Towards Bridging the Gap Between SAT and Max-SAT Refutations

Matthieu Py, Mohamed Cherif, Djamal Habet

Towards Bridging the Gap Between SAT and Max-SAT Refutations

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*—Adapting a resolution proof for SAT to a Max-SAT resolution proof without increasing considerably the size of the proof is an open question. This paper contributes to this topic by exhibiting linear adaptations, in terms of the input SAT proof size, in restricted cases which are regular tree resolution refutations, tree resolution refutations and a new introduced class of refutations that we refer to as semi-tree resolution refutations. We also extend these results by proposing a complete adaptation for any unrestricted SAT refutation to a Max-SAT refutation, which is exponential in the worst case.**

***Index Terms*—Resolution, Max-SAT Resolution, Refutation, Regular Resolution, Tree Resolution**

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, while the SAT problem asks for the existence of an assignment which satisfies all the clauses. A well-known proof system for Max-SAT is Max-SAT resolution [20] which extends the resolution rule [24] used in the context of SAT. Max-SAT resolution plays a prominent role in Max-SAT as it is the most studied inference rule, both in theory and practice [1], [2], [7], [18], [20], [22].

In the context of SAT, an unsatisfiable formula can be refuted with a sequence of resolution steps which leads to the empty clause. Sequences of Max-SAT resolution steps are more constrained than sequences of resolution steps. Indeed, resolution adds the conclusion to the premises whereas the premise clauses are replaced by the conclusions when applying Max-SAT resolution. Switching from a read-once resolution proof, where each clause is used once, to a Max-SAT resolution proof is possible and well-known [11]. However, the adaptation of any resolution proof to a Max-SAT resolution one, especially in the context of a refutation, is an established problem. Bonet et al. state that "it seems difficult to adapt a classical resolution proof to get a Max-SAT resolution proof, and it is an open question if this is possible without increasing substantially the size of the proof" [20].

This paper attempts to contribute to this open question on refutation proofs by proposing a way to deal with non-read-once clauses, i.e. clauses used several times as a premise of a resolution step. Indeed, to adapt any resolution refutation, it is necessary to duplicate the non-read-once clauses in some form while also preserving Max-SAT equivalence. To this end, we augment Max-SAT resolution with a simple split rule which allows to generate two clauses subsumed by the original clause. Intuitively, applying the split rule on a non-read-once clause will duplicate it since only literals that will not affect the rest of the proof will be added.

Accordingly, we deal first with regular tree resolution refutations [4], [26], showing that a linear adaptation of a SAT refutation to a Max-SAT one is possible in this case. Then, we extend this result to tree resolution refutations using a known result in [25] which stipulates that a minimal tree resolution refutation is regular. Furthermore, we introduce a new class of refutations that we refer to as semi-tree-like, which is a generalization of tree resolution refutations, and we extend our linear result to this class of refutations. Finally, we propose a complete adaptation of any (or unrestricted) resolution refutation to a Max-SAT refutation, although with an exponential factor, using the fact that any resolution refutation can be made tree-like with an exponential cost.

This paper is organized as follows. Section II gives some necessary definitions and notations. Sections III to VI describe the above contributions in detail. Finally, we conclude and discuss future work in Section VII.

II. Preliminaries

A. Definitions and Notations

Let X be the set of propositional variables. A literal l is a variable $x \in X$ or its negation $\overline{x}$. A clause c is a disjunction (or a set) of literals $(l_1 \vee l_2 \vee ... \vee l_k)$. A formula in Conjunctive Normal Form (CNF) ϕ is a conjunction (or a multiset) of clauses $\phi = c_1 \wedge c_2 \wedge ... \wedge c_m$. An assignment $I : X \to \{true, false\}$ maps each variable to a boolean value and can be represented as a set of literals. 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 $\Box$ contains zero literals and is always falsified. A clause c opposes a clause c' if c contains a literal whose negation is in c', i.e. $\exists l \in c,\ \bar{l} \in c'$. A clause c subsumes a clause c' if each literal of c is a literal of c', i.e. $\forall l \in c,\ l \in c'$. We denote $var(c)$ the variables appearing in the clause c. 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. Solving the Satisfiability (SAT) problem consists in determining whether there exists an assignment I that satisfies a given CNF formula ϕ. In the case where such an assignment exists, we say that ϕ is satisfiable, otherwise we say that ϕ is unsatisfiable or inconsistent. The cost of an assignment I, denoted $cost_I(\phi)$, is the number of clauses falsified by I. The Maximum Satisfiability (Max-SAT) problem is an optimization extension of SAT which, for a given CNF formula ϕ, consists in determining the maximum number of clauses that can be satisfied by an assignment of ϕ. Equivalently, it consists in determining the minimum number of clauses that each assignment must falsify, i.e. $\min_I \; cost_I(\phi)$.

B. Resolution Refutations in SAT

To certify that a CNF formula is satisfiable, it is sufficient to simply exhibit a model of the formula. On the other hand, to prove that a CNF formula is unsatisfiable, we need to refute the existence of a model. To this end, we can exhibit a SAT refutation which consists of a sequence of equivalence-preserving transformations (in the sense of SAT as defined below) starting from the formula and ultimately deducing an empty clause.

