Proof Complexity for the Maximum Satisfiability Problem and its Use in SAT Refutations [*]

Emma Rollon EROLLON@CS.UPC.EDU
Universitat Politècnica de Catalunya
Barcelona, Spain

Javier Larrosa LARROSA@CS.UPC.EDU
Universitat Politècnica de Catalunya
Barcelona, Spain

Abstract

MaxSAT, the optimization version of the well-known SAT problem, has attracted a lot of research interest in the last decade. Motivated by the many important applications and inspired by the success of modern SAT solvers, researchers have developed many MaxSAT solvers. Since most research is algorithmic, its significance is mostly evaluated empirically. In this paper we want to address MaxSAT from the more formal point of view of *Proof Complexity*. With that aim we start providing basic definitions and proving some basic results. Then we analyze the effect of adding *split* and *virtual*, two original inference rules, to MaxSAT resolution. We show that each addition makes the resulting proof system stronger, even when virtual is restricted to empty clauses (0-virtual). We also analyze the power of our proof systems in the particular case of SAT refutations. We show that our strongest system, **ResSV**, is equivalent to *circular* and *dual rail with split*. We also analyze empirically some known gadget-based reformulations. Our results seem to indicate that the advantage of these three seemingly different systems over general resolution comes mainly from their ability of augmenting the original formula with hypothetical inconsistencies, as captured in a very simple way by the virtual rule.

1. Introduction

Proof complexity is the field aiming to understand the computational cost required to prove or refute statements. Different proof systems may provide different proofs for the same formula and some proof systems are provably more efficient than others. When that happens, proof complexity cares about which elements of the more powerful proof system really make the difference.

In propositional logic, resolution-based proof systems that work with CNF formulas have attracted the interest of researchers for several decades [10]. One of the reasons is that CNF is the working language of the extremely successful Conflict-Driven Clause Learning SAT solvers (CDCL), and it is known that CDCL solvers are as powerful as general resolution [5].

In this paper we are concerned with (Partial Weighted) *MaxSAT* which is the optimization version of SAT. Many discrete optimization problems are naturally represented as MaxSAT problems and, consequently, many researchers have worked on the design of efficient MaxSAT solvers[1]. The generalization of the resolution inference rule to MaxSAT was first proposed in [20] and further studied in [9] and [21].

In the first part of this paper we make, for the first time to the best of our knowledge, an attempt to study MaxSAT proof complexity. Our approach is based on two principles: *i*) to define MaxSAT proof systems in a way as similar as possible to SAT proof systems in order to identify similarities and differences, and *ii*) to study the effect of augmenting MaxSAT resolution with additional inference rules.

∗. Under project RTI2018-094403-B-C33 funded by MCIN/AEI/ 10.13039/501100011033 and FEDER "Una manera de hacer Europa".

1. https://maxsat-evaluations.github.io/

Our contribution in this part is two-fold. First, we extend two classic proof complexity concepts (i.e, entailment and completeness) from SAT to MaxSAT. Thanks to those definitions we show that, similarly to what happens in SAT, refutational completeness makes completeness somehow redundant. In other words, a MaxSAT solver can be used to prove or disprove entailment. Second, we introduce *split* and *virtual*, two new MaxSAT inference rules that complement MaxSAT resolution, and study their power. From the proof system containing only resolution (**Res**), which is known to be sound and refutationally complete, we show that each add-on makes a stronger system: adding the split rule (**ResS**) we obtain completeness and, unlike what happens in SAT, exponential speed-up in certain refutations; further adding the virtual rule (**ResSV**), which allows to keep negative weights during proofs, results in further exponential speed-up. Finally, we show that the restriction of the virtual rule to the special case of empty clauses ($\mathbf{ResSV}^0$) does not lose any power. The top of Figure 1 summarizes these contributions.

The theoretical power of CDCL SAT solvers is tightly related to the power of SAT resolution. Consequently, in the last years some papers propose SAT proof systems stronger than resolution because they may lead to more powerful SAT solvers [13, 27]. Notably, [6, 1] propose to refute SAT formulas by transforming the problem into MaxSAT and solving it with a MaxSAT solver. We find this approach intriguing and its success somehow paradoxical: it indicates that it may be efficient to refute a SAT formula by transforming it into a MaxSAT problem and solve it with a MaxSAT solver which, in turn, uses a SAT solver as an oracle. So the second part of the paper aims to shed some light to this issue with a pure proof complexity approach. In particular, we consider three recent approaches to refute SAT formulas which are stronger than resolution: circular refutations [4], dual rail refutations [6] and gadget-based refutations [1]. While seemingly very different, the three approaches have in common that can solve the pigeon hole principle (PHP) in polynomial time.

Our contribution in the second part of the paper is to show that **ResSV**, circular proofs and dual rail with split can p-simulate each other when refuting SAT formulas. Regarding gadget-based refutations, we cannot relate it with $\mathbf{ResSV}^0$ through p-simulation because gadgets do not preserve the formula cost function, but we show empirically that the good results reported for the PHP problem with some gadgets do not come from the arity reduction or the introduction of auxiliary variables, which seem to be their main common features, but mainly because these gadgets contain unit soft clauses similar to those introduced in the dual rail encoding, in the circular refutation or with the virtual rule. We support our claim by showing that a dummy gadget with ternary clauses is nearly as efficient as the best ones reported in [1]. Additionally, we show that a ternary version of the best performing gadget turns out to be as effective as the binary version. The bottom of Figure 1 summarizes the theoretical contributions of the second part of the paper.

This paper contains and extends previous work from [22] and [23]. In this paper proofs are more detailed and the lower bound of Theorem 1 has been refined. Results about $\mathbf{ResSV}^0$ are new (Theorem 11), including its equivalence with Dual Rail refutations (Theorems 15 and 16). The study of gadgets in Subsection 7.4 and the discussion of soft-probing in Subsection 8.1 is also new.

Bonet and Levy [8] have independently studied the equivalence between our proof system in [22], which is similar to **ResSV**, and circular proofs restricting their attention to the refutation of SAT formula. They prove that **ResSV** and SAT Circular Proofs are polynomially equivalent. This is the same as our Theorems 19 and 20 because their choice of replacing infinities by sufficiently high weights does not affect the effectiveness of **ResSV**, as we shown in Lemma 5. Two other recent papers also report related work conducted in parallel to ours. The work in [14] also studies SAT refutations using MaxSAT resolution. In their work the original SAT formula is transformed into a MaxSAT one replacing each hard clause by an identical soft clause with weight 1. In this particular setting they show, among other things, that standard SAT resolution lies between **ResSV** and **ResS**, and SAT tree-resolution is weaker than **ResS** and it might be weaker than **Res** as well. In [7] a detailed analysis of the dual rail encoding shows that a *minimum hitting set dual rail MaxSAT* algorithm is more efficient than standard dual rail refutations in some combinatorial principle. Such

MaxSAT

$$\begin{tikzcd}
\mathbf{ResSV}^0 \arrow[d, leftrightarrow, "{[\text{Th } 11]}"'] & & \\
\mathbf{ResSV} \arrow[r, "SPHP", "{[\text{Th } 10]}"'] & \mathbf{ResS} \arrow[r, "SPHP^0", "{[\text{Th } 6]}"'] & \mathbf{Res}
\end{tikzcd}$$

SAT refutation

$$\begin{tikzcd}
\mathbf{DRS} \arrow[r, dotted] \arrow[d, leftrightarrow, "{[\text{Th } 15, 16]}"'] & \mathbf{DR} \arrow[dd, "PHP"] & \\
\mathbf{ResSV}^0 \arrow[d, leftrightarrow] & & \\
\mathbf{ResSV} \arrow[r, "PHP", "{[\text{Cor } 1]}"'] \arrow[d, leftrightarrow, "{[\text{Th } 19, 20]}"'] & \mathbf{ResS} \arrow[r, leftrightarrow, "{[\text{Prop } 2]}"'] & \mathbf{Res} \\
\mathbf{Circular} & &
\end{tikzcd}$$

Figure 1: Comparison among proof systems. An **S** $\longleftrightarrow$ **S'** means that **S** and **S'** *p*-simulate each other (i.e., they are equivalent). Note that when **S** $\longleftrightarrow$ **S'** and **S'** $\longleftrightarrow$ **S"** then **S** $\longleftrightarrow$ **S"**. **S** $\longrightarrow$ **S'** means that **S** p-simulates **S'**, but **S'** does not p-simulate **S**. The problem that proves that the p-simulation is not in both directions is indicated over the arrow. A dotted arrow means that it is not known if the *p*-simulation in the other direction holds. The theorem, corollary or property that proves the relation among proof systems is indicated in square brackets. The top box shows a comparison among different MaxSAT proof systems. The lower box shows a comparison restricted to SAT refutations (i.e, input formulas with hard clauses only). DR and DRS correspond to dual rail and dual rail with split, respectively.

algorithm is not a proof system in the traditional sense, but can be seen as one using a looser definition.

The structure of the paper is as follows. Section 2 provides preliminaries on SAT and MaxSAT. Section 3 defines some variations of the Pigeon Hole Problem that are used for proving some theorems. Section 4 reviews some concepts on SAT proof systems and Section 5 extends them to MaxSAT. Section 6 presents and analyzes four proof systems: **Res**, **ResS**, **ResSV** and **ResSV**^{0}. Section 7 proves the equivalence for SAT refutations of **ResSV**, Circular and Dual Rail with split refutations, and discuss gadget-based refutations. Section 8 contextualizes our work with some previous related works and finally, Section 9, gives some conclusions.

2. Background

2.1 SAT Problem

A *boolean variable* x takes values on the set $\{0, 1\}$. A *literal* is a variable x (positive literal) or its negation $\overline{x}$ (negative literal). We will use sets of literals to denote variable assignments (a.k.a. truth assignments) with literals with x (respectively $\overline{x}$) representing that variable x is instantiated with 1 (respectively 0). A *clause* is a disjunction of literals. A clause C is satisfied by a truth assignment X if X contains at least one of the literals in C. The *empty clause* is denoted $\square$ and cannot be satisfied.

A *CNF formula* $\mathcal{F}$ is a set of clauses (taken as a conjunction). A truth assignment satisfies a formula if it satisfies all its clauses. If such an assignment exists, we say that the assignment is a *model* and the formula is *satisfiable*. We say that formula $\mathcal{F}$ *entails* formula $\mathcal{G}$, noted $\mathcal{F} \models \mathcal{G}$, if every model of $\mathcal{F}$ is also a model of $\mathcal{G}$. Two formulas $\mathcal{F}$ and $\mathcal{G}$ are *equivalent*, noted $\mathcal{F} \equiv \mathcal{G}$, if they entail each other.

Given a formula $\mathcal{F}$, the SAT problem, noted $SAT(\mathcal{F})$, is to determine if $\mathcal{F}$ is satisfiable or not. The negation of a clause $C = l_1 \vee l_2 \vee \ldots \vee l_p$ is satisfied if all its literals are falsified and this can be trivially expressed in CNF as the set of unit clause $\overline{C} = \{\bar{l}_1, \bar{l}_2, \ldots, \bar{l}_p\}$.

2.2 MaxSAT Problem

A *weight* w is a non-negative integer or ∞ (i.e, $w \in \mathbb{N} \cup \{\infty\}$). We extend addition and subtraction to weights defining $\infty + w = \infty$ and $\infty - w = \infty$ for all w. Note that $v - w$ is only defined when $w \leqslant v$.

A *weighted clause* is a pair (C, w) where C is a clause and w is a weight associated to its falsification. If $w = \infty$ we say that the clause is *hard*, else it is *soft*.

A *weighted MaxSAT CNF formula* is a multiset of weighted clauses $\mathcal{F} = \{(C_1, w_1), \ldots, (C_p, w_p)\}$. If all the clauses are hard, we say that the formula is *hard*. If all the clauses are soft, we say that the formula is *soft*. Otherwise the formula is *mixed*. Unless we explicitly say otherwise, we will assume mixed formulas. This definition of MaxSAT including soft and hard clauses is sometimes referred to as Partial Weighted MaxSAT [24] and corresponds to the most general MaxSAT language.

Given a formula $\mathcal{F}$, we define the cost of a truth assignment X, noted $\mathcal{F}(X)$, as the sum of weights over the clauses that are falsified by X. We say that formula $\mathcal{F}$ *entails* formula $\mathcal{G}$, noted $\mathcal{F} \models \mathcal{G}$, if for all X, $\mathcal{G}(X)$ is a lower bound of $\mathcal{F}(X)$ (i.e., $\forall X$, $\mathcal{F}(X) \geqslant \mathcal{G}(X)$). We say that two formulas $\mathcal{F}$ and $\mathcal{G}$ are *equivalent*, noted $\mathcal{F} \equiv \mathcal{G}$, if they entail each other (i.e., $\forall X$, $\mathcal{F}(X) = \mathcal{G}(X)$). We say that $\mathcal{G} \sqsubseteq \mathcal{F}$ if for all $(C, w) \in \mathcal{G}$ we have that $\sum_{(C,u) \in \mathcal{G}} u \leqslant \sum_{(C,v) \in \mathcal{F}} v$.

Given a formula $\mathcal{F}$, the MaxSAT problem, noted $MaxSAT(\mathcal{F})$, is to find the minimum cost over the set of all truth assignments,

$$MaxSAT(\mathcal{F}) = \min_X \mathcal{F}(X)$$

Note that if the hard clauses of the formula make it unsatisfiable then $MaxSAT(\mathcal{F}) = \infty$.

