Furthermore, note that, for a CNF formula ϕ, any clause $c \in \phi$ is trivially explainable.

Definition 6 (Clause explanation)**.** *Let ϕ be a CNF formula and c be an explainable clause in ϕ. An explanation of c in ϕ is a Max-SAT equivalence-preserving transformation from ϕ to $c \wedge \phi'$ where ϕ' is a CNF formula.*

Now, we characterize explainable clauses in the following propositions and theorem in order to provide, in Section V, a method to construct explanations for such clauses or to refute their explainability. Proposition 1 ensures that subsumed clauses are explainable while Proposition 2 ensures that a clause opposed to each clause in the formula is unexplainable. These two simple results will pave the way for the characterization of explainable clauses in Theorem 1 which will be used to construct clause explanations in Section V.

Proposition 1. *Let ϕ be a CNF formula and c be a non-tautological clause. If $\exists c' \in \phi$ such that c' subsumes c then c is explainable in ϕ.*

Proof. We suppose that $\exists c' \in \phi$, c' subsumes c. Therefore, we have $c = c' \vee A$ where $A = a_1 \vee a_2 \vee \cdots \vee a_n$. Let $\phi_2 = \bigwedge_{1 \leq i \leq n} (c' \vee (\bigvee_{1 \leq j < i} a_j) \vee \overline{a_i})$. We prove that $c' \equiv \phi_2 \wedge c$. For any assignment I, we have two possible cases:

- Either I satisfies c' in which case there exists a literal $l \in c'$ which is satisfied by I. Since this literal is in every clause of $\phi_2 \wedge c$, we have $cost_I(\phi_2 \wedge c) = 0 = cost_I(c')$.
- Or, I falsifies c' in which case, one and exactly one clause of $\phi_2 \wedge c$ is falsified. Indeed, we consider the assignment of variables $var(a_1), ..., var(a_n)$ in I. By construction, exactly one clause of $\phi_2 \wedge c$ contains a clause opposed to each assignment of variables $var(a_1), ..., var(a_n)$ in I. This clause is falsified by I while the others are satisfied. Therefore, we have $cost_I(\phi_2 \wedge c) = 1 = cost_I(c')$.

Consequently, we have $\phi \equiv (\phi \backslash \{c'\}) \wedge c' \equiv (\phi \backslash \{c'\}) \wedge \phi_2 \wedge c \equiv \phi' \wedge c$ where $\phi' = (\phi \setminus \{c'\}) \wedge \phi_2$. We conclude that c is explainable in ϕ. ■

Proposition 2. *Let ϕ be a CNF formula and c be a non-tautological clause. If $\forall c' \in \phi$, c' opposes c, then c is unexplainable in ϕ.*

Proof. We suppose that $\forall c' \in \phi$, c' opposes c. We consider the assignment $I = \{\bar{l} \mid l \in c\}$. Clearly, if every clause in ϕ opposes c, I is a model of ϕ. Consequently, we have $cost_I(\phi) = 0$. And, by definition of I, c is falsified by I. Therefore, we have $cost_I(c) = 1$ and for each formula ϕ', $cost_I(c \wedge \phi') > cost_I(\phi)$. We conclude that c is unexplainable in ϕ. ■

Theorem 1. *Let ϕ be a CNF formula and c be a non-tautological clause. c is explainable in ϕ if and only if $\exists c' \in \phi$ such that c' doesn't oppose c and $\forall x \notin var(c)$, $c \vee x$ and $c \vee \overline{x}$ are both explainable in ϕ.*

Proof. First, suppose that c is explainable in ϕ, i.e. there exists a CNF formula ϕ' such that $\phi \equiv \phi' \wedge c$. By contraposition of Proposition 2, $\exists c' \in \phi$ such that c' doesn't oppose c. Now, let $x \notin var(c)$ and I be an assignment. As $cost_I(c) = cost_I((c \vee x) \wedge (c \vee \overline{x}))$ [20], we have, $\phi' \wedge c \equiv \phi' \wedge (c \vee x) \wedge (c \vee \overline{x})$ and we conclude that $\forall x \notin var(c)$, $c \vee x$ and $c \vee \overline{x}$ are both explainable in ϕ.

Conversely, suppose that $\exists c' \in \phi, c'$ doesn't oppose c and that $\forall x \notin var(c)$, $c \vee x$ and $c \vee \overline{x}$ are explainable in ϕ. We have two possible cases:

- Either $\nexists x \notin var(c)$, i.e. c contains all variables of ϕ. Also, we know that c' doesn't oppose c and, therefore, c' subsumes c. Finally, we deduce that c is explainable in ϕ by Proposition 1.
- Else, we consider the CNF formula M such that $\forall$ assignment I, the clause $(\bigvee_{l \in I} \bar{l})$ is duplicated $cost_I(\phi)$ times in M. By definition of M, for any assignment I, we have $cost_I(\phi) = cost_I(M)$ and thus $\phi \equiv M$. Intuitively, M is the maximal formula equivalent to ϕ. Similarly, we consider the maximal equivalent formulas for $(c \vee x)$ and $(c \vee \overline{x})$. We denote M_1 (resp. M_2) the CNF formula such that $\forall$ assignment I, the clause $(\bigvee_{l \in I} \bar{l})$ is duplicated $cost_I(c \vee x)$ (resp. $cost_I(c \vee \overline{x})$) times in M_1 (resp. M_2). By definition of M_1 (resp. M_2), we have $M_1 \equiv (c \vee x)$ (resp. $M_2 \equiv (c \vee \overline{x})$). Since $(c \vee x)$ (resp. $(c \vee \overline{x})$) is explainable in ϕ, we have $M_1 \subseteq M$ (resp. $M_2 \subseteq M$). Furthermore, as $(c \vee x)$ opposes $(c \vee \overline{x})$, we have $M_1 \cap M_2 = \emptyset$. We deduce that $\phi \equiv M \equiv (M \setminus (M_1 \cup M_2)) \wedge M_1 \wedge M_2 \equiv (M \backslash (M_1 \cup M_2)) \wedge (c \vee x) \wedge (c \vee \overline{x}) \equiv (M \backslash (M_1 \cup M_2)) \wedge c$. Therefore, c is explainable in ϕ.

We conclude that c is explainable in ϕ if and only if $\exists c' \in \phi$, c' doesn't oppose c and $\forall x \notin var(c)$, $c \vee x$ and $c \vee \overline{x}$ are both explainable in ϕ. ■

Next, we extend the notion of explainability to formulas. Notice that explaining a CNF formula ϕ_2 in a formula ϕ_1 can be seen as inferring ϕ_2 from ϕ_1. Consequently, two CNF formulas ϕ_1 and ϕ_2 are Max-SAT-equivalent if both formulas are mutually explainable.

Definition 7 (Explainable formula)**.** *Let ϕ_1 and ϕ_2 be two CNF formulas, we say that ϕ_2 is explainable in ϕ_1 if there exists a CNF formula ϕ' such that $\phi_1 \equiv \phi_2 \wedge \phi'$, otherwise we say that ϕ_2 is unexplainable in ϕ_1.*

Definition 8 (Formula explanation)**.** *Let ϕ_1 and ϕ_2 be two CNF formulas such that ϕ_2 is explainable in ϕ_1. An explanation of ϕ_2 in ϕ_1 is a Max-SAT equivalence-preserving transformation from ϕ_1 to $\phi_2 \wedge \phi'$ where ϕ' is a CNF formula.*

Intuitively, explaining a formula seems to be more difficult than explaining a clause. Indeed, if two clauses c_1 and c_2 are both explainable in a CNF formula, this does not ensure that $c_1 \wedge c_2$ is explainable too. However, we prove the following result to ensure that explaining a formula can be achieved by iteratively explaining each of its clauses.

Theorem 2. *Let ϕ_1 and $\phi_2 = c_1 \wedge ... \wedge c_m$ be two CNF formulas, ϕ_2 is explainable in ϕ_1 if and only if there exists a sequence of CNF formulas $\langle \phi'_1, ..., \phi'_m \rangle$ such that:*

- *There exists an explanation of c_1 from ϕ_1 to ϕ'_1.*
- *$\forall i \in \{2, ..., m\}$, there exists an explanation of c_i from $(\phi'_{i-1} \backslash \{c_{i-1}\})$ to ϕ'_i.*

Proof. Assume that ϕ_2 is explainable in ϕ_1. We set $\phi'_1 = \phi_2$ and $\forall i \in \{2, ..., m\}$, $\phi'_i = \phi'_{i-1} \backslash \{c_{i-1}\}$.

Conversely, by transitivity of $\equiv$, we have $\phi_1 \equiv \phi_2 \wedge (\phi'_m \backslash \{c_m\})$ and we conclude that ϕ_2 is explainable in ϕ_1. ■

As shown in Theorem 2, any method able to explain clauses can also be used to explain formulas. In Section V, we will propose an algorithm based on Theorem 1 to construct explanations. Then, this algorithm will be easily extended to formulas using the result established in Theorem 2. Inferential completeness [20] is strongly related to the notion of explanation. Naturally, a proof system is inferentially complete if it is possible to provide an explanation using its rules for every explainable clause. As stated in Proposition 3 below, Max-SAT resolution is not inferentially complete. We will therefore use a new proof system for explanation, called the Explanation Calculus (ExC), that we will define in the next section.

Proposition 3. *[20] Max-SAT resolution is not inferentially complete.*

IV. The Explanation Calculus

To achieve clause and formula explanation, we introduce, in this section, a new system called Explanation Calculus (ExC). This system is composed of two rules: the symmetric cut rule and the expansion rule. The expansion rule is defined below and can be considered as a Max-SAT adaptation of the weakening rule used in the context of SAT which replaces a clause by another subsuming it. Clearly, the weakening rule is not sound for Max-SAT and, in order to weaken a clause, it is necessary to introduce compensation clauses (similarly to Max-SAT resolution) to preserve Max-SAT equivalence. Expansion is also closely related to the split rule as we will show in Proposition 5. The motivation behind introducing the expansion rule is the result established in Proposition 1. Indeed, in its proof, we implicitly use expansion in order to prove the explainability of subsumed clauses. This result will be later explicitly stated in Proposition 10 in which we prove that the expansion rule suffices to build an explanation for subsumed clauses.