Definition 1 (SAT Equivalence)**.** *Let ϕ and ϕ' be two CNF formulas. We say that ϕ is equivalent (in the sense of SAT) to ϕ' if for any assignment $I : var(\phi) \cup var(\phi') \to \{true, false\}$, I is a model of ϕ if and only if I is a model of ϕ'.*

A well-known SAT refutation system is based on an inference rule for SAT called resolution [24]. Refutations in this system are referred to as resolution refutations. The resolution rule, defined below, deduces a clause called resolvent from two opposed clauses which can be added to the formula while preserving SAT equivalence. Resolution plays an important role in the context of Conflict Driven Clause Learning (CDCL) [21]. Furthermore, it was shown that CDCL can polynomially simulate general resolution [23]. As showcased in Example 1, a resolution proof can be represented as a Directed Acyclic Graph (DAG) whose nodes are clauses in the proof either having two or zero incoming arcs (resp. if they are resolvents or clauses of the initial formula).

Definition 2 (Resolution [24])**.** *Given two clauses $c_1 = (x \vee A)$ and $c_2 = (\overline{x} \vee B)$, the resolution rule is defined as follows:*

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

Example 1. *We consider the CNF formula $\phi = (\overline{x_1} \vee x_3) \wedge (x_1) \wedge (\overline{x_1} \vee x_2) \wedge (\overline{x_2} \vee \overline{x_3})$. A resolution refutation of ϕ is represented as a DAG in Fig. 1.*

Many restricted classes of resolution refutations have been studied in the literature namely linear resolution [19], unit resolution [12], input resolution [12], regular resolution [26], read-once resolution [14] and tree (or tree-like) resolution [4] refutations among others. In particular, a resolution refutation is tree-like if every intermediate clause, i.e. resolvent, is used at most once in the proof. It is known that DPLL algorithms [8] on unsatisfiable instances correspond to tree resolution refutations [9]. Similarly, a resolution refutation is read-once if each clause is used at most once in the proof. Clearly, read-once resolution refutations are also tree-like since they form a restricted class of tree resolution refutations. It was shown in [14] that there exists unsatisfiable CNF formulas which cannot be refuted using read-once resolution. Finally, a resolution is regular if every variable is resolved on at most once in each branch of the DAG, i.e. path from a clause of the initial formula to the empty clause. It was shown that CDCL without restarts can polynomially simulate regular resolution [6]. We say that an irregularity is a sequence of clauses (each clause must be deduced using the previous one as premise) such that the first clause and the last one contain a literal l but at least one of the intermediate clauses does not contain this literal l. In other words, an irregularity is a certificate that a resolution refutation is not regular.

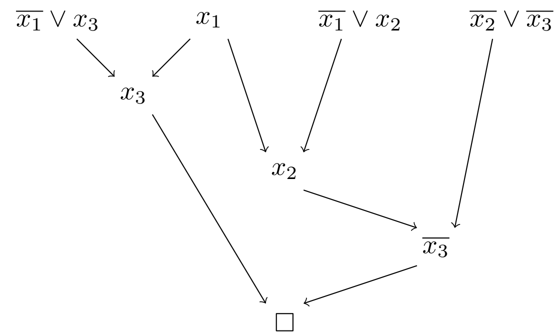

Fig. 1. Resolution refutation

Example 2. *We consider the refutation of ϕ in Example 1. The refutation is clearly tree-like but it is not read-once since clause (x_1) is used two times as a premise of a resolution step. The refutation is also regular as every variable is resolved on at most once in every branch of the DAG in Fig. 1.*

C. Max-SAT Refutations

Several complete proof systems for Max-SAT were introduced in the litterature, namely the Max-SAT resolution Calculus in [20] and the Clause Tableau Calculus in [17]. In particular, Max-SAT resolution, one of the first known complete systems for Max-SAT, was inspired from Resolution. The aim of complete Max-SAT systems is not to refute the formula per se but to compute the Max-SAT optimum of a given CNF formula, i.e. the maximum number of falsified clauses. The formula is thus refuted as many times as its optimum through equivalence-preserving transformations in the sense of Max-SAT as defined below.

Definition 3 (Max-SAT Equivalence)**.** *Let ϕ and ϕ' be two CNF formulas. We say that ϕ is equivalent (in the sense of Max-SAT) to ϕ' if for any assignment $I : var(\phi) \cup var(\phi') \to \{true, false\}$, we have $cost_I(\phi) = cost_I(\phi')$.*

The Max-SAT resolution proof system relies on an inference rule that extends resolution for Max-SAT. Other than the resolvent clause, this rule, called Max-SAT resolution and defined below, introduces new clauses referred to as compensation clauses essential to preserve Max-SAT equivalence. As a sound and complete rule for Max-SAT [20], Max-SAT resolution plays an important role in the context of Max-SAT theory and solving. In particular, it is extensively used in the context of Branch and Bound algorithms for Max-SAT to transform inconsistent subsets [2], [15], [18] as well as in the context of SAT-based algorithms to transform cores returned by SAT oracles [11], [22]. For a given CNF formula, it is possible to generate a Max-SAT resolution proof of its optimum by applying the saturation algorithm [20] to deduce empty clauses. As showcased in Example 3, a Max-SAT resolution proof can also be represented as a DAG whose nodes are multisets of clauses either having two or zero incoming arcs (resp. if they are clauses produced by a Max-SAT resolution step or clauses of the initial formula.

Definition 4 (Max-SAT resolution [5], [16], [20])**.** *Given two clauses $c_1 = x \vee A$ and $c_2 = \overline{x} \vee B$ with $A = a_1 \vee ... \vee a_s$ and $B = b_1 \vee ... \vee b_t$. The Max-SAT resolution rule is defined as follows:*

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

where c_r is the resolvent clause and $cc_1, ..., cc_{t+s}$ are compensation clauses.