In the following sections we will find useful to deal with negated weighted clauses. Hence, the corresponding definitions and useful property. Let A and B be arbitrary disjunctions of literals. Let $(A \vee \overline{B}, w)$ mean that falsifying $A \vee \overline{B}$ incurs a cost of w. Although $A \vee \overline{B}$ is not a clause, the following property shows that it can be efficiently transformed into a weighted CNF equivalent.

Property 1 $\{(A \vee \overline{l_1 \vee l_2 \vee \ldots \vee l_p}, w)\} \equiv \{(A \vee \bar{l}_1, w), (A \vee l_1 \vee \bar{l}_2, w), \ldots, (A \vee l_1 \vee \ldots \vee l_{p-1} \vee \bar{l}_p, w)\}$.

The negation of a MaxSAT formula $\mathcal{F}$ is the negation of all its clauses,

$$\overline{\mathcal{F}} = \{(\overline{C}, w) \mid (C, w) \in \mathcal{F}\}$$

For example, the negation of formula $\mathcal{F} = \{(x \vee y, \infty), (\overline{x} \vee \overline{y}, 3)\}$ is $\overline{\mathcal{F}} = \{(\overline{x}, \infty), (x \vee \overline{y}, \infty), (x, 3), (\overline{x} \vee y, 3)\}$.

3. Pigeon Hole Problem and Variations

We define the well-known Pigeon Hole Problem *PHP* and three MaxSAT soft versions *SPHP*, *SPHP*^{0} and *SPHP*^{1}, that we will be using in the proof of our results.

In the *Pigeon Hole Problem PHP* the goal is to assign $m + 1$ pigeons to m holes without any pair of pigeons sharing their hole. In the usual SAT encoding there is a boolean variable x_{ij} (with $1 \leqslant i \leqslant m+1$, and $1 \leqslant j \leqslant m$) which is true if pigeon i is in hole j. There are two groups of clauses. For each pigeon i, we have the clause,

$$\mathcal{P}_i = \{x_{i1} \vee x_{i2} \vee \ldots \vee x_{im}\}$$

indicating that pigeon i must be assigned to at least one hole. For each hole j we have the set of clauses,

$$\mathcal{H}_j = \{\overline{x}_{ij} \vee \overline{x}_{i'j} \mid 1 \leqslant i < i' \leqslant m+1\}$$

indicating that hole j is occupied by at most one pigeon. Let $\mathcal{K}$ be the union of all these sets of clauses $\mathcal{K} = \cup_{1 \leqslant i \leqslant m+1} \mathcal{P}_i \cup \cup_{1 \leqslant j \leqslant m} \mathcal{H}_j$. It is obvious that $\mathcal{K}$ is an unsatisfiable CNF formula. In MaxSAT notation the pigeon hole problem is,

$$PHP = \{(C, \infty) \mid C \in \mathcal{K}\}$$

and clearly $MaxSAT(PHP) = \infty$.

In the *soft Pigeon Hole Problem SPHP* the goal is to find the assignment that falsifies the minimum number of clauses. In MaxSAT language it is encoded as,

$$SPHP = \{(C, 1) \mid C \in \mathcal{K}\}$$

and it is obvious that $MaxSAT(SPHP) = 1$.

The *SPHP⁰* problem is like the soft pigeon hole problem but augmented with one more clause $(\square, m^2 + m)$ where m is the number of holes. Note that $MaxSAT(SPHP^0) = m^2 + m + 1$.

Finally, the *SPHP¹* problem is like the soft pigeon hole problem but augmented with a set of unit clauses $\{(x_{ij}, 1), (\overline{x}_{ij}, 1) \mid 1 \leqslant i \leqslant m+1, 1 \leqslant j \leqslant m\}$. Note that $MaxSAT(SPHP^1) = m^2 + m + 1$.

4. SAT Proof Systems

A *SAT proof system* **S** is a set of inference rules. An *inference rule* is given by a set of antecedent clauses and a set of consequent clauses. In SAT, an inference rule means that if the antecedents are members of the formula, the consequents can be *added.* The rule is *sound* if every truth assignment that satisfies the antecedents also satisfies the consequents.

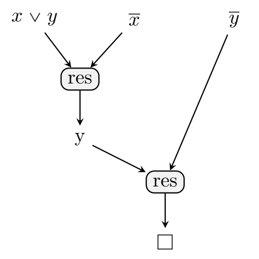
A *proof*, or *derivation*, under a proof system **S** is a finite sequence $C_1, C_2, \ldots, C_e$ where the start of the sequence, $C_1, \ldots, C_p$, is the original formula $\mathcal{F}$ and each C_i (with $i > p$) is obtained by applying an inference rule from **S** with earlier antecedents (i.e., C_j with $j < i$). The *length* of the proof is $e - p$. A *polynomial size* proof is a proof whose length can be bounded by a polynomial on $|\mathcal{F}|$.

We will write $\mathcal{F} \vdash_S \mathcal{G}$ to denote an arbitrary proof $\Pi = (C_1, C_2, \ldots, C_e)$ with $\mathcal{F} = \{C_1, \ldots, C_p\}$ and $\mathcal{G} \subseteq \cup_{i=1}^{e} \{C_i\}$ (abusing notation, in the following we will note $\mathcal{G} \subseteq \cup_{i=1}^{e} \{C_i\}$ as $\mathcal{G} \subseteq \Pi$). When the proof system is irrelevant or implicit from the context we will just write $\mathcal{F} \vdash \mathcal{G}$. A *refutation* of $\mathcal{F}$ is a proof $\mathcal{F} \vdash_S \square$. Refutations are important because they prove unsatisfiability.

A proof $\Pi = (C_1, C_2, \ldots, C_e)$ can be graphically represented as an acyclic directed bi-partite graph $G(\Pi) = (J \cup I, E)$ such that in $J = \{C_1, ..., C_e\}$ and each node in I represents an inference step. Consider the inference step with antecedents $\mathcal{A} \subset \Pi$ and consequents $\mathcal{C} \subset \Pi$. Node $R \in I$ has $\mathcal{A}$ in-neighbours and $\mathcal{C}$ out-neighbours. Since the same clause can be derived several times, different nodes in J may correspond to the same clause. Note that clauses in the original formula $\mathcal{F}$ do not have in-neighbors. The rest of the clauses have exactly one in-neighbour. All clauses may have several out-neighbors since they may be used as an antecedent several times during the proof.

A proof system **S** is *sound* if $\mathcal{F} \vdash_S \mathcal{G}$ implies $\mathcal{F} \models \mathcal{G}$. It is *complete* if $\mathcal{F} \models \mathcal{G}$ implies $\mathcal{F} \vdash_S \mathcal{G}$. A proof system **S** is *refutationally complete* if $\mathcal{F} \models \square$ implies $\mathcal{F} \vdash_S \square$. In words, for every unsatisfiable formula $\mathcal{F}$ there is a refutation $\mathcal{F} \vdash_S \square$ (i.e, completeness is required only for refutations). Refutational completeness is enough for most practical purposes because $\mathcal{F} \models \mathcal{G}$ if and only if $\mathcal{F} \cup \overline{\mathcal{G}} \models \square$ (i.e., $\mathcal{F} \cup \overline{\mathcal{G}}$ is unsatisfiable), so any refutationally complete proof system can prove the entailment by deriving $\square$ from a CNF formula equivalent to $\mathcal{F} \cup \overline{\mathcal{G}}$.

The most usual way to compare the strength of different proof systems is with the concept of p-simulation. We say that proof system **S** p-simulates proof system **S'** if there is a polynomially



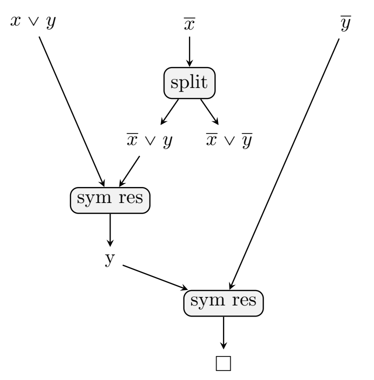


Figure 2: Refutation graph for $\{x \vee y, \overline{x}, \overline{y}\}$ using the resolution rule (left) and symmetric resolution with split (right).

computable function f such that for every **S**-refutation Π of formula $\mathcal{F}$, $f(\Pi)$ is an **S'**-refutation of the same formula $\mathcal{F}$. If **S** p-simulates **S'** and **S'** does not p-simulate **S** we say that **S** is stronger or more powerful than **S'**.

Consider the following three sound inference rules [25] [4],

$$\frac{x \vee A \qquad \overline{x} \vee B}{A \vee B} \qquad \frac{x \vee A \qquad \overline{x} \vee A}{A} \qquad \frac{A}{A \vee x \qquad A \vee \overline{x}}$$

`(resolution)` `(symmetric resolution)` `(split)`

where A and B are arbitrary (possibly empty) disjunctions of literals and x is an arbitrary variable. In propositional logic it is customary to define rules with just one consequent because one rule with s consequents can be obtained from s one-consequent rules. As we will see, this is not the case in MaxSAT. For this reason, here we prefer to introduce the two-consequents split rule instead of the equivalent weakening rule [4] to keep the parallelism with MaxSAT more evident.

It is well-known that the proof system made exclusively of resolution is refutationally complete and adding the split rule makes the system complete. However, the following property says that adding the split rule does not give any advantage to resolution in terms of refutational power,

Property 2 *[(see Lemma 7 in [2]] A proof system with resolution and split as inference rules cannot make shorter refutations than a proof system with only resolution.*

It is easy to see that resolution can be simulated by split and symmetric resolution, so the resulting proof system is also complete. Figure 2 shows a refutation graph of $\{x \vee y, \overline{x}, \overline{y}\}$ using the resolution rule (left) and symmetric resolution with split (right).

5. MaxSAT Proof Systems and Completeness

A *MaxSAT proof system* **S** is a set of MaxSAT inference rules. A *MaxSAT inference rule* is given by a set of antecedent clauses and a set of consequent clauses. In MaxSAT, the application of an inference rule is to *replace* the antecedents by the consequents. The process of applying an inference rule to a formula $\mathcal{F}$ is noted $\mathcal{F}; \mathcal{F}'$. The rule is *sound* if it preserves the equivalence of the formula i.e, $\mathcal{F} \equiv \mathcal{F}'$.

A *proof*, or *derivation*, with proof system **S** is a sequence $\mathcal{F}_0; \mathcal{F}_1; \ldots; \mathcal{F}_e$ where $\mathcal{F}_0$ is the original formula $\mathcal{F}$ and each $\mathcal{F}_i$ is obtained by applying an inference rule from **S**. The length of the proof is e. Note that MaxSAT proofs are sequences of formulas while SAT proofs are sequences of clauses.

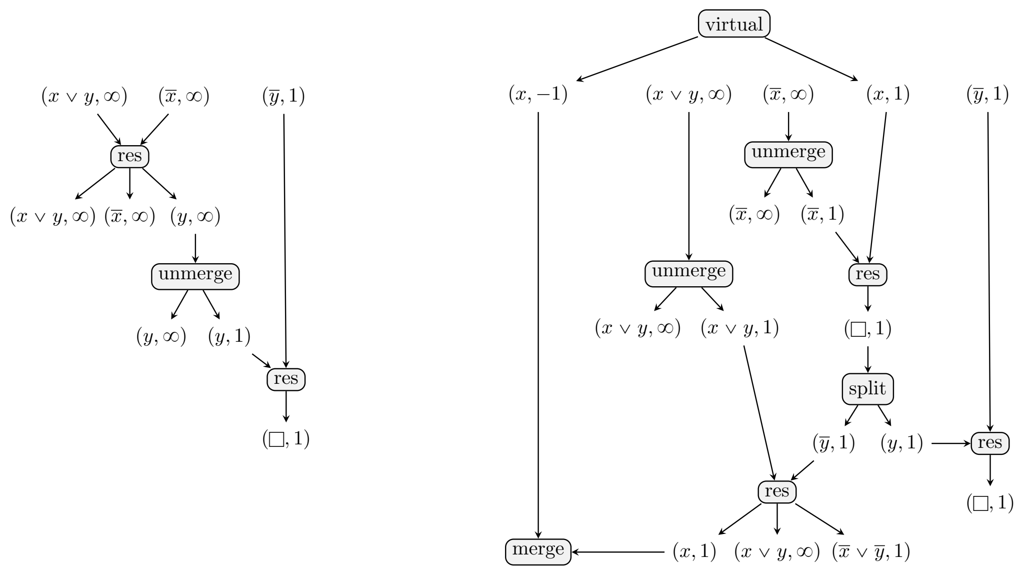


Figure 3: A MaxSAT refutation graph for MaxSAT formula $\{(x \vee y, \infty), (\overline{x}, \infty), (\overline{y}, 1)\}$ using **Res** (left) and **ResSV** (right), respectively.

We use the semi-colon to emphasize this distinction. The reason is that MaxSAT inference rules modify clauses already in the formula in order to derive new ones, so each step of the proof must carry along the whole formula. Note that a SAT proof $C_1, C_2, \ldots, C_e$ with $\mathcal{F}_0 = \{C_1, ..., C_p\}$ with comma notation can easily be transformed to the semi-colon notation as $\mathcal{F}_0; \mathcal{F}_1; \ldots; \mathcal{F}_{e-p}$ where each formula $\mathcal{F}_i$ contains the new clauses and all the previous clauses, $\mathcal{F}_i = \{C_1, ..., C_{p+i}\}$.