Definition 9 (Expansion)**.** *Given a clause $c = (A)$ where A is a disjunction of literals and $B = b_1 \vee ... \vee b_k$, the expansion rule applied on c and B replaces c by a set of new clauses:*

$$\frac{c_1 = (A)}{\begin{array}{c} cc_1 = (A \vee \overline{b_1}) \\ cc_2 = (A \vee b_1 \vee \overline{b_2}) \\ ... \\ cc_k = (A \vee b_1 \vee ... \vee b_{k-1} \vee \overline{b_k}) \\ c_2 = (A \vee B) \end{array}}$$

We denote k-expansion an application of the expansion rule on a clause c and a disjunction of literals B such that $|B| = k$. Now, we will study the relation between the expansion rule and the split rule in Propositions 4 and 5. Then, we will prove the soundness of the expansion rule in Proposition 6.

Proposition 4. *Each split step can be simulated using one 1-expansion step.*

Proof. If we consider one split step $(A) \vdash (x \vee A) \wedge (\overline{x} \vee A)$. We just have to set $B = x$ and the application of the expansion rule on clause (A) and literals B simulates the application of the split rule on clause (A) and on variable x. ■

Proposition 5. *Each k-expansion can be simulated using k split steps.*

Proof. Let $(A) \vdash (A \vee \overline{b_1}) \wedge ... \wedge (A \vee b_1 \vee ... \vee b_{k-1} \vee \overline{b_k}) \wedge (A \vee B)$ a k-expansion step. We apply the split rule k times to simulate the expansion step as below:

$$\begin{array}{rcl} (A) & \vdash & (A \vee \overline{b_1}) \wedge (A \vee b_1) \\ & \vdash & (A \vee \overline{b_1}) \wedge (A \vee b_1 \vee \overline{b_2}) \vee (A \vee b_1 \vee b_2) \\ & ... & \\ & \vdash & (A \vee \overline{b_1}) \wedge ... \wedge (A \vee b_1 \vee ... \vee b_{k-1} \vee \overline{b_k}) \wedge (A \vee B) \end{array}$$

■

Proposition 6. *The expansion rule is sound for Max-SAT.*

Proof. The split rule is sound for Max-SAT [20] and it can simulate the expansion rule as shown in Proposition 5. We conclude that the expansion rule is sound for Max-SAT. ■

To construct an explanation for any explainable clause or formula, we define below a proof system composed of the

symmetric cut and the expansion rules, called Explanation Calculus (**ExC**). Then, we prove in Proposition 7 that it is inferentially complete. Notice that, although this result is established for **ExC**, it remains valid for **Max-SAT resolution + expansion**, **symmetric cut + split** and **ResS** (**Max-SAT resolution + split**).

Definition 10 (Explanation Calculus)**.** *The Explanation Calculus (ExC) is composed of two rules: symmetric cut and expansion.*

Proposition 7. ***ExC*** *is inferentially complete.*

Proof. The proof system **ResS** has been proved inferentially complete in [20]. Furthermore, it was shown that **symmetric cut + split** simulates **ResS** in [7] and, therefore, it is also inferentially complete. Finally, since the expansion rule simulates the split rule, we also conclude that **ExC** is inferentially complete. ■

Now, we study the relation between **ExC** and other proof systems. It is important to note that in this paper, we are interested in the notion of inference in the context of Max-SAT on a wider scope, i.e. we want to study the general inferential power of proof systems and compare them in this sense and not just focus on their refutational power. Therefore, we provide a new definition for simulation below which we refer to as inferential simulation. It differs from the traditional definition of simulation [20], [15], which we can refer to as refutational simulation to avoid confusion, in the sense that it does not restrict the proofs to refutations. Clearly, if P_1 inferentially p-simulates P_2 then P_1 also refutationally p-simulate P_2 but the opposite is not necessarily true. We prove in Proposition 8 that **ExC** i-p-simulates **symmetric cut + split**, **Max-SAT resolution** and ResS [20]. Finally, we observe that it is possible to weaken Max-SAT resolution into the symetric-cut without loosing any power if we add expansion. This can be directly implied from Proposition 9 in which we establish that ExC is i-p-equivalent to **Max-SAT resolution + expansion**.