Remark 1. *Unlike resolution, the Max-SAT resolution rule replaces the premises by the conclusions.*

Example 3. *We consider the CNF formula from Example 1. A hand-made Max-SAT resolution refutation of ϕ was proposed in [20] and is represented in Fig. 2.*

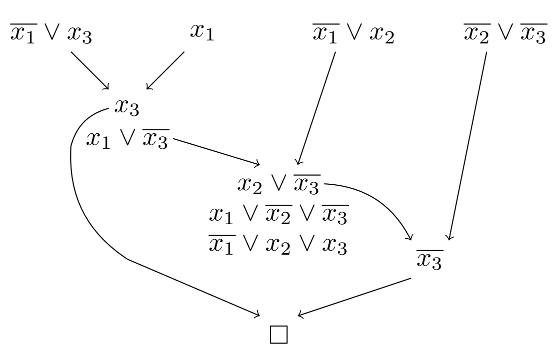

Fig. 2. A Max-SAT resolution proof

In this paper, we also augment Max-SAT resolution with the split rule defined below. Intuitively, this rule allows to duplicate a clause by adding one literal.

Definition 5 (Split rule)**.** *Given a clause $c_1 = (A)$ where A is a disjunction of literals and x a variable, the Max-SAT split rule is defined as follows:*

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

Remark 2. *Like the Max-SAT resolution rule, the split rule replaces the premise by the conclusions.*

Finally, we choose to maintain the designation 'refutation' in the context of Max-SAT. A Max-SAT refutation (or max-refutation) will thus consist of a sequence of Max-SAT preserving transformations, namely Max-SAT resolutions and splits in this paper, allowing to deduce an empty clause from a given unsatisfiable formula. The size of a (SAT or Max-SAT) refutation is the number of its inference steps.

III. From Regular Tree Resolution Refutations to Max-Refutations

In this section, we show how it is possible to adapt a regular tree resolution refutation to obtain a max-refutation with linear size. If a clause c is used k times ($k > 1$) as a premise of a resolution step, we use the split rule to duplicate clause c into k distinct clauses subsumed by c. We will then use these new clauses to replace c as a premise of a resolution step. Given a branch starting from a clause c, we say that this branch accepts the substitution of c by $c \vee l$ if updating the branch after the substitution of c by $c \vee l$ does not affect the validity of the resolution refutation. The following lemma guarantees that, for a given non-read-once clause, there exists a variable x such that some branches starting from c accept the substitution of c by $c \vee x$ while the rest accept the substitution of c by $c \vee \overline{x}$.

Lemma 1. *Given a non-read-once regular tree resolution refutation P and a non-read-once clause c in P, there exists a variable $x \notin var(c)$ such that it is possible to partition the branches starting from c into two non-empty subsets of branches, the branches in the first subset accepting the substitution of c by $c \vee x$ and the branches in the second accepting the substitution of c by $c \vee \overline{x}$.*

Proof. Let P a non-read-once regular tree resolution refutation and c a non-read-once clause in P. There exists a node v of the DAG of P representing a resolution step on variable x such that v is the first junction point of all the paths starting from c. The existence is ensured since this junction point is eventually the empty clause. Furthermore, every path starting from the clause c leads to one (and only one) of the premises of the resolution step in the node v. Indeed, a path leading to both premises entails the existence of an intermediate non-read-once clause which is not possible since the refutation is tree-like. We partition the branches starting from c into two subsets containing respectively the paths leading to the first and second premise of the resolution step in the node v. Each partition is non empty since if there exists an empty subset v can't be the first junction point of the branches. Let x be the variable eliminated at this resolution step and suppose w.l.o.g that the first premise contains literal x while the second contains literal $\overline{x}$. As P is regular, x is not a variable of c and the subset of branches starting from c leading to the first premise accepts the substitution of c by $c \vee x$ while the subset of branches leading to the second premise accepts the substitution of c by $c \vee \overline{x}$.

■

The result established in Lemma 1 ensures the possibility to fix any non-read-once clause used $k > 1$ times by using the split rule. Indeed, we can apply this rule to replace a non-read-once clause used $k > 1$ times by two clauses used respectively $1 \leq k_1 < k$ and $1 \leq k_2 < k$ such that $k = k_1 + k_2$. By iterating this method, we can fix every non-read-once clause. Then, we only need to replace the resolution rule by the Max-SAT resolution rule to obtain an adaptation from any regular tree resolution refutation to a max-refutation in linear size.

Theorem 1. *Given an unsatisfiable formula ϕ and a regular tree resolution refutation P of ϕ, there exists a max-refutation of ϕ containing $O(|P|)$ inference steps.*

Proof. Let P be a regular tree resolution refutation of ϕ. We set $T_1 = \emptyset$ and $T_2 = MR(P)$, where $MR(P)$ is obtained from P after replacing each resolution by Max-SAT resolution. If P is read-once, T_2 is a max-refutation of ϕ containing $|P|$ inference steps (which is obviously in $O(|P|)$). Now, let c be a non-read-once clause of P. Using Lemma 1, there exists a variable $x \notin var(c)$ and a partition of the branches starting from c into two non-empty subsets, the first accepting $c \vee x$ and the second accepting $c \vee \overline{x}$. We apply the Max-SAT split rule on c to obtain $c \vee x$ and $c \vee \overline{x}$ and we replace c as premise by $c \vee x$ on the first subset of branches and c by $c \vee \overline{x}$ on the second. Doing this, we augment T_1 by adding one split and we change T_2 by replacing the premise clause c as described above. As T_2 is a tree-like regular resolution refutation of $(\phi \backslash c) \wedge (c \vee x) \wedge (c \vee \overline{x})$, it is possible to iteratively apply this operation on T_2 until we obtain a read-once regular tree resolution refutation. Therefore, after the last iteration, we have a couple (T_1, T_2) such that T_1 is a sequence of applications of the split rule transforming ϕ into a Max-SAT equivalent ϕ' and T_2 is a read-once regular max-refutation of ϕ'. Therefore, these transformations form a max-refutation of ϕ.