We will write $\mathcal{F} \vdash_S \mathcal{G}$ to denote an arbitrary proof $\mathcal{F}_0; \mathcal{F}_1; \ldots; \mathcal{F}_e$ with $\mathcal{F} = \mathcal{F}_0$ and $\mathcal{G} \subseteq \mathcal{F}_e$. A proof system **S** is *sound* if $\mathcal{F} \vdash_S \mathcal{G}$ implies $\mathcal{F} \models \mathcal{G}$. It is *complete* if $\mathcal{F} \models \mathcal{G}$ implies $\mathcal{F} \vdash_S \mathcal{G}$. A *k-refutation* of $\mathcal{F}$ is a proof $\mathcal{F} \vdash_S (\square, k)$. A proof system is *refutationally complete* if there is a proof $\mathcal{F} \vdash_S (\square, k)$ for every formula $\mathcal{F}$ and every $k \leqslant MaxSAT(\mathcal{F})$.

As in the SAT case, a MaxSAT proof $\Pi = (\mathcal{F}_0; \mathcal{F}_1; \ldots; F_e)$ can be graphically represented as an acyclic directed bi-partite graph $G(\Pi) = (J \cup I, E)$ where there is one node in J for each clause and one node in I for each inference step. Clauses in $\mathcal{F}_0$ do not have in-neighbours. Consider proof step $\mathcal{F}_{i-1}; \mathcal{F}_i$ where the antecedents of the inference are $\mathcal{A} \subseteq \mathcal{F}_{i-1}$ and the consequents are $\mathcal{C} \subseteq \mathcal{F}_i$. The inference node has $\mathcal{A}$ as in-neighbors and $\mathcal{C}$ as out-neighbors.

There are two differences with respect to the SAT case: nodes in J contain a clause and a weight, and they have at most one out-neighbor. Figure 3 shows two refutation graphs for $\{(x \vee y, \infty), (\overline{z}, \infty), (\overline{y}, 1)\}$ using two different proof systems, to be defined later.

Now we show that, similarly to what happens in SAT, refutationally completeness is sufficient for practical purposes. The reason is that it can also be used to prove or disprove general entailment, making completeness somehow redundant. Let the *roof* of a formula $\mathcal{F}$, noted $rf(\mathcal{F})$, be the sum of its weights,

$$rf(\mathcal{F}) = \sum_{(C,w) \in \mathcal{F}} w$$

The following property shows the effect of negating a soft formula.

Property 3 *If $\mathcal{F}$ is a soft MaxSAT formula then*

$$\overline{\mathcal{F}}(X) = rf(\mathcal{F}) - \mathcal{F}(X)$$

Proof. Given a clause (C, w), any truth assignment always falsifies either (C, w) or $(\overline{C}, w)$, but not both. Therefore, for each clause $(C, w) \in \mathcal{F}$, any truth assignment X will incur a cost w in $\mathcal{F} \cup \overline{\mathcal{F}}$. Consequently, $\mathcal{F}(X) + \overline{\mathcal{F}}(X) = \sum_{(w,C)\in\mathcal{F}} w = rf(\mathcal{F})$, which proves the property.

Next, we show that an entailment $\mathcal{F} \models \mathcal{G}$ can be rephrased as a refutation,

Theorem 1 *Let $\mathcal{F}$ and $\mathcal{G}$ be two MaxSAT formulas, possibly with soft and hard clauses. Then,*

$$\mathcal{F} \models \mathcal{G} \text{ iff } (\mathcal{F} \cup \overline{\mathcal{G}}^{\gamma}) \models (\Box, rf(\mathcal{G}^{\gamma}))$$

where $\mathcal{G}^{\gamma}$ is similar to $\mathcal{G}$ but its infinity weights are replaced by a value γ higher than the maximum finite cost of $\mathcal{F}$,

$$\gamma > \max_{X \mid \mathcal{F}(X) < \infty} \mathcal{F}(X)$$

with $\gamma > 0$ if $\mathcal{F}$ is unsatisfiable.

Proof. Let us prove the if direction. $\mathcal{F} \models \mathcal{G}$ means that $\mathcal{F}(X) \geqslant \mathcal{G}(X)$ for all X. We know, by construction of G^{γ} that $\mathcal{G}(X) \geqslant \mathcal{G}^{\gamma}(X)$. Therefore, $\mathcal{F}(X) \geqslant \mathcal{G}^{\gamma}(X)$ for all X. Because $\mathcal{G}^{\gamma}$ does not contain hard clauses, $\mathcal{G}^{\gamma}(X) < \infty$, which means that, $\mathcal{F}(X) - \mathcal{G}^{\gamma}(X) \geqslant 0$. Adding $rf(\mathcal{G}^{\gamma})$ to both sides of the inequality we get, $\mathcal{F}(X) + rf(\mathcal{G}^{\gamma}) - \mathcal{G}^{\gamma}(X) \geqslant rf(\mathcal{G}^{\gamma})$. By Property 3, we have, $\mathcal{F}(X) + \overline{\mathcal{G}}^{\gamma}(X) \geqslant rf(\mathcal{G}^{\gamma})$ which clearly means that, $MaxSAT(\mathcal{F} \cup \overline{\mathcal{G}}^{\gamma}) \geqslant rf(\mathcal{G}^{\gamma})$.

Let us prove now the only if direction. $MaxSAT(\mathcal{F} \cup \overline{\mathcal{G}}^{\gamma}) \geqslant rf(\mathcal{G}^{\gamma})$ implies that $\mathcal{F}(X) + \overline{\mathcal{G}}^{\gamma}(X) \geqslant rf(\mathcal{G}^{\gamma})$ for all X. Moreover, since $\overline{\mathcal{G}}^{\gamma}$ does not have hard clauses, from Property 3 we know that, $\mathcal{F}(X) + rf(\mathcal{G}^{\gamma}) - \mathcal{G}^{\gamma}(X) \geqslant rf(\mathcal{G}^{\gamma})$ so we have that $\mathcal{F}(X) \geqslant \mathcal{G}^{\gamma}(X)$ and we need to prove that, $\mathcal{F}(X) \geqslant \mathcal{G}(X)$. There are two possibilities for $\mathcal{G}^{\gamma}(X)$,

1. If $\mathcal{G}^{\gamma}(X) < \gamma$ it means that X does not falsify any of the clauses that are hard in $\mathcal{G}$. Therefore, $\mathcal{G}^{\gamma}(X) = \mathcal{G}(X)$, which means that $\mathcal{F}(X) \geqslant \mathcal{G}(X)$.
2. If $\mathcal{G}^{\gamma}(X) \geqslant \gamma$, since $\mathcal{F}(X) \geqslant G^{\gamma}(X)$, then $\mathcal{F}(X) \geqslant \gamma$ which, by definition of γ, means that $\mathcal{F}(X) = \infty$. Therefore, $\mathcal{F}(X) \geqslant \mathcal{G}(X)$.

which proves the theorem.

Example 1 *Consider formulas $\mathcal{F} = \{(z, 2), (x, 5), (y, \infty)\}$ and $\mathcal{G} = \{(x \vee z, u), (y \vee z, \infty\}$ with u being a finite weight. We can apply Theorem 1 to find out whether $\mathcal{F} \models \mathcal{G}$.*

Clearly $\gamma = 8 > \max_{X\mid\mathcal{F}(X)<\infty} \mathcal{F}(X)$, so we define $\mathcal{G}^{\gamma} = \{(x \vee z, u), (y \vee z, 8)\}$, $rf(\mathcal{G}^{\gamma}) = u + 8$ and $\overline{\mathcal{G}}^{\gamma} = \{(\overline{x}, u), (x \vee \overline{z}, u), (\overline{y}, 8), (y \vee \overline{z}, 8)\}$. With $u = 5$ we have that $MaxSAT(\mathcal{F} \cup \overline{\mathcal{G}}^{\gamma}) = 13$ and $rf(\mathcal{G}^{\gamma}) = 13$ which implies that $\mathcal{F} \models \mathcal{G}$. However, with $u = 8$ we have $MaxSAT(\mathcal{F} \cup \overline{\mathcal{G}}^{\gamma}) = 15$ and $rf(\mathcal{G}^{\gamma}) = 16$ which implies that $\mathcal{F} \not\models \mathcal{G}$.

6. MaxSAT resolution-based Proof Systems

MaxSAT proof systems implicitly assume the following two self-explanatory inference rules:

$$\frac{(C, v) \quad (C, w)}{(C, v + w)} \qquad\qquad \frac{(C, w)}{(C, v) \quad (C, w - v)}$$
$$\texttt{(merge)} \qquad\qquad\qquad \texttt{(unmerge)}$$

where in the unmerge rule $w > 0$ and $v < w$.

In the following, we introduce and analyze the impact of three MaxSAT inference rules: *resolution*, *split* and *virtual*. After the definition of each rule, we discuss the completeness of the new proof system and what type of PHP problems it solves, which shows the incremental power of each proof system.

6.1 Resolution

The MaxSAT *resolution* rule [20] is

$$\frac{(x \vee A, w) \quad (\overline{x} \vee B, w)}{\begin{array}{c}(A \vee B, w) \\ (x \vee A, w - w) \quad (\overline{x} \vee B, w - w) \\ (x \vee A \vee \overline{B}, w) \quad (\overline{x} \vee B \vee \overline{A}, w)\end{array}}$$

where A and B are arbitrary (possibly empty) disjunctions of literals, $w > 0$. When A (resp. B) is empty, $\overline{A}$ (resp. $\overline{B}$) is constant true, so $x \vee \overline{A} \vee B$ (resp. $x \vee A \vee \overline{B}$) is tautological. When $w \neq \infty$ the antecedents will disappear because $w - w = 0$. When $w = \infty$ the antecedents are replaced by themselves because $\infty - \infty = \infty$ or, in other words, hard clauses remain throughout the proof as they do in classical SAT resolution, which means that they can be used as antecedents any number of times (see the refutation graph of Figure 3 (left)).

Example 2 *The application of MaxSAT resolution to* $(x \vee y \vee z, 1)$ *and* $(\neg x \vee y \vee p, 1)$ *corresponds to,*

$$\frac{(x \vee y \vee z, 1) \quad (\neg x \vee y \vee p, 1)}{\begin{array}{c}(y \vee z \vee p, 1) \\ (x \vee y \vee z, 0) \quad (\neg x \vee y \vee p, 0) \\ (x \vee y \vee z \vee \neg y, 1) \quad (\neg x \vee \neg y \vee y \vee p, 1) \\ (x \vee y \vee z \vee y \vee \neg p, 1) \quad (\neg x \vee y \vee \neg z \vee y \vee p, 1)\end{array}}$$

Removing zero-cost clauses, tautologies and repeated literals, the resulting set of clauses is $\{(y \vee z \vee p, 1), (x \vee y \vee z \vee \neg p, 1), (\neg x \vee y \vee \neg z \vee p, 1)\}$.

It is known that the proof system **Res** = { *resolution* } is sound and refutationally complete [9, 21]. However, as we show next, it is not complete.

Theorem 2 *Proof system **Res** is not complete.*

Proof. Consider formula $\mathcal{F} = \{(x, 1), (y, 1)\}$. It is clear that $\mathcal{F} \models (x \vee y, 1)$ which cannot be derived with **Res**.

It is known that **Res** cannot compute polynomial size refutations for *PHP* [16] or *SPHP* [9]. However, we show next that it can efficiently refute $SPHP^1$. We write it as a property because it will be instrumental in the proof of several results in the rest of this section. The refutation graph (which is a straightforward adaptation of what was proved in [18] and [22]) appears in Figure 5. The refutation uses the following Lemma.

Lemma 1 *Consider a MaxSAT formula* $\mathcal{F} = \{(x_1 \vee \ldots \vee x_{n-1}, 1)\} \cup \{(\overline{x}_i \vee \overline{x}_n, 1) \mid 1 \leqslant i < n\}$. *There is a proof*

$$\mathcal{F} \vdash_{Res} \{(\overline{x}_n, 1)\} \cup \{(x_1 \vee \ldots \vee x_n, 1)\} \ \cup \{(\overline{x}_i \vee \overline{x}_n \vee \overline{x_{i+1} \vee \ldots \vee x_{n-1}}, 1) \mid 1 \leqslant i < n-1\}$$

of length $n - 1$.

Proof. The resolution proceeds as shown in Figure 4.

Property 4 *There is a polynomial size **Res** refutation of* $SPHP^1$.