Definition 11 (inferential simulation)**.** *Let P_1 and P_2 be two proof systems. P_1 i-p-simulates (inferentially polynomially simulates) P_2 if there exists a polynomial computable function f such that for any proof Π deducing a clause c in P_2, $f(\Pi)$ is a proof deducing c in P_1. P_1 and P_2 are i-p-equivalent if P_1 i-p-simulates P_2 and P_2 i-p-simulates P_1.*

Proposition 8. ***ExC*** *i-p-simulates the following proof systems:*

- ***Max-SAT resolution,***
- ***symmetric cut + split***
- ***ResS***

Proof. A Max-SAT resolution step can be simulated using two expansion steps and one symmetric cut step as follows:

$(x \vee A) \vdash (x \vee A \vee \overline{b_1}) \wedge ... \wedge (x \vee A \vee B)$ [expansion]
$(\overline{x} \vee B) \vdash (\overline{x} \vee B \vee \overline{a_1}) \wedge ... \wedge (\overline{x} \vee B \vee A)$ [expansion]
$(x \vee A \vee B) \wedge (\overline{x} \vee B \vee A) \vdash (A \vee B)$ [symmetric cut]

Therefore, **ExC** i-p-simulates **Max-SAT Resolution**. Furthermore, as shown in Proposition 4, each split step can be simulated using one 1-expansion step and, consequently, **ExC** i-p-simulates **symmetric cut + split**. Finally, since one Max-SAT resolution step can be simulated by two expansions and one symmetric cut, and since one split step can be simulated by one expansion, we conclude that **ExC** i-p-simulates **ResS**. ■

Proposition 9. ***ExC*** *is i-p-equivalent to* ***Max-SAT resolution + expansion****.*

Proof. As shown in Proposition 8, **ExC** i-p-simulates **Max-SAT resolution** and, consequently, it also i-p-simulates **Max-SAT resolution + expansion**. Conversely, **Max-SAT resolution + expansion** trivially i-p-simulates **ExC** since they share the expansion rule and the symmetric cut rule is a specific case of Max-SAT resolution. ■

We summarize the results presented in Propositions 8 and 9 on the relation between **ExC** and other proof systems in Figure 2. In the next Section, we will show how it is possible to construct clause and formula explanations in **ExC**. This result is easily extendable for the following inferentially complete proof systems: **symmetric cut + split**, **ResS** and **Max-SAT resolution + expansion**.

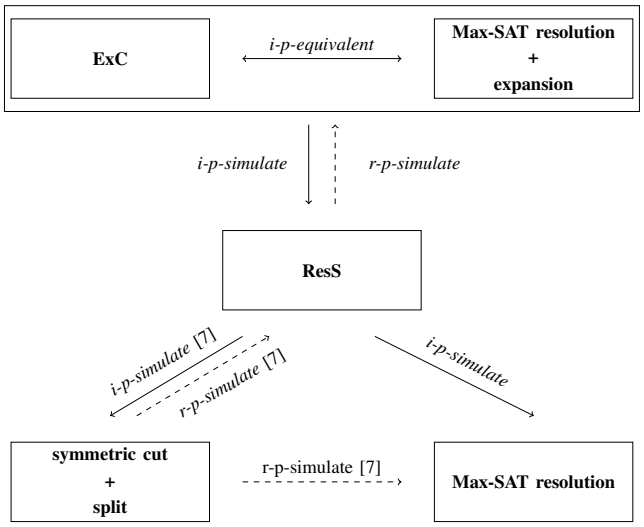

Fig. 2. Relationship between **ExC** and other proof systems

V. The Explanation Algorithm

In this section, we use the characterization in Theorem 1 to introduce a method to construct an explanation of a given clause, if possible. The intuition is that, given a CNF formula ϕ, we can exhibit an explanation of clause c by reverse inferring the sequence of transformations starting from c instead of ϕ. First, we prove an intermediate result which follows from the proof of Proposition 1.

Proposition 10. *Let ϕ be a CNF formula and c be a non-tautological clause. If $\exists c' \in \phi$ such that c' subsumes c then there exists an explanation of c in ϕ using the expansion rule.*

Proof. Suppose $\exists c' \in \phi$ such that c' subsumes c. The formula ϕ_2 in the proof of Proposition 1 corresponds, by definition, to an expansion step on clause c' with $B = c \setminus c'$. Therefore, this application of the expansion rule is an explanation of c in ϕ. ■

Now, we prove our main result in which we show that we can exhibit a transformation for any explainable clause in ExC and we provide a bound on the number of inference steps for such explanations.

Theorem 3. *Let ϕ be a CNF formula with n variables and c an explainable clause in ϕ. There exists an explanation of c in ϕ using ExC rules containing $O(2^n)$ inference steps.*

Proof. Suppose that c is explainable in ϕ. We have two cases:

- If $\exists c' \in \phi$ such that c' subsumes c then we can explain c by expansion of c' as proved in Proposition 10.
- Else, by Theorem 1, we know that $\exists c' \in \phi$ such that c' doesn't oppose c and, thus, that $\exists x \notin var(c)$. Since we also have $\forall x \notin var(c)$, $c \vee x$ and $c \vee \overline{x}$ are both explainable by Theorem 1, we can pick any variable $x \notin var(c)$ and recursively construct explanations T_1 and T_2 respectively for $c \vee x$ and $c \vee \overline{x}$. Notice that the recursion is guaranteed to stop because the size of clauses is bounded by n, in which case they are clearly subsumed. Therefore, we can provide a proof for c by applying the symmetric cut as follows $(c \vee x) \wedge (c \vee \overline{x}) \vdash c$. More precisely, if this application of the symmetric cut is denoted T_3 then $T = (T_1, T_2, T_3)$ is an explanation of clause c.