To prove that the size of the max-refutation is in $O(|P|)$, we first consider how to fix a leaf clause of P (i.e. how to replace it by read-once clauses). If c is a leaf clause of P used k times, we prove by induction on k that it is possible to fix this clause using at most $k - 1$ splits:

- If $k = 1$, we clearly need 0 splits to fix the read-once clause c.
- Suppose that the assertion is true for any $k' < k$ and let c be a clause used k times. Using Lemma 1, it is possible to use 1 split to replace c by two clauses c_1 and c_2 respectively used k_1 and k_2 times with $k_1, k_2 > 0$ and $k_1 + k_2 = k$. Using our assertion for k_1 and k_2, it is possible to fix c_1 with at most $k_1 - 1$ splits and c_2 with at most $k_2 - 1$ splits. Therefore, it is possible to fix c with at most $1 + (k_1 - 1) + (k_2 - 1) = k - 1$ splits.

Let $c_1, ..., c_p$ be the leaf clauses of P used respectively $k^1, k^2, ..., k^p$ times. Notice that $k^1 + k^2 + ... + k^p = |P| + 1$ since P has exactly $2|P|$ premises, i.e. uses of clauses, and $|P| - 1$ intermediate clauses (the empty clause is not used and we neglect the trivial cases where a non-empty intermediate clause is not used and where the proof produces several empty clauses). Using the previous induction, we need at most $k^1 - 1 + k^2 - 1 + ... + k^p - 1 \leq |P|$ splits to fix every non-read-once leaf clause of P. Consequently, $|T_1| \leq |P|$. On the other hand, the number of Max-SAT resolutions in T_2 is by construction equal to the number of resolution steps in P and, therefore, $|T_2| = |P|$. We conclude that the complete max-refutation contains at most $2|P|$ inference steps, which is in $O(|P|)$.

■

Example 4. *We consider the regular tree resolution refutation from Example 1 represented by the DAG in Fig. 1. We observe that the original clause (x_1) is used two times as a premise of a resolution step. The junction point of the left and right branches eliminates variable x_3 such that the branch on the left leads to the premise containing literal x_3 and the branch on right leads to the premise containing literal $\overline{x_3}$. We apply the split rule on clause (x_1) to get $(x_1 \vee x_3)$ and $(x_1 \vee \overline{x_3})$ and we replace (x_1) by $(x_1 \vee x_3)$ and $(x_1 \vee \overline{x_3})$ respectively on the left and right branches. Finally, we replace all resolutions by Max-SAT resolutions to obtain the complete max-refutation.*

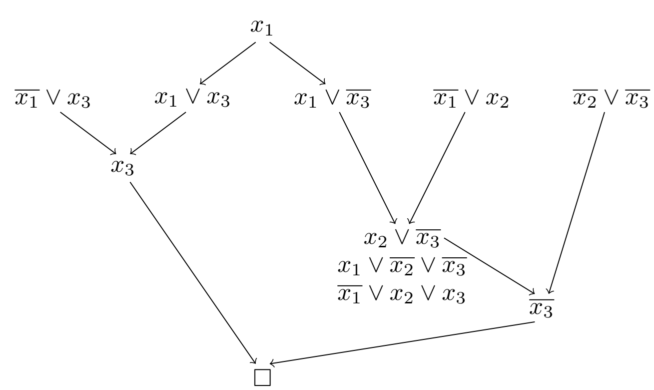

Fig. 3. Applying the split rule to deal with a non-read once clause

IV. From Tree Resolution Refutations to Max-Refutations

In the previous section, we proposed a linear adaptation from regular tree resolution refutations to max-refutations. We propose in this section to extend the case where this adaptation guarantees linear size of the obtained max-refutation to tree resolution refutations. To this end, we simply exhibit a known transformation from any tree resolution refutation to a regular tree resolution refutation without increasing its size. This result was proved in [25] in the form of the following lemma (cf. Lemma 5.1 in [25]). The proof relies on a transformation which consists in iteratively discarding the first resolution in the case of an irregularity and updating the rest of the resolution proof accordingly, potentially discarding other resolution steps which are no longer necessary.

Lemma 2. *[25] A tree resolution refutation of minimal size is regular.*

Example 5. *We consider the tree resolution refutation represented in Fig. 5. This refutation is not regular since x_1 is eliminated two times in the same branch. As shown in Fig. 5, this refutation can be minimized and thus made regular by discarding the first resolution on variable x_1 in the irregularity and updating the rest of the proof, Notice that after the transformation, clauses $(\overline{x_1} \vee \overline{x_3})$ and (x_3) are no longer used in the refutation.*

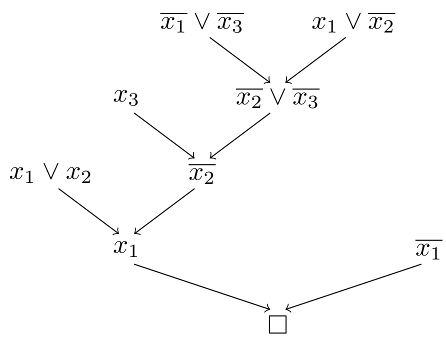

Fig. 4. Tree resolution refutation containing an irregularity on variable x_1

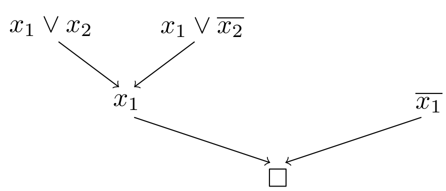

Fig. 5. Regular tree resolution refutation after minimization

Since it is possible to make a tree resolution refutation regular without increasing the size of the proof, we can apply the adaptation in Theorem 1 to produce a max-refutation with linear size as shown in the following corollary.