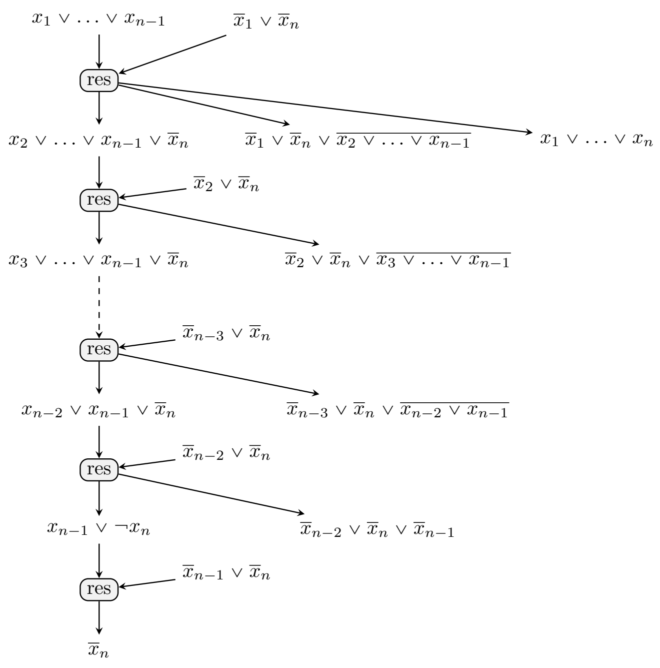

$x_1 \vee \ldots \vee x_{n-1}$
$\overline{x}_1 \vee \overline{x}_n$
res
$x_2 \vee \ldots \vee x_{n-1} \vee \overline{x}_n$
$\overline{x}_1 \vee \overline{x}_n \vee \overline{x_2 \vee \ldots \vee x_{n-1}}$
$x_1 \vee \ldots \vee x_n$
$\overline{x}_2 \vee \overline{x}_n$
$x_3 \vee \ldots \vee x_{n-1} \vee \overline{x}_n$
$\overline{x}_2 \vee \overline{x}_n \vee \overline{x_3 \vee \ldots \vee x_{n-1}}$
$\overline{x}_{n-3} \vee \overline{x}_n$
$x_{n-2} \vee x_{n-1} \vee \overline{x}_n$
$\overline{x}_{n-3} \vee \overline{x}_n \vee \overline{x_{n-2} \vee x_{n-1}}$
$\overline{x}_{n-2} \vee \overline{x}_n$
$x_{n-1} \vee \neg x_n$
$\overline{x}_{n-2} \vee \overline{x}_n \vee \overline{x}_{n-1}$
$\overline{x}_{n-1} \vee \overline{x}_n$
$\overline{x}_n$

Figure 4: Proof of Lemma 1. All clauses have cost 1.

Proof. The refutation is divided in two parts. First, for each one of the $m+1$ pigeons there is a derivation
$$\{(x_{i1} \vee x_{i2} \vee \ldots \vee x_{im}, 1)\} \cup \{(\overline{x}_{ij}, 1) | \ 1 \leqslant j \leqslant m\} \vdash_{Res} (\Box, 1)$$
Figure 5 (left) shows the derivation graph that corresponds to an arbitrary pigeon i. Second, for each one of the m holes there is a derivation
$$\{(\overline{x}_{ij} \vee \overline{x}_{i'j}, 1) \mid 1 \leqslant i < i' \leqslant m+1\} \cup \{(x_{ij}, 1) \mid 1 \leqslant i \leqslant m+1\} \vdash_{Res} \{(\Box, m)\}$$
Figure 5 (right) shows the derivation graph that corresponds to an arbitrary hole j.

Because each derivation is independent of the other, they can be done one after another, aggregating all the empty clauses, which produces
$$SPHP^1 \vdash_{Res} \{(\Box, m^2 + m + 1)\}$$
which is a refutation of $SPHP^1$. Observe that each pigeon proof has length $O(m)$ and each hole proof has length $O(m^2)$. Therefore, the length of the refutation is $O(m^3)$.

Property 5 *There is no polynomial size **Res** refutation of $SPHP^0$.*

Proof. Res cannot produce a polynomial size refutation for $SPHP^0$ because the resolution rule cannot be applied to the empty clause $(\Box, w)$, so it must remain unaltered during any derivation. If **Res** could refute *SPHP⁰* in polynomial time it would also refute *SPHP* in polynomial time, which is not the case [9].

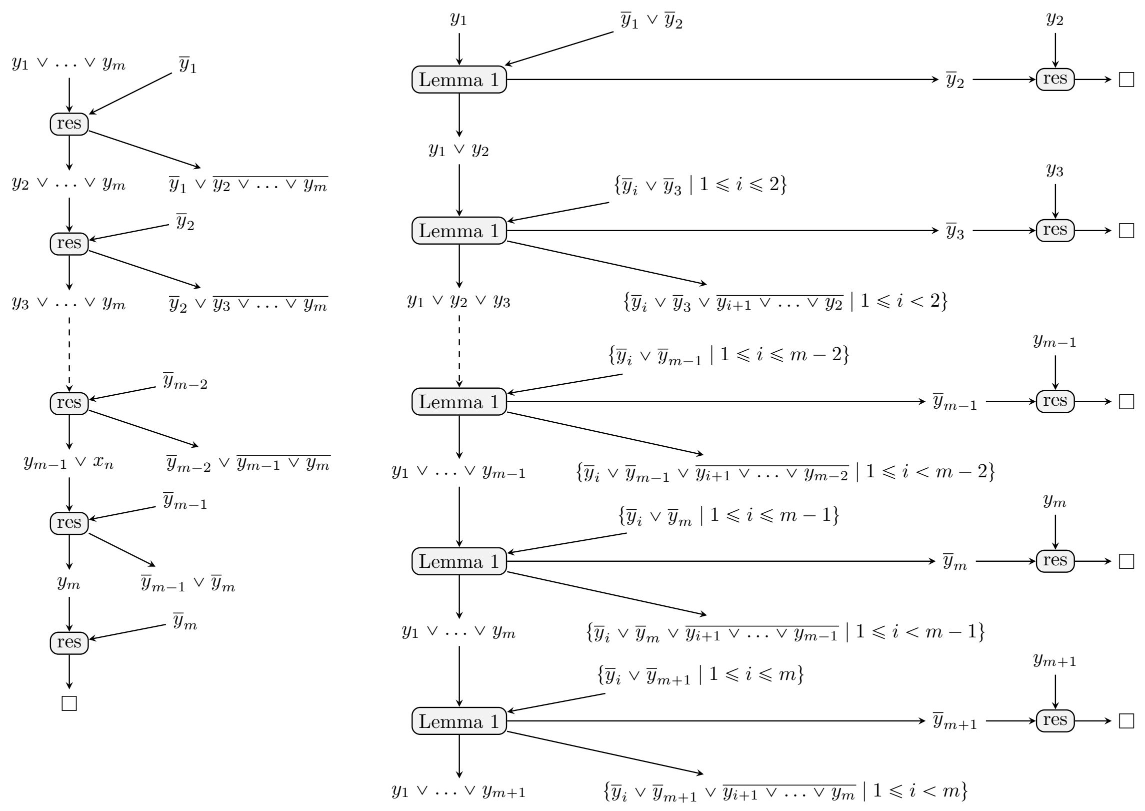

$y_1 \vee \ldots \vee y_m$
$\overline{y}_1$
res
$y_2 \vee \ldots \vee y_m$
$\overline{y}_1 \vee \overline{y_2 \vee \ldots \vee y_m}$
$\overline{y}_2$
res
$y_3 \vee \ldots \vee y_m$
$\overline{y}_2 \vee \overline{y_3 \vee \ldots \vee y_m}$
$\overline{y}_{m-2}$
res
$y_{m-1} \vee x_n$
$\overline{y}_{m-2} \vee \overline{y_{m-1} \vee y_m}$
$\overline{y}_{m-1}$
res
y_m
$\overline{y}_{m-1} \vee \overline{y}_m$
$\overline{y}_m$
res
$\Box$
y_1
$\overline{y}_1 \vee \overline{y}_2$
Lemma 1
$\overline{y}_2$
y_2
res
$\Box$
$y_1 \vee y_2$
$\{\overline{y}_i \vee \overline{y}_3 \mid 1 \leqslant i \leqslant 2\}$
Lemma 1
$\overline{y}_3$
y_3
res
$\Box$
$y_1 \vee y_2 \vee y_3$
$\{\overline{y}_i \vee \overline{y}_3 \vee \overline{y_{i+1} \vee \ldots \vee y_2} \mid 1 \leqslant i < 2\}$
$\{\overline{y}_i \vee \overline{y}_{m-1} \mid 1 \leqslant i \leqslant m-2\}$
Lemma 1
$\overline{y}_{m-1}$
y_{m-1}
res
$\Box$
$y_1 \vee \ldots \vee y_{m-1}$
$\{\overline{y}_i \vee \overline{y}_{m-1} \vee \overline{y_{i+1} \vee \ldots \vee y_{m-2}} \mid 1 \leqslant i < m-2\}$
$\{\overline{y}_i \vee \overline{y}_m \mid 1 \leqslant i \leqslant m-1\}$
Lemma 1
$\overline{y}_m$
y_m
res
$\Box$
$y_1 \vee \ldots \vee y_m$
$\{\overline{y}_i \vee \overline{y}_m \vee \overline{y_{i+1} \vee \ldots \vee y_{m-1}} \mid 1 \leqslant i < m-1\}$
$\{\overline{y}_i \vee \overline{y}_{m+1} \mid 1 \leqslant i \leqslant m\}$
Lemma 1
$\overline{y}_{m+1}$
y_{m+1}
res
$\Box$
$y_1 \vee \ldots \vee y_{m+1}$
$\{\overline{y}_i \vee \overline{y}_{m+1} \vee \overline{y_{i+1} \vee \ldots \vee y_m} \mid 1 \leqslant i < m\}$

Figure 5: Left: derivation graph corresponding to pigeon i (for clarity purposes, we rename each x_{ij}, $1 \leqslant j \leqslant m$, to y_j). Right: derivation graph corresponding to hole j (for clarity purposes, we rename each variable x_{ij}, $1 \leqslant i \leqslant m+1$, to y_i.). All clauses have cost 1.

6.2 Split

The *split* rule,

$$\frac{(A, w)}{(A \vee x, w) \quad (A \vee \overline{x}, w)}$$

where $w > 0$, is the natural extension of its SAT counterpart.

Theorem 3 *The split rule is sound.*

Proof. We have to prove that $\mathcal{F} \cup \{(A, w)\} \equiv \mathcal{F} \cup \{(A \vee x, w), (A \vee \overline{x}, w)\}$ Consider an arbitrary truth assignment. If it satisfies A, then it also satisfies $A \vee x$ and $A \vee \overline{x}$ so the the cost of the truth assignment is the same before and after the split. If the truth assignment does not satisfy A, then there is a cost of w caused by A. After the application of the split the same cost will be caused either by $A \vee x$ or by $A \vee \overline{x}$ depending on whether the truth assignment satisfies x or not.

The proof system **ResS** = { *resolution, split* } is sound and complete.

Theorem 4 *Proof system **ResS** is sound.*

Proof. We have to prove that $\mathcal{F} \vdash_{ResS} \mathcal{G}$ implies $\mathcal{F} \models \mathcal{G}$. Because resolution and split are sound, $\mathcal{F} \vdash_{ResS} \mathcal{G}$ implies that there is a derivation $\mathcal{F} \vdash_{ResS} \mathcal{G} \cup \mathcal{H}$ for some $\mathcal{H}$ such that $\forall X, \mathcal{F}(X) = \mathcal{G}(X) + \mathcal{H}(X)$. Therefore, $\mathcal{F} \models \mathcal{G}$, which completes the proof.

Theorem 5 *Proof system **ResS** is complete.*

Proof. We have to prove that if $\mathcal{F} \models \mathcal{G}$ then there is derivation $\mathcal{F} \vdash_{ResS} \mathcal{G}$. The proof is based on the following two facts:

1. For every formula $\mathcal{F}$ there is a derivation $\mathcal{F} \vdash_{ResS} \mathcal{F}^{ext}$ made exclusively of splits and merges such that: *i)* $\mathcal{F} \equiv \mathcal{F}^{ext}$, *ii)* all the clauses of $\mathcal{F}^{ext}$ contain all the variables in the formula and *iii)* there are no repeated clauses. In the derivation each clause $(C, w) \in \mathcal{F}$ can be expanded to a new variable not in C using the split rule. The process is repeated until all clauses in the current formula contain all the variables in the formula. Then, pairs of equal clauses (C', u), (C', v) are merged and, thus, $\mathcal{F}^{ext}$ does not contain repeated clauses. As a result, $\mathcal{F}^{ext}$ contains one clause (C, w) for each $\mathcal{F}^{ext}(X) = w > 0$, where C is falsified exactly by X.

2. If there is a derivation $\mathcal{F} \vdash_{ResS} \mathcal{F}^{ext}$ made exclusively of splits and merges, then there is a derivation $\mathcal{F}^{ext} \vdash_{ResS} \mathcal{F}$ made exclusively of resolutions and unmerges. Let $\Pi = (\mathcal{F}_0 = \mathcal{F}; \mathcal{F}_1; \ldots; \mathcal{F}_p = \mathcal{F}_{ext})$ be the first derivation. Then, the later derivation is $\Gamma = (\mathcal{F}_{ext} = \mathcal{F}_p; \mathcal{F}_{p-1}; \ldots; \mathcal{F}_0)$ where $\mathcal{F}_i; \mathcal{F}_{i-1} \in \Gamma$ is an unmerge if $\mathcal{F}_{i-1}; \mathcal{F}_i \in \Pi$ is a merge; and $\mathcal{F}_i; \mathcal{F}_{i-1} \in \Gamma$ is a resolution if $\mathcal{F}_{i-1}; \mathcal{F}_i \in \Pi$ is an split.

From fact (1) we know that $\mathcal{F} \vdash_{ResS} \mathcal{F}^{ext}$. Since $\mathcal{F} \models \mathcal{G}$ we know $\mathcal{F}^{ext} \models \mathcal{G}^{ext}$. We can separate $\mathcal{F}^{ext}$ as $\mathcal{F}^{ext} = \mathcal{G}^{ext} \cup \mathcal{H}$. From fact (1) and (2) we know that $\mathcal{G}^{ext} \vdash_{ResS} \mathcal{G}$. Joining the two derivations we have $\mathcal{F} \vdash_{ResS} \mathcal{F}^{ext} \vdash_{ResS} \mathcal{H} \cup \mathcal{G}$, which proves the theorem.