To explain c, we clearly have at most 2^n applications of the symmetric cut. Indeed, in the worst case, each clause c' including c has to be explained by two clauses $c' \vee x$ and $c' \vee \overline{x}$ (where x is a variable not appearing in c') until we obtain clauses of size n. In the worst case all the non-explained clauses that are subsumed have to be explained by the expansion of a clause in the formula, so we need at most one application of expansion per non-explained clause and hence at most 2^n expansions steps for the entire explanation. We conclude that if c is explainable then there exists an explanation of c in ExC in less than 2^{n+1} inference steps. ■

Notice how the previous results stay valid if we replace the symmetric cut rule by the Max-SAT resolution rule or the expansion rule by the split rule (in that case we need to add an additional factor of n to the bound). The proof of Theorem 3 can be described as an algorithm, that we refer to as the explanation algorithm, which can explain a clause in ExC or refute its explainability. We present it below where $T(\phi)$ denotes the application of the transformation T on the formula ϕ, $T_1.T_2$ denotes the concatenation of the transformations T_1 and T_2 and $expansion(c', c)$ denotes the application of the expansion rule on clause c' to get clause c.

Example 2. *We consider the CNF formula $\phi = (x_1 \vee x_2) \wedge (x_1 \vee \overline{x_2} \vee \overline{x_3}) \wedge (x_3) \wedge (\overline{x_1})$ and we want to explain $c = (x_1)$. We denote T and C respectively the sequence of transformations and the set of clauses that need to be explained. We construct the explanation of $c = (x_1)$, where variables are chosen in the lexicographic order, as follows:*

1) *$T = ()$ and $C = \{(x1)\}$*
   - *(x_1) is not subsumed by any clause of ϕ.*
   - *(x_1) is not opposed to all clauses of ϕ.*
   - *We pick the variable x_2 not appearing in (x_1). Now, we need to explain clauses $(x_1 \vee x_2)$ and $(x_1 \vee \overline{x_2})$. The application of the symmetric cut on these clauses is also added to T.*
2) *$T = (T_1)$ where $T_1 = (x_1 \vee x_2) \wedge (x_1 \vee \overline{x_2}) \vdash (x_1)$ and $C = \{(x_1 \vee x_2), (x_1 \vee \overline{x_2})\}$*
   - *Clause $(x_1 \vee x_2) \in \phi$ so it is trivially explained.*
3) *$T = (T_1)$ and $C = \{(x_1 \vee \overline{x_2})\}$*
   - *$(x_1 \vee \overline{x_2})$ is not subsumed by any clause of ϕ.*
   - *$(x_1 \vee \overline{x_2})$ is not opposed to all clauses of ϕ.*
   - *We pick the variable x_3 not appearing in $(x_1 \vee \overline{x_2})$. Now, we need to explain clauses $(x_1 \vee \overline{x_2} \vee x_3)$ and $(x_1 \vee \overline{x_2} \vee \overline{x_3})$. The application of the symmetric cut on these clauses is also added to T.*
4) *$T = (T_2, T_1)$ where $T_2 = (x_1 \vee \overline{x_2} \vee x_3) \wedge (x_1 \vee \overline{x_2} \vee \overline{x_3}) \vdash (x_1 \vee \overline{x_2})$ and $C = \{(x_1 \vee \overline{x_2} \vee x_3), (x_1 \vee \overline{x_2} \vee \overline{x_3})\}$*
   - *Clause $(x_1 \vee \overline{x_2} \vee x_3)$ is explained by expansion of $(x_3) \in \phi$ which subsumes it.*
5) *$T = (T_3, T_2, T_1)$ where $T_3 = (x_3) \vdash (\overline{x_1} \vee x_3) \wedge (x_1 \vee \overline{x_2} \vee x_3) \wedge (x_1 \vee x_2 \vee x_3)$ and $C = \{(x_1 \vee \overline{x_2} \vee \overline{x_3})\}$*
   - *Clause $(x_1 \vee \overline{x_2} \vee \overline{x_3}) \in \phi$ so it is trivially explained.*
6) *$T = (T_3, T_2, T_1)$ and $C = \emptyset$. Explanation of clause (x_1) is complete.*

We conclude that $\phi \vdash (\overline{x_1} \vee x_3) \wedge (x_1 \vee x_2 \vee x_3) \wedge (x_1) \wedge (\overline{x_1})$ by applying the transformation T, represented in Figure 3. Notice how the explanation T is inferred reversely. Clauses $(\overline{x_1} \vee x_3)$ and $(x_1 \vee x_2 \vee x_3)$ are compensation clauses, essential to preserve Max-SAT equivalence.