Corollary 1. *Given an unsatisfiable formula ϕ and a tree resolution refutation P of ϕ, there exists a max-refutation of ϕ containing $O(|P|)$ inference steps.*

Proof. Using Lemma 2, there exists a regular tree resolution refutation P_2 such that $|P_2| = O(|P|)$. By applying Theorem 1, we obtain a max-refutation containing $O(|P_2|) = O(|P|)$ inference steps.

■

V. FROM SEMI-TREE RESOLUTION REFUTATIONS TO MAX-REFUTATIONS

In Section IV, we proposed a linear adaptation from any tree resolution refutation to a max-refutation. We propose in this section to extend this linear result to semi-tree resolution refutations defined below. As shown in Proposition 1, this class of refutations extends tree resolution refutations, i.e. every tree resolution refutation is semi-tree-like.

Definition 6 (semi-tree resolution refutation)**.** *A resolution refutation is semi-tree-like if, for any branch of the refutation, at most one clause is non-read-once.*

Example 6. *We consider the resolution refutation P in Fig. 6. P is clearly semi-tree-like since in each branch at most one clause is non-read-once. Notice also that P is not tree-like since (x_1) is an intermediate non-read-once clause.*

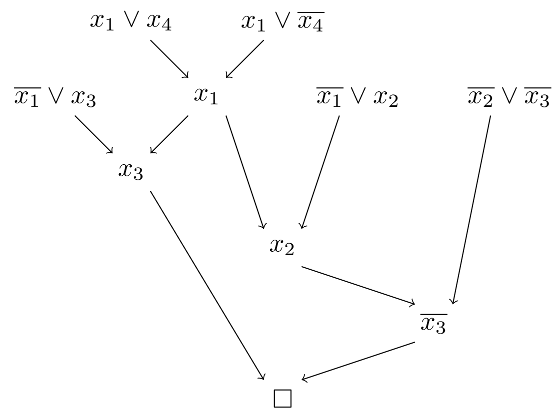

Fig. 6. Semi-tree-like resolution refutation

Proposition 1. *Let P be a resolution refutation. If P is tree-like then P is semi-tree-like.*

Proof. Suppose that P is tree-like. By definition, each intermediate clause is read-once. In each branch, the only clause that can be non-read-once in P is by definition a leaf. Therefore, at most one clause is non-read-once in each branch and we conclude that P is semi-tree-like.

■

To extend our result to semi-tree-like resolution refutations, we propose a method which relies on the fact that semi-tree resolution refutations can be partitioned into two parts where the first part is a read-once sequence of resolutions and the second part is a tree-like resolution refutation. As the first part is a read-once sequence of resolutions, it is possible to adapt it for Max-SAT using a similar method to the one in [11], i.e. replacing each resolution by a Max-SAT resolution. As the second part is a tree resolution refutation, it is possible to adapt it for Max-SAT using the result in Corollary 1. After transforming the two parts, we glue them back to construct the complete max-refutation.

Theorem 2. *Given an unsatisfiable formula ϕ and a semi-tree resolution refutation P of ϕ, there exists a max-refutation of ϕ containing $O(|P|)$ inference steps.*

Proof. As P is semi-tree-like, each branch of P contains at most one non-read-once clause. We partition P into two parts P_1 and P_2 as follows:

- For each branch containing one non-read-once clause, the transformations until this clause are put in P_1 and the transformations after this clause are put in P_2.
- For each branch not containing a non-read-once clause, the transformations are put in P_2.

By construction, P_1 is a read-once sequence of resolutions so it is possible to adapt it to obtain a Max-SAT transformation P_1' containing exactly $|P_1|$ inference steps by replacing resolutions with Max-SAT resolutions as in [11]. Furthermore, P_2 is a tree resolution refutation because the non-read-once clauses of P are leaf clauses in P_2. Consequently, it is possible to adapt P_2 into a max-refutation P_2' containing $O(|P_2|)$ inference steps using the result in Corollary 1. Finally, we can combine P_1' and P_2' to obtain a Max-SAT refutation containing at most $O(|P_1| + |P_2|)$ and we conclude that the complete adaptation contains $O(|P|)$ inference steps.