We show now which pigeon problems **ResS** can and cannot solve.

Property 6 *There is a polynomial size **ResS** refutation for $SPHP^0$.*

Proof. **ResS** can produce a polynomial size refutation for $SPHP^0$ because it can transform $SPHP^0$ into $SPHP^1$ and then apply Property 4. The transformation is done by a sequence of splits,

$$\frac{(\square, 1)}{(x_{ij}, 1) \quad (\overline{x}_{ij}, 1)}$$

that move one unit of weight from the empty clause to every variable in the formula and its negation.

Property 7 *There is no polynomial size **ResS** refutation for PHP.*

Proof. ResS with hard formulas corresponds to the SAT proof system containing SAT resolution and SAT split. From Property 2, we know that it is equivalent to the SAT proof system containing only resolution. Therefore, the existence of a polynomial size **ResS** refutation for PHP would imply the existence of a polynomial size refutation with SAT resolution, which is not possible [25].

Property 8 *There is no polynomial size **ResS** refutation for $SPHP$.*

Proof. We show that we can build a **ResS** refutation for PHP from a **ResS** refutation for $SPHP$ without increasing its length. Therefore, a polynomial size refutation for $SPHP$ would imply a polynomial size refutation for PHP, which is a contradiction to Property 7.

Let $\Pi = (\mathcal{F}_0; \mathcal{F}_1; \ldots; \mathcal{F}_e)$ with $SPHP = \mathcal{F}_0$ and $(\Box, 1) \in \mathcal{F}_e$ be the refutation and $G(\Pi)$ its associated graph. We are going to transform $G(\Pi)$ into a PHP refutation following the derivation steps. First, replace weight 1 by ∞ in all the zero in-neighbors clauses (namely, original clauses). Then follow the refutation step by step. If the inference step is a split, just replace the weight of the consequents by infinity. If the inference is a resolution between $x \vee A$ and $\overline{x} \vee B$, merge nodes $\{A \vee B, x \vee A \vee \overline{B}, \overline{x} \vee \overline{A} \vee B\}$ into $A \vee B$, and replace the weight of all the consequents by infinity. By construction, when considering any inference step all its in-neighbors will already have infinity weight making the graph correct. At the last step, node $(\Box, 1)$ will be transformed into $(\Box, \infty)$ making the graph a PHP refutation.

A consequence of the previous results is that, unlike what happens in the SAT case (see Property 2), **ResS** is stronger than **Res**,

Theorem 6 ***ResS** is stronger than **Res**.*

Proof. On the one hand, it is clear that **ResS** can p-simulate any proof of **Res** since it is a superset of **Res**. On the other hand **Res** cannot p-simulate **ResS** because there is a polynomial size **ResS** refutation of $SPHP^0$ which cannot exist for **Res**.

Finally, we show that, similarly to what happens in the SAT case, the split rule allows to restrict the use of resolution to its symmetric form (this result will be useful in Section 7.3). The *symmetric resolution* rule,

$$\frac{(A \vee x, w) \quad (A \vee \overline{x}, w)}{\begin{array}{c}(A, w) \\ (A \vee x, w - w) \quad (A \vee \overline{x}, w - w)\end{array}}$$

is the natural extension of its SAT counterpart. In combination with split, symmetric resolution already guarantees completeness.

Property 9 *The MaxSAT resolution rule can be replaced by $O(n)$ splits and one symmetric resolution, where n is the number of variables in the formula.*

Proof. Consider clauses $(x \vee A, u)$ and $(\overline{x} \vee B, u)$. $|C - A|$ splits transform the first clause into $\{(x \vee A \vee B, u), (x \vee A \vee \overline{B}, u)\}$. Similarly, $|C - B|$ splits transform the second clause into $\{(\overline{x} \vee A \vee B, u), (\overline{x} \vee B \vee \overline{A}, u)\}$. Finally, it is possible to apply symmetric resolution between $(x \vee A \vee B, u)$ and $(\overline{x} \vee A \vee B, u)$, which proves our claim.

6.3 Virtual and 0-Virtual

Now we introduce our third and last rule, *virtual*, and show that it can further speed-up refutations. Roughly speaking, it allows to anticipate weighted clauses that will be derived later on and use them right away. Any derivation obtained from this anticipated clauses will be sound as long as the anticipation turns out to be true. The *virtual* rule is,

$$\frac{}{(A, w) \quad (A, -w)}$$

with $w \neq \infty$. It allows to introduce a fresh clause (A, w) into the formula. To preserve soundness (i.e, cancel out the effect of the addition) it also adds $(A, -w)$. The use of virtual requires to allow clauses with negative finite weights [2].

Theorem 7 *The virtual inference rule is sound.*

Proof. We have to prove that the cost of any truth assignment is the same for $\mathcal{F}$ and $\mathcal{F} \cup \{(A, w), (A, -w)\}$. If the truth assignment satisfies A, then the new clauses are also satisfied and they do not affect its cost. If the truth assignment does not satisfy A, the cost will be increased by w because of the first clause and decreased by w because of the second clause, which leaves the total cost unaltered.

Let **ResSV** = { *resolution, split, virtual* }. Recall that resolution and split were only defined for antecedents with positive weights and we keep this restriction in the **ResSV** proof system. Therefore, they can use as an antecedent positive clauses introduced by virtual, but not the negative clauses.

The following theorem indicates that proof system **ResSV** is sound, but it requires to adapt the definition of $\vdash$ to the existence of negative weights. In Section 5 we introduced $\mathcal{F} \vdash_S \mathcal{G}$ to denote an arbitrary proof $\mathcal{F}; \mathcal{F}_1; \ldots; \mathcal{H}$ with $\mathcal{G} \subseteq \mathcal{H}$ under proof system **S**, and defined the soundness of **S** using that notation. Because the virtual rule introduces negative weights, this definition needs to be revised. To see why, consider a one step derivation $\{\}; \{(\square, -1), (\square, 1\}$ that only applies the virtual rule. According to that definition in Section 5, $\{\} \vdash_{ResSV} (\square, 1)$. However, $\{\}$ corresponds to constant zero and $(\square, 1)$ corresponds to constant 1 and it is false that $0 \geqslant 1$ (i.e., $\{\} \not\models (\square, 1)$). We solve this problem by defining $\vdash_{\mathbf{S}}$ considering the possible existence of clauses with negative weight during a derivation.

Definition 1 ($\vdash$) *$\mathcal{F} \vdash_S \mathcal{G}$ denotes an arbitrary **S** proof $\mathcal{F}; \ldots; \mathcal{H}$ with $\mathcal{G} \subseteq \mathcal{H}$ and all the clauses in $\mathcal{H}$ having positive weights*

Note that this new definition does not affect the other proof systems **Res** and **ResS** because they always deal with positive weights.

Theorem 8 *Proof system **ResSV** is sound.*

Proof. We have to prove that $\mathcal{F} \vdash_{ResSV} \mathcal{G}$ implies $\mathcal{F} \models \mathcal{G}$. Consider an arbitrary derivation $\mathcal{F} \vdash_{ResSV} \mathcal{G}$. By definition of $\vdash_{ResSV}$, $\mathcal{F} \vdash_{ResSV} \mathcal{G}$ means $\mathcal{F}; \ldots; \mathcal{H}$ where $\mathcal{G} \subseteq \mathcal{H}$ and all clauses in $\mathcal{H}$ have positive weight. Because resolution, split and virtual are sound, we have that $\mathcal{F}(X) = \mathcal{G}(X) + \mathcal{R}(X)$, where $\mathcal{H} = \mathcal{G} \cup \mathcal{R}$. Therefore $\mathcal{F}(X) \geqslant \mathcal{G}(X)$, which completes the proof.

Figure 3 (right) shows a refutation graph with **ResSV**. Note that the refutation is correct since all nodes with no out-neighbours have positive weight.

The intuition behind the virtual rule and its soundness theorem is that the rule introduces hypothetical clauses that can be temporarily used to derive new knowledge, but this new knowledge is valid only if the proof manages to cancel out the clauses with negative weight. Since negative clauses cannot be manipulated by inference rules, one way to interpret them is like a *reminder* of what needs to be re-derived to make the proof sound.

2. Note that the virtual rule can be seen as a generalization of the unmerge rule. Here we prefer to define it as an independent rule for clarity purposes.

Next, we discuss the completeness of **ResSV**. Note that completeness of **ResSV** is obvious since **ResS** is complete, so we can just ignore the virtual rule in any **ResSV** proof. However, a related and more interesting question is whether the use of the virtual rule can take an ongoing proof to a state from which the objective formula cannot be derived. If that was the case, the practical use of **ResSV** would be jeopardized. The following theorem shows that this is not the case. No matter which are the first inference steps, we can always proceed with the derivation, get rid of the negative clauses introduced by the virtual rule, and end up deriving any entailed formula. To prove that, we find useful the following lemma.

Lemma 2 *There is a* ***ResSV*** *proof* $\mathcal{F}; \ldots; \mathcal{F} \cup \{(\Box, -w), (C, w), (\overline{C}, w)\}$ *for any formula* $\mathcal{F}$*, clause* C *and weight* $0 < w$*.*

Proof. Let $C = l_1 \vee l_2 \vee \cdots \vee l_r$. The derivation is done by first introducing $(\Box, w)$ and $(\Box, -w)$ with the virtual rule, followed by a sequence of r splits,

$$\begin{array}{l}\mathcal{F}; \mathcal{F} \cup \{(\Box, -w), (\Box, w)\}; \\ \mathcal{F} \cup \{(\Box, -w), (\bar{l}_1, w), (l_1, w)\}; \\ \mathcal{F} \cup \{(\Box, -w), (\bar{l}_1, w), (l_1 \vee \bar{l}_2, w), (l_1 \vee l_2, w)\}; \\ ; \ldots ; \\ \mathcal{F} \cup \{(\Box, -w), (\bar{l}_1, w), (l_1 \vee \bar{l}_2, w), \ldots, \\ (l_1 \vee l_2 \vee \ldots \vee \bar{l}_r, w), (l_1 \vee l_2 \vee \ldots \vee l_r, w)\}\end{array}$$

By Property 1, the last element in the derivation is equivalent to

$$\mathcal{F} \cup \{(\Box, -w), (\overline{C}, w), (C, w)\}$$

Theorem 9 *Consider formulas* $\mathcal{F}$ *and* $\mathcal{G}$ *such that* $\mathcal{F} \models \mathcal{G}$*, and a* ***ResSV*** *proof* $\mathcal{F}; \mathcal{F}_1; \ldots; \mathcal{F}_i$*. There is a proof* $\mathcal{F}_i \vdash_{ResSV} \mathcal{G}$*.*

Proof. Let $\mathcal{N} \subseteq \mathcal{F}_i$ be the set of clauses with negative weights. If $\mathcal{N} = \varnothing$ then completeness follows trivially from the completeness of **ResS**. Otherwise, for each $(C, -w) \in \mathcal{N}$ we add (using the previous lemma) $\{(\Box, -w), (C, w), (\overline{C}, w)\}$. After clause merging, $\mathcal{F}_i; \ldots; \{(\Box, -r)\} \cup \mathcal{F}'_j$ with $-r = \sum_{(C,-w) \in \mathcal{N}} -w$ being a negative number and $\mathcal{F}'_j$ contains only positive weights because each $(C, -w)$ vanishes when aggregating (C, w).

Since the three inference rules in **ResSV** are sound, we have that $\mathcal{F}(X) = -r + \mathcal{F}'_j(X)$, which implies that $\mathcal{F}'_j(X) \geqslant \mathcal{F}(X)$. Together with $\mathcal{F}(X) \geqslant \mathcal{G}(X)$, they imply $\mathcal{F}'_j(X) \geqslant \mathcal{G}(X) + r$, which means that $\mathcal{F}'_j \models \mathcal{G} \cup \{(\Box, r)\}$. Since **ResS** is complete, $\mathcal{F}'_j \vdash_{ResS} \mathcal{G} \cup \{(\Box, r)\}$ (i.e., $\mathcal{F}'_j; \ldots; \mathcal{G} \cup \{(\Box, r)\} \cup \mathcal{H}$ where all clauses in $\mathcal{G} \cup \mathcal{H}$ have positive weights). Joining the two facts,

$$\mathcal{F}_i; \ldots; \{(\Box, -r)\} \cup \mathcal{F}'_j; \ldots; \{(\Box, -r)\} \cup \mathcal{G} \cup \{(\Box, r)\} \cup \mathcal{H}$$

After merging $(\Box, -r)$ and $(\Box, r)$, the previous derivation can be written as $\mathcal{F}_i \vdash_{ResSV} \mathcal{G}$.