Algorithm 1 Explanation Algorithm

Require: clause c, CNF formula ϕ
Ensure: (R, T, ϕ') with $R \in \{\text{EXPLAINABLE}, \text{UNEXPLAINABLE}\}$, T the proof of explanability of c and $\phi' = T(\phi)$

```
if ∀c' ∈ φ : c' opposes c then
    return (UNEXPLAINABLE, ∅, φ)
if ∃c' ∈ φ : c' subsumes c then
    T ← expansion(c', c)
    return (EXPLAINABLE, T, T(φ))
Choose a variable x ∉ var(c)
(E1, T1, φ) ← Explain(c ∨ x, φ)
(E2, T2, φ) ← Explain(c ∨ ¬x, φ)
if E1 = UNEXPLAINABLE or E2 = UNEXPLAINABLE then
    return (UNEXPLAINABLE, ∅, φ)
T3 ← ((c ∨ x) ∧ (c ∨ ¬x) ⊢ c)
T ← T1.T2.T3
return (EXPLAINABLE, T, T3(φ))
```

Now that we can explain any clause (or refute its explainability), we will use the result of Theorem 2 to extend our

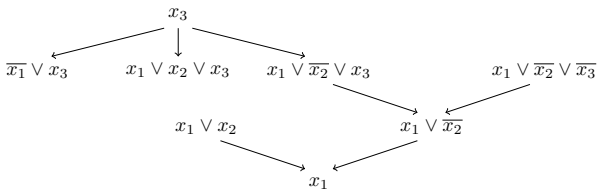

Fig. 3. Explanation of clause (x_1) in ϕ

algorithm to explain formulas. To this aim, we can iteratively explain each clause of the wanted formula as established in Theorem 2.

Theorem 4. *Let ϕ_1 and ϕ_2 be two CNF formulas with n variables such that ϕ_2 is explainable in ϕ_1 and let $m = |\phi_2|$. There exists an explanation of ϕ_2 in ϕ_1 using ExC rules containing $O(m \times 2^n)$ inference steps.*

Proof. We construct an explanation of ϕ_2 in ϕ_1 using the result of Theorem 2, i.e. explaining successively each clause of ϕ_2 in the current formula in which we eliminate progressively each explained clause. As shown in Theorem 3, explaining one clause of ϕ_2 costs at most $O(2^n)$ inference steps, and we have to explain m clauses. We conclude that the complete explanation of ϕ_2 in ϕ_1 contains at most $O(m \times 2^n)$ inference steps. ■

VI. Proofs for the Max-SAT problem

Now that we can explain any formula using the explanation algorithm, we want to use it to build proofs for the Max-SAT problem. Intuitively, the idea is to iteratively try to explain the empty clause until it is no longer explainable in the current formula. By doing it, we iteratively infer as many empty clauses as the optimum of the initial formula.

Theorem 5. *Let ϕ be a CNF formula with n variables and m clauses, we can deduce $\phi \vdash \underbrace{\square \wedge \cdots \wedge \square}_{opt(\phi)} \wedge \phi'$ where ϕ' is satisfiable using ExC rules in $O(m \times 2^n)$ inference steps.*

Proof. We build a complete proof for ϕ in several steps, each step being an explanation of the empty clause $\square$ which transforms the working formula ϕ_i (with $\phi_0 = \phi$) to an equivalent formula ϕ_{i+1} and an empty clause $\square$. Therefore, we have $\phi_0 \vdash \square \wedge \phi_1 \vdash \cdots \vdash \underbrace{\square \wedge \cdots \wedge \square}_{s} \wedge \phi_s$ $(s \geq 0)$ where the explanation algorithm fails to explain $\square$ in ϕ_s. Since the formula ϕ_s is satisfiable, we conclude that $s = opt(\phi)$.

By Theorem 3, we can provide an explanation for each empty clause in $O(2^n)$ inference steps. Notice that explaining an empty clause does not increase the number of variables of the formula. Furthermore, $opt(\phi)$, i.e. the minimum number of falsified clauses in ϕ is clearly bounded by m. Therefore, we construct at most m proofs of the empty clause as explained in the proof of Theorem 2. We conclude that $\phi \vdash \underbrace{\square \wedge \cdots \wedge \square}_{opt(\phi)} \wedge \phi'$ can be deduced in $O(m \times 2^n)$ ExC inference steps. ■

The result established in Theorem 5 shows that the Explanation Algorithm can be used to generate proofs for Max-SAT with a better bound on their size than the existing result using Max-SAT resolution which is in $O(m \times n \times 2^n)$ inference steps [9]. In the next section, we extend our results to weighted partial Max-SAT.

VII. Weighted Partial Max-SAT

In weighted partial Max-SAT, a finite or infinite weight is associated to each clause c, representing the penalty of falsifying it. Clauses with infinite weight are called hard clauses and must be satisfied while clauses with finite weight $w_c \in \mathbb{N}^*$ are called soft clauses. The cost of an assignment I is the sum of the weights of clauses falsified by I. Solving the weighted Max-SAT problem consists in determining the minimum cost over all possible assignments. The notion and characterization of explainable clauses and formulas are easily extendable to weighted partial clauses and formulas. We extend the proof systems studied in Section IV to weighted partial formulas by adding two rules to deal with weights, namely the unfold and fold rules which respectively allow to duplicate clauses by splitting their weights and to merge duplicated clauses by summing their weights. We will refer to the system **ExC + fold + unfold** as the Weighted Explanation Calculus (WExC).