■

Example 7. *We consider the semi-tree resolution refutation in Example 6, represented in Fig. 6. To adapt this semi-tree resolution refutation to a max-refutation, we put aside the top resolution on variable x_1 taking clauses $(x_1 \vee x_4)$ and $(x_1 \vee \overline{x_4})$ and we obtain the tree-like resolution refutation in Example 1, represented in Fig. 1. We adapt this tree-like resolution refutation as in Example 4 and we replace the resolution step on x_1 by a Max-SAT resolution step (in this case no compensation clauses are generated). We glue back the two parts to obtain the complete max-refutation represented in Fig. 7.*

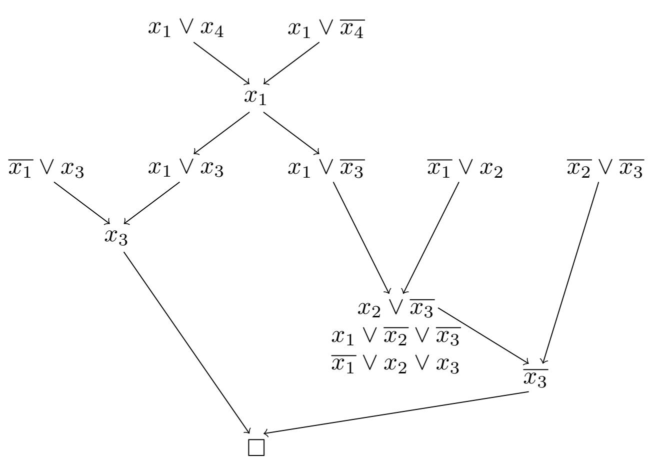

Fig. 7. Adapting a semi-tree resolution refutation to a max-refutation

VI. From Unrestricted Resolution Refutations to Max-Refutations

In Sections III, IV and V, we proposed linear adaptations from specific classes of resolution refutations to max-refutations. In this section, we propose an adaptation from any resolution refutation to a max-refutation. To make this adaptation, we will simply extend the adaptation described in section IV by adding a first transformation to make the initial resolution refutation tree-like as described in Lemma 3. Notice that we could make the initial resolution refutation semi-tree-like (instead of tree-like) but this choice does not affect the theoretical size of the obtained max-refutation.

To achieve this first intermediate transformation, we will iteratively search in the proof for the first non-read-once intermediate clause c. If this clause is used $k > 1$ times as a premise of another resolution step, we consider the part of the proof leading to c and we dupplicate it k times in order to get a tree-like sequence of resolutions generating k resolvents $c_1, c_2, ..., c_k$ (with $c_1 = c$), each resolvent c_i containing exactly the same literals as c and is generated by a similar sequence of resolution steps. Consequently, c is no longer used several times as a premise of a resolution step, the input clauses are. Repeating this operation forces the resolution refutation to become tree-like. Fixing a non-read-once intermediate clause can, in the worst case, double the size of the current resolution refutation. As such, the size of the obtained tree-like resolution refutation is exponentially bounded by the size of the initial unrestricted resolution refutation. To polish this upper bound, we introduce a new parameter defined below, which is the number of multi-uses of intermediate clauses. Notice how, in the definition, we subtract 1 use for each clause. Intuitively, we consider the first use of any non-read-once intermediate clause as authorized.

Definition 7. *Let P be a resolution refutation. The number of multi-uses of intermediate non-read-once clauses, denoted $\mu(P)$, is defined as follows:*

$$\mu(P) = \sum_{c \text{ intermediate non-read-once in } P} (d^+(c) - 1)$$

where $d^+(c)$ denotes the number of uses of the clause c, i.e. the number of outgoing arcs from c in the DAG representation of P.

Lemma 3. *Given an unsatisfiable formula ϕ and a resolution refutation P of ϕ, there exists a tree resolution refutation of ϕ containing $O(2^{\mu(P)} \times |P|)$ resolution steps.*

Proof. Let P be a resolution refutation of ϕ. We iteratively make the intermediate non-read-once clauses read-once. Each time, we pick the first intermediate non-read-once clause c and duplicate the sub-proof deriving c exactly $d^+(c) - 1$ times. Each iteration decrements the number of intermediate non-read-once clauses by 1 until the resolution refutation becomes tree-like. Clearly, for each duplication, the size of the proof is doubled in the worst case and we perform exactly $\mu(P)$ duplications. We conclude that the size of the obtained tree resolution refutation is bounded by $O(2^{\mu(P)} \times |P|)$.

■

Now that we can transform any resolution refutation to a tree-like resolution refutation, we just have to adapt the obtained tree-like resolution refutation with the method described in section IV as shown in the following Theorem.

Theorem 3. *Given an unsatisfiable formula ϕ and an unrestricted resolution refutation P of ϕ, there exists a max-refutation of ϕ with $O(2^{\mu(P)} \times |P|)$ inference steps.*

Proof. Let P be an unrestricted resolution refutation of ϕ. We adapt P to obtain a tree resolution refutation P_t of size $O(2^{\mu(P)} \times |P|)$ using Lemma 3. Then, using Theorem 1, we obtain a max-refutation of size $O(2^{\mu(P)} \times |P|)$.

■

Example 8. *We consider the resolution refutation represented in Fig. 8. This refutation is not semi-tree-like since the clauses (x_1) and (x_4) are two non read-once clauses in the same branch. First, we duplicate the resolutions leading to (x_1) and we obtain the tree-like resolution refutation represented in Fig. 9. Then, we apply the transformations described in Section IV to get the max-refutation represented in Fig. 10.*