Property 10 *There is a polynomial size* ***ResSV*** *refutation of* $SPHP$*.*

Proof. First, for each variable x_{ij} in $SPHP$ we introduce clauses $(x_{ij}, 1)$, $(x_{ij}, -1)$ and $(\overline{x}_{ij}, 1)$, $(\overline{x}_{ij}, -1)$ thanks to the virtual rule. As a consequence,

$$SPHP; \ldots; SPHP^1 \cup \{(x_{ij}, -1), (\overline{x}_{ij}, -1) \mid 1 \leqslant i \leqslant m+1, 1 \leqslant j \leqslant m\}$$

Since, by Property 4, there exists a proof $SPHP^1;\dots;\mathcal{G} \cup \{(\square, m^2+m+1)\}$ where all clauses in $\mathcal{G}$ have positive weights, then

$$SPHP;\dots;\mathcal{G} \cup \{(\square, m^2+m+1)\} \cup \{(x_{ij}, -1), (\overline{x}_{ij}, -1) \mid 1 \leqslant i \leqslant m+1, 1 \leqslant j \leqslant m\}$$

Finally, clause $(\square, m^2+m+1)$ is unmerged to m^2+m+1 clauses $(\square, 1)$ and each $(\square, 1)$ is split to one pair $(x_{ij}, 1), (\overline{x}_{ij}, 1)$. Since there are m^2+m variables, one clause $(\square, 1)$ still remains. That is,

$$\begin{aligned} SPHP;\dots;\mathcal{G} \cup \{(\square, 1)\} \cup \{(x_{ij}, 1), (\overline{x}_{ij}, 1) \mid 1 \leqslant i \leqslant m+1, 1 \leqslant j \leqslant m\} \cup \\ \cup \{(x_{ij}, -1), (\overline{x}_{ij}, -1) \mid 1 \leqslant i \leqslant m+1, 1 \leqslant j \leqslant m\} \end{aligned}$$

After merging clauses with positive and negative weights,

$$SPHP \vdash_{ResSV} \{(\square, 1)\}$$

The length of the refutation is $O(m^3)$.

The main consequence of the previous property is that **ResSV** is stronger than **ResS**,

Theorem 10 ***ResSV*** *is stronger than* ***ResS****.*

Proof. On the one hand, **ResSV** p-simulates **ResS** since it is a superset of **ResS**. On the other hand, **ResSV** can produce a polynomial size refutation of $SPHP$, while **ResS** cannot.

We show next that the applications of the virtual rule of **ResSV** can be all done as a first phase of the derivation, leaving a second phase with only splits and resolutions. This property will be used in Theorem 11, Theorem 16 and Theorem 20.

Property 11 *If there is a proof $\mathcal{F} \vdash_{ResSV} \mathcal{G}$ then there is a shorter proof $\mathcal{F} \cup \mathcal{B} \vdash_{ResS} \mathcal{G} \cup \mathcal{B}$ where $\mathcal{B}$ is the set of clauses with positive weight added by the virtual rule in the original* ***ResSV*** *proof.*

Proof. Let e be the length of $\mathcal{F} \vdash_{ResSV} \mathcal{G}$ and let $\mathcal{N} = \{(A, -u) \mid (A, u) \in \mathcal{B}\}$. Since the virtual rule has no antecedents, all its applications can be done at the beginning of the derivation. That is, $\mathcal{F} \vdash_{Virtual} \mathcal{F} \cup \mathcal{B} \cup \mathcal{N} \vdash_{ResS} \mathcal{G}$. The length of $\mathcal{F} \cup \mathcal{B} \cup \mathcal{N} \vdash_{ResS} \mathcal{G}$ is $e' \leqslant e$. Note that $\mathcal{G}$ does not have clauses with negative weights, which means that during the derivation clauses in $\mathcal{N}$ have been cancelled out with the symmetric set $\mathcal{B}$. Therefore, we could have skipped the cancellations having derivation $\mathcal{F} \cup \mathcal{B} \cup \mathcal{N} \vdash_{ResS} \mathcal{G} \cup \mathcal{B} \cup \mathcal{N}$ with the same length e'. Note that in this derivation the set $\mathcal{N}$ is only carried along the proof, therefore there is a derivation $\mathcal{F} \cup \mathcal{B} \vdash_{ResS} \mathcal{G} \cup \mathcal{B}$ with length $e' \leqslant e$, which completes the proof.

We close this section showing that a restricted version of **ResSV** in which the virtual rule is only used to add empty clauses does not lose any proving power. Consider proof system **ResSV**^{0}= { *resolution, split, 0-virtual* } where 0-virtual is,

$$\frac{}{(\square, w) \quad (\square, -w)}$$

Theorem 11 ***ResSV*** *and* ***ResSV***^{0} *p-simulate each other.*

Proof. **ResSV** trivially p-simulates $\mathbf{ResSV}^0$ since it is more general than $\mathbf{ResSV}^0$. So we have to show that if there is a proof $\mathcal{F} \vdash_{ResSV} \mathcal{G}$ of length e, then there is another proof $\mathcal{F} \vdash_{ResSV^0} \mathcal{G}$ of length bounded by $p(e)$ where p is a polynomial.

Let $\mathcal{B}$ be the set of clauses with positive weight added by the virtual rule in the original **ResSV** derivation. First, we show that there is a $\mathbf{ResSV}^0$ derivation $\mathcal{F} \vdash_{ResSV^0} \mathcal{F} \cup \mathcal{B} \cup \overline{\mathcal{B}} \cup \{(\Box, -w)\}$ where $w = \sum_{(C,u)\in\mathcal{B};u>0} u$. It is obtained by first adding $\{(\Box, w), (\Box, -w)\}$ using the 0-virtual rule and then applying unmerge and splits over $(\Box, w)$ to obtain $\mathcal{B} \cup \overline{\mathcal{B}}$.

By property 11, we know that $\mathcal{F} \vdash_{ResSV} \mathcal{G}$ implies the existence of a derivation $\mathcal{F} \cup \mathcal{B} \vdash_{ResS} \mathcal{G} \cup \mathcal{B})$ of length at most e. Joining both proofs we have $\mathcal{F} \vdash_{ResSV^0} \mathcal{G} \cup \mathcal{B} \cup \overline{\mathcal{B}} \cup \{(\Box, -w)\}$.

Finally, resolving over clauses in $\mathcal{B} \cup \overline{\mathcal{B}}$ we obtain $(\Box, w)$ which merged with $(\Box, -w)$ cancels out. As a result, $\mathcal{F} \vdash_{ResSV^0} \mathcal{G}$, which completes the proof.

7. MaxSAT Proof Systems for SAT Refutations

Refuting unsatisfiable SAT formulas is one of the central topics of proof complexity. In this Section we will study the power of **ResSV** for SAT refutations showing its equivalence to two recently proposed proof systems and discussing its relation with a third one. To do that, we show how Theorem 1 provides a weaker definition of SAT refutation in the MaxSAT framework that we can take advantage from.

7.1 ResSV Refutations

Refuting a SAT formula $\mathcal{F}$ with a MaxSAT proof system corresponds to a derivation $\mathcal{F} \models \{(\Box, \infty)\}$, with $\mathcal{F}$ being a set of hard clauses. The following theorem shows that **ResSV** cannot refute the PHP efficiently,

Theorem 12 *There is no polynomial size proof* $PHP \vdash_{ResSV} (\Box, \infty)$.

Proof. By definition, the virtual rule cannot introduce hard clauses. Resolution and split only produce new hard consequents if their antecedents are hard. Therefore, $(\Box, \infty)$ can only be obtained by resolving or splitting hard clauses in PHP. Consequently, if there is a polynomial size **ResSV** refutation $PHP \vdash_{ResSV} (\Box, \infty)$, then it is a polynomial size **ResS** refutation $PHP \vdash_{ResS} (\Box, \infty)$, which contradicts Property 7.

However, Theorem 1 gives us a simpler way to refute SAT formulas. For this particular case, the theorem says that $\mathcal{F} \models \{(\Box, \infty)\}$ iff $\mathcal{F} \models (\Box, 1)$ or, in words, that we can refute a SAT formula by just deriving one soft inconsistency. The following two theorems show that **ResSV** can do such refutation efficiently while **ResS** cannot. Thus, **ResSV** is stronger than **ResS** for SAT refutations.

Theorem 13 *There is a polynomial size proof* $PHP \vdash_{ResSV} (\Box, 1)$.

Proof. We only need to apply the virtual rule,

$$\frac{}{(\Box, m^2 + m) \quad (\Box, -m^2 - m)}$$

and then split,

$$\frac{(\Box, 1)}{(x_{ij}, 1) \quad (\overline{x}_{ij}, 1)}$$

for each i, j. Then, we can unmerge each (hard) clause of PHP extracting weight one. The resulting problem is $PHP \cup PHP^1$. At this point the proof of Property 4 shows that we can derive $(\square, m^2 + m + 1)$ which cancels out the negative weight while still retaining $(\square, 1)$.

Theorem 14 *There is no polynomial size proof $PHP \vdash_{ResS} (\square, 1)$.*

Proof. By definition, resolution and split over hard clauses only produce new hard consequents. As a consequence, the existence of a polynomial size proof $PHP \vdash_{ResS} (\square, 1)$ would imply the existence of a polynomial size proof $PHP \vdash_{ResS} (\square, \infty)$. However, this is impossible because it contradicts Property 7.

Corollary 1 ***ResSV*** *is stronger than* ***ResS*** *for SAT refutations.*

Proof. It follows from Theorem 13 and Theorem 14.

Accordingly, in the rest of this Section we will refute SAT formulas with **ResSV** by just deriving $(\square, 1)$.

7.2 Dual Rail Encoding

The (weighted) *dual rail encoding* [18, 6, 7] transforms a SAT formula $\mathcal{F}$ over variables $\mathcal{X} = \{x_1, \ldots, x_s\}$ into a MaxSAT formula $\mathcal{F}^{dr}$ over variables $N = \{n_1, \ldots, n_s\}$ and $P = \{p_1, \ldots, p_s\}$. The dual encoding of clause $C \in \mathcal{F}$ is a hard clause (C^{dr}, ∞) in which each positive literal x_i in C is replaced by $\overline{n}_i$, and each negative literal $\overline{x}_i$ in C is replaced by $\overline{p}_i$. Additionally, for each variable x_i the dual encoding adds three new clauses: (p_i, w_i), (n_i, w_i) and $(\overline{p}_i \vee \overline{n}_i, \infty)$. The resulting MaxSAT formula $\mathcal{F}^{dr}$ is made exclusively of horn clauses, where only unit clauses are soft. We define $w = \sum_{i=1}^{s} w_i$ as the sum of weights, which are arbitrary positive integers of polynomial size. It is shown that $\mathcal{F}$ is satisfiable iff $w = MaxSAT(\mathcal{F}^{dr})$. They also show that $w \leqslant MaxSAT(\mathcal{F}^{dr})$. Accordingly, a *Dual Rail MaxSAT refutation* of $\mathcal{F}$ is defined as a derivation $\mathcal{F}^{dr} \vdash_{Res} \{(\square, w + 1)\}$.

They show that there is a polynomial size proof $PHP^{dr} \vdash_{Res} \{(\square, w + 1)\}$ which indicates that the dual rail encoding makes the PHP tractable [3] and, therefore, dominates the SAT resolution proof system. In their work it is not clear which of the dual rail ingredients (e.g. horn clauses, unit cost soft clauses, renaming,...) if not all, are really needed for this domination. We show that **ResSV** and dual encoding with split p-simulate each other, which indicates that the advantage of dual rail refutations over **ResS** refutations comes from the introduction of the unit soft clauses. For the purpose of proving our results, in this section we will allow MaxSAT formulas to have tautological clauses.

Let C^{dr} be a clause over dual rail variables p_i and n_i. Its *decoding* is the replacement of each occurrence of literals p_i and $\overline{n}_i$ by x_i and each occurrence of literals $\overline{p}_i$ and n_i by $\overline{x}_i$. The decoding of a formula over variables p_i and n_i is the decoding of all its clauses. For instance, let $\mathcal{F}^{dr}$ be the dual rail encoding of formula $\mathcal{F}$ over variables $\mathcal{X}$. Then, the decoding of $\mathcal{F}^{dr}$ is $\mathcal{F} \cup \{(x_i, w_i), (\overline{x}_i, w_i) \mid x_i \in \mathcal{X}\} \cup \{(x_i \vee \overline{x}_i, \infty) \mid x_i \in \mathcal{X}\}$. Note that the decoding of a dual rail formula has tautological clauses.

The following lemma shows that the renaming of the dual rail encoding does not have any advantage when using the **Res** proof system.

Lemma 3 *Let $\mathcal{F}^{dr}$ be the dual-rail encoding of hard MaxSAT formula $\mathcal{F}$ over variables $\mathcal{X}$. If there is a derivation $\mathcal{F}^{dr} \vdash_{ResS} \mathcal{M}^{dr}$, then there is also a derivation $\mathcal{F} \cup \{(x_i, w_i), (\overline{x}_i, w_i) \mid x_i \in \mathcal{X}\} \cup \{(x_i \vee \overline{x}_i, \infty) \mid x_i \in \mathcal{X}\} \vdash_{ResS} \mathcal{M}$ of the same length where $\mathcal{M}$ is the dual rail decoding of $\mathcal{M}^{dr}$.*

3. the refutation is very similar to the proof of Property 4

Proof. Let $\Pi^{dr} = (\mathcal{F}^{dr} = \mathcal{F}_0^{dr}; \mathcal{F}_1^{dr}; \ldots; \mathcal{F}_e^{dr} = \mathcal{M}^{dr})$ be a **ResS** derivation. We are going to prove that there is a **ResS** derivation $\Pi = (\mathcal{F}_0; \ldots; \mathcal{F}_e = \mathcal{M}_e)$ with $\mathcal{F}_0 = \mathcal{F} \cup \{(x_i, w_i), (\overline{x}_i, w_i) \mid x_i \in \mathcal{X}\} \cup \{(x_i \vee \overline{x}_i, \infty) \mid x_i \in \mathcal{X}\}$ where for each $\mathcal{F}_j^{dr}$ there is a $\mathcal{F}_j$ which is its dual decoding. We proceed by induction,

Base case (j = 0): let $\mathcal{F}_0 = \mathcal{F} \cup \{(x_i, w_i), (\overline{x}_i, w_i) \mid x_i \in \mathcal{X}\} \cup \{(x_i \vee \overline{x}_i, \infty) \mid x_i \in \mathcal{X}\}$, which is the dual decoding of $\mathcal{F}_0^{dr}$.