Definition 12 (Fold & Unfold)**.** *Given a weighted clause c and two positive weights w_1 and w_2, the fold and unfold rules are respectively defined as follows:*

$$\frac{(c, w_1) \quad (c, w_2)}{(c, w_1 + w_2)} \qquad \frac{(c, w_1 + w_2)}{(c, w_1) \quad (c, w_2)}$$

To explain any soft clause (c, w), we ignore the weights in the formula and we try to explain c in its unweighted version. The obtained (unweighted) explanation can be easily adapted to the weighted case, possibly with a lower weight than w. In such case, we repeat the same treatment until we get the wanted weight. This result is established in Theorem 6. Remark that explaining a hard clause (c, ∞) is exactly the same as explaining the unweighted clause c in the unweighted set of the hard clauses. Hereafter, we consider w.l.o.g. formulas with finite weights.

Theorem 6. *Let ϕ be a weighted formula with n variables and (c, w) an explainable soft clause in ϕ. There exists an explanation of c in ϕ using WExC in $O(w \times 2^n)$ inference steps.*

Proof. We consider the unweighted version of the formula ϕ, i.e. $\phi' = \{c | (c, w) \in \phi\}$. Since (c, w) is explainable in ϕ, c is clearly explainable in ϕ'. We apply the explanation algorithm to get an explanation T of c in ϕ. Let ξ be the initial premises in T. We set $w' = \min\{w \mid (c, w) \in \phi \text{ and } c \in \xi\}$ and we apply the unfold rule on the clauses which correspond to ξ in ϕ, and then the weighted version of T (where the rules of ExC are exchanged with their weighted version) to get an explanation of (c, w'). Now, we have three possible cases:

- if $w' = w$, we have an explanation of (c, w) in ϕ.

- if $w' > w$, we simply apply one unfold step on (c, w') to get the explanation of (c, w) in ϕ.
- if $w' < w$, we repeat the previous procedure on $WT(\phi) \setminus \{(c, w')\}$, where WT is the (weighted) explanation of (c, w') in ϕ, until we explain clauses (c, w'_i) such that $1 \leq i \leq k$ and $\sum_{1 \leq i \leq k} w'_i \geq w$. Then, we merge these clause using k fold steps to get an explanation of $(c, \sum_{1 \leq i \leq k} w'_i)$ and we can thus refer to the previous two cases to get the full explanation of (c, w) in ϕ.

As shown in Theorem 3, each unweighted clause can be explained in $O(2^n)$ inference steps and we need at most $O(2^n)$ applications of the unfold rule to get the weighted explanation. This is repeated k times where k is clearly bounded by w. Therefore, we conclude that there exists an explanation of (c, w) in ϕ using WEXC in $O(w \times 2^n)$ inference steps. ■

Theorem 7. *Let ϕ_1 and ϕ_2 be two weighted formulas with n variables such that ϕ_2 is explainable in ϕ_1 and let $m = |\phi_2|$ and $w = \max\{w' \mid (c, w') \in \phi_2$ and $w' \neq \infty\}$. There exists an explanation of ϕ_2 in ϕ_1 using WExC containing $O(m \times w \times 2^n)$ inference steps.*

Proof. We iteratively explain each of the clauses of ϕ_2. Each clause can be explained in $O(w \times 2^n)$ inference steps as shown in Theorem 6 and we have m clauses to explain. We conclude that there exists an explanation of ϕ_2 in ϕ_1 using WExC containing $O(m \times w \times 2^n)$ inference steps. ■

Definition 13 (roof)**.** *Let ϕ be a weighted formula. The roof of ϕ, denoted $rf(\phi)$, is defined as follows:*

$$rf(\phi) = \sum_{(c,w) \in \phi \mid w \neq \infty} w$$

Theorem 8. *Let ϕ be a weighted formula with n variables and m clauses, we can deduce $\phi \vdash (\square, opt(\phi)) \wedge \phi'$ where ϕ' is satisfiable using WExC in $O(rf(\phi) \times 2^n)$ inference steps.*

Proof. We iteratively explain a new empty clause until it is no longer possible and we merge these empty clauses into one clause $(\square, opt(\phi))$. The number of empty clauses $opt(\phi)$ is bounded by $rf(\phi)$ and each of the empty clauses $(\square, 1)$ can be explained in $O(2^n)$ inference steps as shown in Theorem 6. Consequently, we can build a proof for the weighted Max-SAT problem with $O(rf(\phi) \times 2^n)$ inference steps. ■

VIII. Conclusions and Future Work

This paper focuses on inference in Max-SAT. In particular, we introduced the notion of explainable clauses (and formulas) which can be deduced from a given formula by applying Max-SAT-equivalent transformations. Since Max-SAT resolution is not inferentially complete, we defined a new proof system called ExC which is inferentially complete and we compared it to other proof systems. We presented a procedure, called the explanation algorithm, to explain any clause or to refute its explainability using ExC. We extended the explanation algorithm to explain any formula or refute its explainability and to construct proofs for the Max-SAT problem with a better bound compared to Max-SAT resolution. Finally, we explained how to adapt our work to weighted partial formulas.