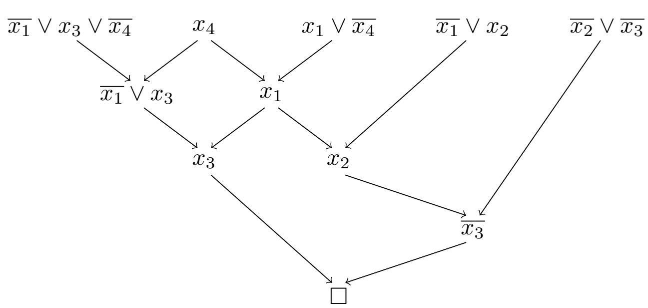

Fig. 8. Unrestricted resolution refutation

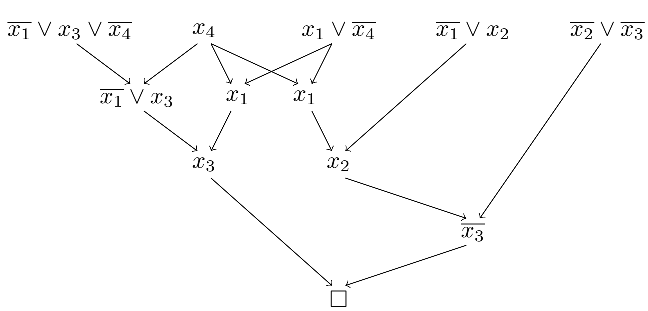

Fig. 9. Adapting a resolution refutation to a tree-like resolution refutation

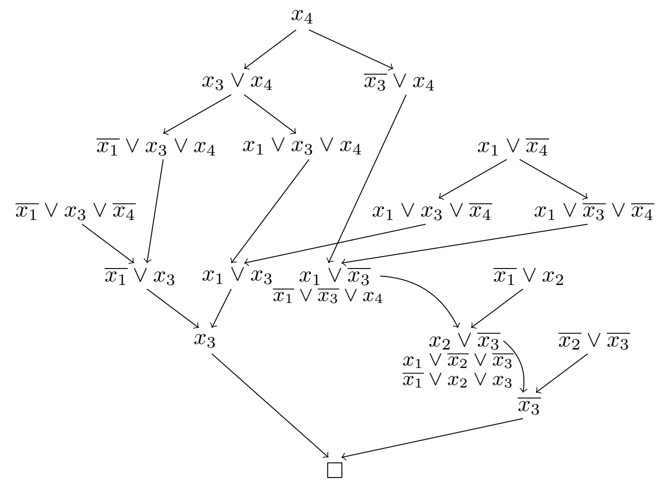

Fig. 10. Adapting an unrestricted resolution refutation to a max-refutation

We finish this section by exhibiting resolution refutations whose adaptations as in Theorem 3 is exponential. To this end, we introduce in the following definition a new pattern which we will use to build such refutations.

Definition 8 (Diamond pattern)**.** *Let A be a disjunction of literals and let $x \notin var(A)$ and $y \notin var(A)$ two distinct variables. We define the diamond pattern (x, y, A) as the sequence of resolutions represented in Fig.11.*

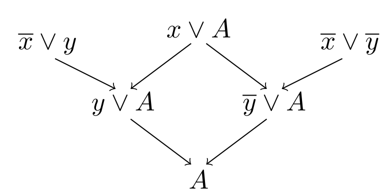

Fig. 11. Diamond pattern (x, y, A)

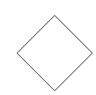

Fig. 12. Simplified representation of a diamond pattern

We can represent this pattern by a diamond as in Fig. 12. Notice that in particular, the diamond pattern $(x, y, \Box)$ is a resolution refutation. Now, imagine that the topmost clause of $(x, y, \Box)$ is derived through another diamond pattern. We iterate the same reasoning to define a k-stacked diamonds pattern as follows:

Definition 9 (k-stacked diamond pattern)**.** *Let $k \geq 1$ be a natural number and let x_i and y_i where $1 \leq i \leq k$ be distinct variables. A k-stacked diamond pattern is formed by k diamond patterns (x_i, y_i, A_i) where $1 \leq i \leq k$ such that $A_1 = \Box$ and $A_i = (x_1 \lor \cdots \lor x_{i-1})$ for $1 < i \leq k$. Each diamond (x_i, y_i, A_i) is stacked on top of $(x_{i-1}, y_{i-1}, A_{i-1})$ such that the last conclusion of the former is the topmost central premise of the latter.*

A k-stacked diamond pattern is represented as a stack of diamonds as shown in Fig.13 for $k = 3$. Clearly, k-stacked diamond are resolution refutations as they deduce the empty clause $\Box$. In particular, when $k > 2$, a k-stacked diamond is not semi-tree-like. The size of a k-stacked diamond P is $|P| = 3k$. Furthermore, we have $\mu(P) = k - 1$. Therefore, after the application of the adaptation described in Theorem 3, we obtain a max-refutation whose size is at least 2^{k-1} showing that the proposed adaptation can be exponential in the worst case.

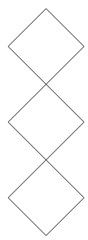

Fig. 13. Simplified representation of a 3-stacked diamond pattern

VII. Conclusions and Future Work

The contributions of this work are related to adapting resolution refutations to Max-SAT refutations. In particular, we have proposed linear adaptations regarding the size of the resolution refutations in the following cases: regular tree resolution, tree resolution and semi-tree resolution. These results are achieved by augmenting Max-SAT resolution with the split rule which enabled us to duplicate clauses by adding literals when necessary. We have also generalised our adaptation to unrestricted resolution refutations, even though the proposed transformation can produce a max-refutation whose size is exponential in the worst case. Notice that our results remain valid for weighted Max-SAT formulas as we simply need to augment the previous rules with another split rule for weights. Indeed, the overhead of this rule is linear in terms of the size of the refutation since we need to apply it once on the clauses in the weighted MAX-SAT formula to produce clauses with same weight, i.e. the minimum of all the weights as done in the context of SAT-based (weighted) Max-SAT algorithms [3].