Inductive step: Let $\mathcal{A}^{dr} \subseteq \mathcal{F}_{j-1}^{dr}$ and $\mathcal{C}^{dr} \subseteq \mathcal{F}_j^{dr}$ be the set of antecedent and consequent clauses in step $\mathcal{F}_{j-1}^{dr}; \mathcal{F}_j^{dr}$, respectively. Therefore, $\mathcal{A}^{dr}; \mathcal{C}^{dr}$ and $\mathcal{F}_j^{dr} = (\mathcal{F}_{j-1}^{dr} \backslash \mathcal{A}^{dr}) \cup \mathcal{C}^{dr}$. By induction hypothesis, $\mathcal{F}_{j-1}$ is the dual-rail decoding of $\mathcal{F}_{j-1}^{dr}$ and, as a consequence, there exists a set $\mathcal{A} \subseteq \mathcal{F}_{j-1}$ which is the dual-rail decoding of $\mathcal{A}^{dr}$. We will show that the set of consequents $\mathcal{C}$ of step $\mathcal{A}; \mathcal{C}$ is the dual-rail decoding of $\mathcal{C}^{dr}$ and, since $\mathcal{F}_j = (\mathcal{F}_{j-1} \backslash \mathcal{A}) \cup \mathcal{C}$, $\mathcal{F}_j$ is the dual-rail decoding of $\mathcal{F}_j^{dr}$.

We analyze each possible step $\mathcal{A}^{dr}; \mathcal{C}^{dr}$:

- Resolution step:
  - over n_i: Let $\mathcal{A}^{dr} = \{(A^{dr} \vee n_i, u), (B^{dr} \vee \overline{n}_i, v)\}$, and $\mathcal{C}^{dr} = \{(A^{dr} \vee B^{dr}, m), (A^{dr} \vee n_i, u - m), (B^{dr} \vee \overline{n}_i, v - m), (A^{dr} \vee n_i \vee \overline{B}^{dr}, m), (B^{dr} \vee \overline{n}_i \vee \overline{A}^{dr}, m)\}$, where $m = min\{u, v\}$. Let $\mathcal{A} = \{(A \vee \overline{x}_i, u), (B \vee x_i, v)\}$. After resolution over $\mathcal{A}$, $\mathcal{C} = \{(A \vee B, m), (A \vee \overline{x}_i, u - m), (B \vee x_i, v - m), (A \vee \overline{x}_i \vee \overline{B}, m), (B \vee x_i \vee \overline{A}, m)\}$, which is the dual-rail decoding of $\mathcal{C}^{dr}$.
  - over p_i: Let $\mathcal{A}^{dr} = \{(A^{dr} \vee p_i, u), (B^{dr} \vee \overline{p}_i, v)\}$, and $\mathcal{C}^{dr} = \{(A^{dr} \vee B^{dr}, m), (A^{dr} \vee p_i, u - m), (B^{dr} \vee \overline{p}_i, v - m), (A^{dr} \vee p_i \vee \overline{B}^{dr}, m), (B^{dr} \vee \overline{p}_i \vee \overline{A}^{dr}, m)\}$, where $m = min\{u, v\}$. Let $\mathcal{A} = \{(A \vee x_i, u), (B \vee \overline{x}_i, v)\}$. After resolution over $\mathcal{A}$, $\mathcal{C} = \{(A \vee B, m), (A \vee x_i, u - m), (B \vee \overline{x}_i, v - m), (A \vee x_i \vee \overline{B}, m), (B \vee \overline{x}_i \vee \overline{A}, m)\}$, which is the dual-rail decoding of $\mathcal{C}^{dr}$.
- Split step:
  - over n_i: Let $\mathcal{A}^{dr} = \{(A^{dr}, u)\}$, and $\mathcal{C}^{dr} = \{(A^{dr} \vee n_i, u), (A^{dr} \vee \overline{n}_i, u)\}$. Let $\mathcal{A} = \{(A, u)\}$. After split over variable x_i, $\mathcal{C} = \{(A \vee x_i, u), (A \vee \overline{x}_i, u)\}$, which is the dual decoding of $\mathcal{C}^{dr}$.
  - over p_i: Let $\mathcal{A}^{dr} = \{(A^{dr}, u)\}$, and $\mathcal{C}^{dr} = \{(A^{dr} \vee p_i, u), (A^{dr} \vee \overline{p}_i, u)\}$. Let $\mathcal{A} = \{(A, u)\}$. After split over variable x_i, $\mathcal{C} = \{(A \vee x_i, u), (A \vee \overline{x}_i, u)\}$, which is the dual decoding of $\mathcal{C}^{dr}$.

The following lemma shows that tautological clauses do not bring any advantage to the **ResS** proof system,

Lemma 4 *Let $\mathcal{F}$ and $\mathcal{G}$ be MaxSAT formulas without tautological clauses and let $\mathcal{T}$ be a set of tautological clauses. If there is a derivation $\mathcal{F} \cup \mathcal{T} \vdash_{ResS} \mathcal{G}$ of length e, then there is also a derivation $\mathcal{F} \vdash_{ResS} \mathcal{G}$ of length bounded by $p(e)$ where all inference steps are over non-tautological clauses and p is a polynomial.*

Proof. Let $\Pi = (\mathcal{F} \cup \mathcal{T} = \mathcal{F}_0 \cup \mathcal{T}_0; \mathcal{F}_1 \cup \mathcal{T}_1; \ldots; \mathcal{F}_e \cup \mathcal{T}_e)$ be the **ResS** derivation where $\mathcal{G} \subseteq \mathcal{F}_e$. We are going to prove that each formula $\mathcal{F}_i \in \Pi$ can be obtained from $\mathcal{F}_{i-1}$ with **ResS** in a small number of steps without using tautologies, so all tautologies can be removed along the proof.

Let $\mathcal{F}_{i-1} \cup \mathcal{T}_{i-1}; \mathcal{F}_i \cup \mathcal{T}_i$ be any inference step in Π and let $\mathcal{A} \subseteq \mathcal{F}_{i-1} \cup \mathcal{T}_{i-1}$ and $\mathcal{C} \subseteq \mathcal{F}_i \cup \mathcal{T}_i$ be its set of antecedent and consequent clauses. If $\mathcal{A}$ has no tautological clauses (i.e, $\mathcal{A} \subseteq \mathcal{F}_{i-1}$) there is nothing to prove. If $\mathcal{A}$ only has tautological clauses (i.e., $\mathcal{A} \subseteq \mathcal{T}_{i-1}$) then all its consequents are also tautological (i.e., $\mathcal{C} \subseteq \mathcal{T}_i$) no matter the inference rule. Therefore, $\mathcal{F}_i = \mathcal{F}_{i-1}$ without the need of any inference step.

The interesting case takes place when $\mathcal{A}$ has only one tautological clause and the inference is a resolution. Let $\mathcal{A} = \{(A \vee x, u), (B \vee \overline{x}, v)\}$ where, wlog, $(A \vee x, u) \in \mathcal{F}_{i-1}$ and $((B \vee \overline{x}, v) \in \mathcal{T}_{i-1}$. There are the following possibilities:

- Let $B = B' \vee x$. Then, $\mathcal{C} = \{(A \vee B' \vee x, m), (A \vee x, u - m), (A \vee x \vee \overline{B}', m)\} \cup \{(B' \vee x \vee \overline{x}, v - m), (B' \vee x \vee \overline{x} \vee \overline{A}, m), (A \vee x \vee B' \vee \overline{x}, m)\}$ where $m = min\{u, v\}$ and only the first subset of clauses could be non-tautological. That subset of clauses can be obtained as follows. First, unmmerge $\{(A \vee x, u)\}$ into $\{(A \vee x, u - m), (A \vee x, m)\}$. Then, by a series of splits, from $(A \vee x, m)$ we obtain $\{(A \vee B' \vee x, m), (A \vee \overline{B}' \vee x, m)\}$. Note that if $B' = \emptyset$, then $\mathcal{F}_i = \mathcal{F}_{i-1}$ without any inference step.

- Let $B = B' \vee y \vee \overline{y}$. Then, $\mathcal{C} = \{(A \vee x, u - m), (A \vee x \vee y, m), (A \vee x \vee \overline{y}, m)\} \cup \{(A \vee B, m), (B \vee \overline{x}, m), (B \vee \overline{x} \vee \overline{A}, m), (A \vee x \vee y \vee \overline{y} \vee \overline{B}')\}$ where $m = min\{u, v\}$ and only the first subset of clauses could be non-tautological. That subset of clauses can be obtained as follows. First, unmmerge $\{(A \vee x, u)\}$ into $\{(A \vee x, u - m), (A \vee x, m)\}$. Then, by a split, from $(A \vee x, m)$ we obtain $\{(A \vee x \vee y, m), A \vee x \vee \overline{y}, m)\}$

Note that each step $\mathcal{F}_{i-1}; \mathcal{F}_i$ in derivation Π is transformed into at most $1 + n - 1$ steps. Therefore, the length of the new derivation is at most $e \times n$.

Corollary 2 *Let $\mathcal{F}^{dr}$ be the dual-rail encoding of a hard MaxSAT formula $\mathcal{F}$ over variables $\mathcal{X}$. If there is a derivation $\mathcal{F}^{dr} \vdash_{ResS} \mathcal{M}^{dr}$ of size e, then there is a derivation $\mathcal{F} \cup \{(x_i, w_i), (\overline{x}_i, w_i) \mid x_i \in \mathcal{X}\} \vdash_{ResS} \mathcal{M}$ of size m where $\mathcal{M}$ is the dual rail decoding of $\mathcal{M}^{dr}$. Besides, $m \leqslant p(e)$ with p being a polynomial.*

Proof. It follows from Lemma 3 and Lemma 4.

We are now ready to prove the p-simulation.

Theorem 15 *(**ResSV**^{0} p-simulates **Dual Rail with Split**)*
Let $\mathcal{F}$ be a hard MaxSAT formula and $\mathcal{F}^{dr}$ its dual-rail encoding. If there is a refutation $\mathcal{F}^{dr} \vdash_{ResS} (\square, w + 1)$, then there is also a refutation $\mathcal{F} \vdash_{ResSV^0} (\square, 1)$.

Proof. Let $\mathcal{X}$ be the set of variables of $\mathcal{F}$. The first part of the refutation is $\mathcal{F} \vdash_{ResSV^0} \mathcal{F} \cup \{(\square, -w)\} \cup \{(x_i, w_i), (\overline{x}_i, w_i) \mid x_i \in \mathcal{X}\}$ which is obtained as follows. First, we introduce $(\square, w), (\square, -w)$ with the 0-virtual rule. Then, we unmerge $(\square, w)$ to obtain a clause $(\square, w_i)$ for each variable $x_i \in \mathcal{X}$. Finally, we split each $(\square, w_i)$ into $(x_i, w_i), (\overline{x}_i, w_i)$.

The second part of the refutation is $\mathcal{F} \cup \{(\square, -w)\} \cup \{(x_i, w_i), (\overline{x}_i, w_i) \mid x_i \in \mathcal{X}\} \vdash_{ResS} \{(\square, -w) \cup (\square, w + 1)\}$ is obtained using Corollary 2. Note that decoding $(\square, w + 1)$ does not have any effect. Finally, merging the two empty clauses we prove our result.

Theorem 16 *(**Dual Rail with Split** p-simulates **ResSV**^{0}) Let $\mathcal{F}$ be a hard formula and $\mathcal{F}^{dr}$ its dual-rail encoding. If there is a derivation $\mathcal{F} \vdash_{ResSV^0} (\square, 1)$ then there is also a derivation $\mathcal{F}^{dr} \vdash_{ResS} (\square, w + 1)$.*

Proof. Let $\mathcal{X} = \{x_1, \ldots, x_e\}$ be the set of variables of $\mathcal{F}$.

First, by Property 11, we know that the existence of a proof $\mathcal{F} \vdash_{ResSV^0} (\square, 1)$ implies the existence of a derivation,

$$\Pi = (\mathcal{F}_0 = \mathcal{F} \cup \mathcal{B}; \mathcal{F}_1; \ldots; \mathcal{G} \cup \mathcal{B} \cup \{(\square, 1)\} = \mathcal{F}_m)$$

where $\mathcal{B} = \{(\square, b_1), \ldots (\square, b_p)\}$ and the only inference rules needed are split and resolution (along with the usual merge and unmerge).