References

[1] André Abramé and Djamal Habet. On the Extension of Learning for Max-SAT. In *Proceedings of the 7th European Starting AI Researcher Symposium (STAIRS)*, volume 241, pages 1–10, 2014.

[2] André Abramé and Djamal Habet. Ahmaxsat: Description and Evaluation of a Branch and Bound Max-SAT Solver. *Journal on Satisfiability, Boolean Modeling and Computation*, 9:89–128, 2015.

[3] André Abramé and Djamal Habet. On the Resiliency of Unit Propagation to Max-Resolution. In *IJCAI*, pages 268–274, 2015.

[4] Josep Argelich, Chu Min Li, and Felip Manyà. A Preprocessor for Max-SAT Solvers. In *SAT*, pages 15–20, 2008.

[5] Nikhil Bansal and Venkatesh Raman. Upper Bounds for MaxSat: Further Improved. In *International Symposium on Algorithms and Computation (ISAAC)*, volume 1741, pages 247–258, 01 1999.

[6] Armin Biere, Marijn Heule, Hans van Maaren, and Toby Walsh, editors. *Handbook of Satisfiability*, volume 185 of *Frontiers in Artificial Intelligence and Applications*. IOS Press, 2009.

[7] Maria Luisa Bonet and Jordi Levy. Equivalence Between Systems Stronger Than Resolution. In *SAT*, pages 166–181, 2020.

[8] Maria Luisa Bonet, Jordi Levy, and Felip Manyà. A complete calculus for MAX-SAT. In *SAT*, volume 4121, pages 240–251, 08 2006.

[9] María Luisa Bonet, Jordi Levy, and Felip Manyà. Resolution for Max-SAT. In *Artificial Intelligence*, volume 171, pages 606–618, 2007.

[10] Mohamed Sami Cherif and Djamal Habet. Towards the Characterization of Max-Resolution Transformations of UCSs by UP-Resilience. In *CP*, pages 91–107, 2019.

[11] Mohamed Sami Cherif, Djamal Habet, and André Abramé. Understanding the power of Max-SAT resolution through UP-resilience. *Artificial Intelligence*, 289:103397, 2020.

[12] Yuval Filmus, Meena Mahajan, Gaurav Sood, and Marc Vinyals. MaxSAT Resolution and Subcube Sums. In *SAT*, pages 295–311, 2020.

[13] Federico Heras, Javier Larrosa, and Albert Oliveras. MiniMaxSAT: An efficient weighted Max-SAT solver. *Journal of Artificial Intelligence Research*, 31:1–32, 2008.

[14] Federico Heras and Joao Marques-Silva. Read-Once resolution for Unsatisfiability-Based Max-SAT Algorithms. In *IJCAI*, pages 572–577, 2011.

[15] Philipp Hertel, Fahiem Bacchus, Toniann Pitassi, and Allen Van Gelder. Clause learning can effectively p-simulate general propositional resolution. In *AAAI*, pages 283–290, 2008.

[16] Adrian Küegel. Improved exact solver for the weighted max-sat problem. In *POS-10. Pragmatics of SAT*, EPiC Series in Computing, pages 15–27, 2012.

[17] Javier Larrosa and Federico Heras. Resolution in Max-SAT and its relation to local consistency in weighted CSPs. In *IJCAI*, pages 193–198, 2005.

[18] Javier Larrosa, Federico Heras, and Simon De Givry. A logical approach to efficient Max-SAT solving. *Artificial Intelligence*, 172:204–233, 2008.

[19] Javier Larrosa and Emma Rollon. Augmenting the Power of (Partial) MaxSat Resolution with Extension. In *AAAI*, pages 1561–1568, 2020.

[20] Javier Larrosa and Emma Rollon. Towards a Better Understanding of (Partial Weighted) MaxSAT Proof Systems. In *SAT*, pages 218–232, 2020.

[21] Chu Min Li, Felip Manyà, and Jordi Planes. New Inference Rules for Max-SAT. *Journal of Artificial Intelligence Research*, 30:321–359, 2007.

[22] Chu-Min Li, Felip Manyà, and Joan Ramon Soler. A Clause Tableau Calculus for MaxSAT. In *IJCAI*, page 766–772, 2016.

[23] Chu-Min Li, Felip Manyà, and Jordi Planes. Detecting Disjoint Inconsistent Subformulas for Computing Lower Bounds for Max-SAT. In *AAAI*, pages 86–91, 2006.

[24] Nina Narodytska and Fahiem Bacchus. Maximum satisfiability using core-guided MaxSAT resolution. In *AAAI*, pages 2717–2723, 2014.

[25] Matthieu Py, Mohamed Sami Cherif, and Djamal Habet. Towards Bridging the Gap Between SAT and Max-SAT Refutations. In *ICTAI*, pages 137–144, 2020.

[26] Matthieu Py, Mohamed Sami Cherif, and Djamal Habet. A Proof Builder for Max-SAT. In *SAT*, volume 12831, pages 488–498, 2021.

[27] John Alan Robinson. A Machine-Oriented Logic Based on the Resolution Principle. *Journal of the Association for Computing Machinery*, 12:23–41, 1965.