These results may help to exhibit proofs based on Max-SAT resolution for Max-SAT algorithms which remains an unexplored topic whereas, in SAT, practically all modern solvers are able to compute a resolution proof of unsatisfiability (in different formats [10], [13]). Indeed, it would be interesting to include the proposed adaptations in SAT-based algorithms for Max-SAT. Such extended algorithms would thus iteratively call a SAT oracle to get a resolution refutation, adapt this resolution refutation to get a Max-SAT refutation based on our results and transform the formula accordingly. This treatment is repeated until reaching a satisfiable formula and a set of empty clauses whose size is the optimum value of the formula. Such implementation would require an efficient SAT oracle which returns a resolution refutation. Finally, the existence of an adaptation that does not increase substantially the size of an unrestricted resolution proof remains an open question. We will continue to investigate this topic either by exhibiting a polynomial adaptation or refuting its existence.

References

[1] A. Abramé and D. Habet. On the resiliency of unit propagation to max-resolution. In *IJCAI*, 2015.

[2] A. Abramé and D. Habet. ahmaxsat: Description and evaluation of a branch and bound max-sat solver. In *Journal on Satisfiability, Boolean Modeling and Computation. Vol. 9, pp. 89–128*, 2015.

[3] C. Ansótegui, M. L. Bonet, and J. Levy. Solving (weighted) partial maxsat through satisfiability testing. In O. Kullmann, editor, *SAT 2009*, volume 5584 of *Lecture Notes in Computer Science*, pages 427–440. Springer, 2009.

[4] E. Ben-sasson, R. Impagliazzo, and A. Wigderson. Near optimal separation of tree-like and general resolution. *Combinatorica*, 24:585–603, 09 2004.

[5] M. Bonet, J. Levy, and F. Manyà. A complete calculus for max-sat. In *SAT 2006*, volume 4121, pages 240–251, 08 2006.

[6] S. Buss, J. Hoffmann, and J. Johannsen. Resolution Trees with Lemmas: Resolution Refinements that Characterize DLL Algorithms with Clause Learning. *Logical Methods in Computer Science*, 4, 11 2008.

[7] M. S. Cherif and D. Habet. Towards the characterization of max-resolution transformations of ucss by up-resilience. In T. Schiex and S. de Givry, editors, *CP*, pages 91–107, Cham, 2019. Springer International Publishing.

[8] M. Davis, G. Logemann, and D. Loveland. A machine program for theorem-proving. *Communications of the ACM*, 5(7):394–397, 1962.

[9] A. V. Gelder. Extracting (easily) checkable proofs from a satisfiability solver that employs both preorder and postorder resolution. In *ISAIM*, 2002.

[10] A. V. Gelder. Verifying rup proofs of propositional unsatisfiability. In *ISAIM*, 2008.

[11] F. Heras and J. Marques-Silva. Read-once resolution for unsatisfiability-based max-sat algorithms. In *IJCAI*, 2011.

[12] A. Hertel and A. Urquhart. Algorithms and complexity results for input and unit resolution. *Journal of Satisfiability, Boolean Modeling and Computation*, 6, 05 2009.

[13] M. J. Heule, W. A. Hunt, and N. Wetzler. Trimming while checking clausal proofs. In *2013 Formal Methods in Computer-Aided Design*, pages 181–188. IEEE, 2013.

[14] K. Iwama and E. Miyano. Intractability of read-once resolution. In *Proceedings of Structure in Complexity Theory. Tenth Annual IEEE Conference*, 1995.

[15] A. Küegel. Improved exact solver for the weighted max-sat problem. In *Daniel Le Berre (editor). POS-10. Pragmatics of SAT, vol 8, pages 15–27*, 2012.

[16] J. Larrosa and F. Heras. Resolution in max-sat and its relation to local consistency in weighted csps. In *IJCAI*, pages 193–198, 01 2005.

[17] C.-M. Li, F. Manyà, and J. R. Soler. A clause tableau calculus for maxsat. In *IJCAI*, IJCAI'16, page 766–772. AAAI Press, 2016.

[18] C.-M. Li, F. Manyà, and J. Planes. New inference rules for max-sat. *J. Artif. Intell. Res. (JAIR)*, 30:321–359, 09 2007.

[19] D. Loveland. A linear format for resolution. *Symposium on Automatic Demonstration*, pages 147–162, 01 1970.

[20] M. Luisa Bonet, J. Levy, and F. Manyà. Resolution for max-sat. *Artificial Intelligence Volume 171, Issues 8–9, June 2007, Pages 606-618*, 2007.

[21] J. P. Marques Silva and K. A. Sakallah. Grasp-a new search algorithm for satisfiability. In *Proceedings of International Conference on Computer Aided Design*, pages 220–227, 1996.

[22] N. Narodytska and F. Bacchus. Maximum satisfiability using core-guided maxsat resolution. In *AAAI*, 2014.

[23] K. Pipatsrisawat and A. Darwiche. On the power of clause-learning sat solvers as resolution engines. *Artificial Intelligence*, 175(2):512 – 525, 2011.

[24] J. A. Robinson. A machine-oriented logic based on the resolution principle. In *Journal of the Association for Computing Machinery, vol. 12 (1965), pp. 23–41.*, 1965.

[25] A. Urquhart. The complexity of propositional proofs. *Bull. Symbolic Logic*, 1(4):425–467, 12 1995.

[26] A. Urquhart. A near-optimal separation of regular and general resolution. *SIAM J. Comput.*, 40:107–121, 01 2011.