Second, from $\mathcal{F}^{dr}$ we obtain a formula $\mathcal{F}'$ defined over variables p_i only, as follows. For each variable $x_i \in \mathcal{X}$, we apply the following resolution (which is the result of two MaxSAT resolutions):

$$\frac{\begin{array}{c}(n_i, w_i)\\(\overline{n}_i \vee \overline{p}_i, \infty)\\(p_i, w_i)\end{array}}{\begin{array}{c}(\overline{n}_i \vee \overline{p}_i, \infty)\\(\square, w_i)\\(n_i \vee p_i, w_i)\end{array}}$$

Then, we merge all $(\square, w_i)$ into $(\square, w)$ where $w = \sum_{i=1}^{s} w_i$. Then, we unmerge $(\square, w)$ into $\mathcal{B} \cup \{(\square, w - b)\}$ where $b = \sum_{(\square, b_j) \in \mathcal{B}} b_j$. Finally, let $\mathcal{C}_i = \{(C^{dr}, \infty) \in \mathcal{F}^{dr} \mid \overline{n}_i \in C\}$ be the subset of clauses having literal $\overline{n}_i$. We unmerge $(n_i \vee p_i, w_i)$ into as many clauses $(n_i \vee p_i, w_{ij})$ as $|\mathcal{C}_i|$ and, for each $(A^{dr} \vee \overline{n}_i, \infty) \in \mathcal{C}_i$, we apply resolution:

$$\frac{\begin{array}{c}(A^{dr} \vee \overline{n}_i, \infty)\\(n_i \vee p_i, w_{ij})\end{array}}{\begin{array}{c}(A^{dr} \vee p_i, w_{ij})\\\cdots\end{array}}$$

The result of the previous steps is:

$$\mathcal{F}^{dr} \vdash_{Res} \mathcal{F}' \cup \mathcal{B} \cup \{(\square, w - b)\} \cup \mathcal{G}'$$

where for each clause $(C, \infty) \in \mathcal{F}$ there is a clause $(C', u') \in F'$ in which each literal $x_i \in C$ is replaced by p_i and each literal $\overline{x}_i \in C$ is replaced by $\overline{p}_i$. Moreover, u' is either ∞ or an arbitrarily large number.

Next, we show by induction that for each $\mathcal{F}_j$ in derivation Π there is an $\mathcal{F}'_j$ where literals x_i and $\overline{x}_i$ in $\mathcal{F}_j$ are renamed with p_i and $\overline{p}_i$ in $\mathcal{F}'_j$, finite weights in $\mathcal{F}_j$ are preserved in $\mathcal{F}'_j$, and infinitive weights in $\mathcal{F}_j$ are either preserved or substituted by an arbitrarily large number in $\mathcal{F}'_j$.

Base case (j = 0): we define $F'_0 = \mathcal{F}' \cup \mathcal{B}$, which satisfies the conditions with respect to $\mathcal{F}_0 = \mathcal{F} \cup \mathcal{B}$.

Inductive step: let $\mathcal{A} \subseteq \mathcal{F}_{j-1}$ and $\mathcal{C} \subseteq \mathcal{F}_j$ be the set of antecedent and consequent clauses at step $\mathcal{F}_{j-1}; \mathcal{F}_j$, respectively. Therefore, $\mathcal{A}; \mathcal{C}$ and $\mathcal{F}_j = (\mathcal{F}_{j-1} \backslash \mathcal{A}) \cup \mathcal{C}$. By induction hypothesis, $\mathcal{F}'_{j-1}$ satisfies the conditions wrt $\mathcal{F}_{j-1}$ and, as a consequence, there exists a set $\mathcal{A}' \subseteq \mathcal{F}'_{j-1}$ which also satisfies the conditions over $\mathcal{A}$. We will show that the set of consequents $\mathcal{C}'$ of step $\mathcal{A}'; \mathcal{C}'$ satisfies the conditions over $\mathcal{C}$ and, since $\mathcal{F}'_j = (\mathcal{F}'_{j-1} \backslash \mathcal{A}') \cup \mathcal{C}'$ then $\mathcal{F}'_j$ satisfies the conditions over $\mathcal{F}_j$.

We analyze the inference rule $\mathcal{A}; \mathcal{B}$:

- Split: let $\mathcal{A} = \{(A, u)\}$ and $\mathcal{C} = \{(A \vee x_i, u), (A \vee \overline{x}_i, u)\}$. Then, splitting $\mathcal{A}' = \{(A', u')\}$ we obtain $\mathcal{C}' = \{(A \vee p_i, u'), (A \vee \overline{p}_i, u')\}$. Since by induction hypothesis, $u = u'$ if $u \neq \infty$ and u' is either ∞ or an arbitrarily large number otherwise, $\mathcal{C}'$ satisfies the conditions over $\mathcal{C}$.

- Resolution: let $\mathcal{A} = \{(A \vee x_i, u), (B \vee \overline{x}_i, v)\}$ and $\mathcal{C} = \{(A \vee B, m), (A \vee x_i, u - m), (B \vee \overline{x}_i, v - m), (A \vee x_i \vee \overline{B}, m), (B \vee \overline{x}_i \vee \overline{A}, v - m)\}$ where $m = \min\{u, v\}$. Then, resolving over $\mathcal{A}' = \{(A' \vee p_i, u'), (B' \vee \overline{p}_i, v')\}$ we obtain $\mathcal{C}' = \{(A' \vee B', m'), (A' \vee p_i, u' - m'), (B' \vee \overline{p}_i, v' - m'), (A' \vee p_i \vee \overline{B}', m'), (B' \vee \overline{p}_i \vee \overline{A}', v' - m')\}$ where $m' = m$ if $m \neq \infty$, or m' is ∞ or an arbitrarily large number otherwise. As a consequence, $\mathcal{C}$ satisfies the conditions over $\mathcal{C}$.

The result is that:

$$\mathcal{F}' \cup \mathcal{B} \cup \{(\Box, w-b)\} \vdash_{ResS} \mathcal{G}' \cup \mathcal{B} \cup \{(\Box, 1)\} \cup \{(\Box, w-b)\}$$

Merging empty clauses in $\mathcal{B}$ and $\{(\Box, w-b), (\Box, 1)\}$ we obtain:

$$\mathcal{F}' \cup \mathcal{B} \cup \{(\Box, w-b)\} \vdash_{ResS} \{(\Box, w+1)\}$$

which completes the proof.

7.3 Circular Proofs

Circular proofs [4] from a SAT formula $\mathcal{F}$ are proofs that allow the addition of an arbitrary set of clauses to $\mathcal{F}$. It can be seen that conclusions are sound as long as the added clauses are *re-derived* as many times as they are used. This condition is characterized as the existence of a flow in a graphical representation of the proof. The existence of polynomial size circular proofs for the PHP shows their superiority over general resolution. Here we show that the **ResSV** proof system naturally captures the same idea with an arguably simpler notation. In particular, the virtual rule with its soundness theorem that requires that weights must be positive at the end of the derivation guarantees the existence of the flow.

Given a CNF formula $\mathcal{F}$ and a SAT proof system **S**, a *circular pre-proof* of C_r from $\mathcal{F}$ is a SAT proof

$$\Pi = (C_1, C_2, \dots, C_p, C_{p+1}, C_{p+2}, \dots, C_{p+q}, C_{p+q+1}, C_{p+q+2}, \dots, C_r)$$

such that $\mathcal{F} = \{C_1, C_2, \dots, C_p\}$, $\mathcal{B} = \{C_{p+1}, C_{p+2}, \dots, C_{p+q}\}$ is an arbitrary set of clauses, and each C_i (with $i > p+q$) is obtained from previous clauses by applying an inference rule in **S**. Therefore, a pre-proof is no more than a proof where the original formula $\mathcal{F}$ is augmented with an arbitrary set of new clauses $\mathcal{B}$.

A *circular pre-proof* Π is associated with a (possibly cyclic) directed bi-partite graph $G(\Pi)$. To define such graph, consider first the acyclic graph as defined in Section 4 using $\mathcal{F} \cup \mathcal{B}$ as the start of the proof. $G(\Pi)$ is the compactation of that graph by considering every clause in $C \in \mathcal{B}$ and merging all nodes whose associated clause is identical to it. After the compactation the graph may become cyclic due to the back-edges from derived clauses that were already in $\mathcal{B}$.

A *flow assignment* for a circular pre-proof is an assignment $f : I \longrightarrow \mathbb{N}\backslash\{0\}$ of positive integers where I is the set of inference nodes (see Lemma 1 in [4]). The *balance* of node $C \in J$ is the inflow minus the outflow,

$$b(C) = \sum_{R \in N^-(C)} f(R) - \sum_{R \in N^+(C)} f(R)$$

where $N^-(C)$ and $N^+(C)$ denote the set of in and out-neighbors of node $C \in J$, respectively.

Definition 2 *Given a SAT proof system **S**, a SAT circular proof under **S** of clause A from CNF formula $\mathcal{F}$ is a pre-proof Π whose proof-graph $G(\Pi)$ admits a flow in which all clauses not in $\mathcal{F}$ have non-negative balance and A has a strictly positive balance.*

Property 12 *(see Section 3.5 in [3]) An inference rule satisfies the* multiple consequence property *iff any truth assignment that falsifies one of its consequent formulas satisfies all other consequent formulas.*

Theorem 17 *(see Theorem 4 in [3]) Assuming a sound SAT proof system **S** such that all its inference rules satisfy the multiple consequence property, if there is a SAT circular proof of clause A from $\mathcal{F}$ under SAT proof system **S** then $\mathcal{F} \models A$.*

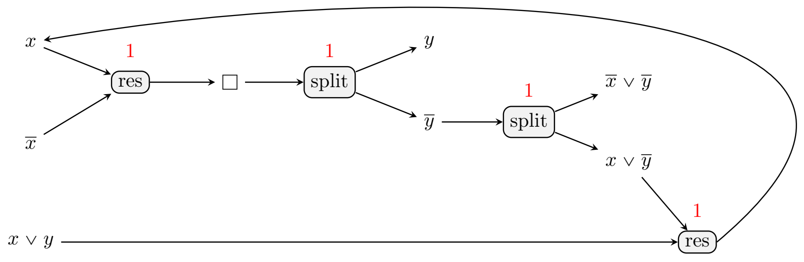


Figure 6: Graph of a circular proof of $\{y\}$ from $\{(x \vee y), (\overline{x})\}$. The certifying flow is indicated above each inference node.

Theorem 18 *(see Theorem 4 in [4]) There is a circular refutation of polynomial length of PHP using the proof system with symmetric resolution and split.*

Figure 6 shows the graph and certifying flow of a circular proof of $\{y\}$ from $\{x \vee y, \overline{x}\}$ with symmetric resolution and split.

Now we show that the MaxSAT **ResSV** proof system is an extension of circular proofs from SAT to MaxSAT. The following two theorems show that, when restricted to hard formulas, **ResSV** and SAT circular can simulate each other.

Theorem 19 *(**ResSV** p-simulates **Circular**) Let Π be a SAT circular proof of clause A from formula $\mathcal{F} = \{C_1, \ldots, C_p\}$ using the proof system symmetric resolution and split. There is a proof $\{(C_1, \infty), \ldots, (C_p, \infty), (\overline{A}, 1)\} \vdash_{ResSV} \{(\square, 1)\}$ whose length is $O(|\Pi|)$.*

Proof. Let $G(\Pi) = (J \cup I, E)$ be the proof graph and $f(\cdot)$ be the flow of Π. By definition of SAT circular proof, $A \in J$ and $b(A) > 0$. Let $o_C = \sum_{R \in N^+(C)} f(R)$ denote the outflow of every clause $C \in J$ and $i_C = \sum_{R \in N^-(C)} f(R)$ denote the inflow of every clause $C \in J$.

First, we show that there is a proof (made exclusively of virtual and unmerge steps),

$$\emptyset; \ldots; \{(C, -o_C) \mid C \in J\} \cup \{(C, f(R)) \mid C \in J, R \in N^+(C)\}$$

This is obtained by considering each clause node $C \in J$, adding $\{(C, -o_C), (C, o_C)\}$ thanks to the virtual rule, and unmerging (C, o_C) as needed. Note that, after these steps, all the antecedents of the inference nodes in I are available.

Second, we show that there is a proof (made exclusively of splits and symmetric resolutions),

$$\{(C, f(R)) \mid C \in J, R \in N^+(C)\} \vdash_{ResS} \{(C, f(R)) \mid C \in J, R \in N^-(C)\}$$

This is obtained by considering each inference node $R \in I$ and transforming its antecedents into its consequences as follows. If R is a SAT split $\{C\} \vdash \{C \vee x, C \vee \overline{x}\}$ then the inference step is a MaxSAT split $\{(C, f(R))\} \vdash \{(C \vee x, f(R)), (C \vee \overline{x}, f(R))\}$. If R is a symmetric SAT resolution $\{C \vee x, C' \vee \overline{x}\} \vdash \{C \vee C'\}$ then the inference step is a (symmetric) MaxSAT resolution $\{(C \vee x, f(R)), (C' \vee \overline{x}, f(R))\} \vdash \{(C \vee C', f(R))\}$.

From the previous two proofs,

$$\mathcal{F} \cup (\overline{A}, 1); \ldots; \mathcal{F} \cup \{(\overline{A}, 1)\} \cup \{(C, -o_C) \mid C \in J\} \cup \{(C, i_C) \mid C \in J\};$$

$$\ldots; \mathcal{F} \cup \{(\overline{A}, 1)\} \cup \{(C, b(C)) \mid C \in J\}$$

Let $A = a_1 \vee a_2 \vee \ldots \vee a_q$. Since, $A \in J$, $b(A) > 0$ and $(\overline{A}, 1)$ is shorthand for $\{(\overline{a}_1, 1), (a_1 \vee \overline{a}_2, 1), \ldots, (a_1 \vee \ldots \vee a_{q-1} \vee \overline{a}_q, 1)\}$, after q MaxSAT resolutions $\{(\overline{A}, 1), (A, 1), (A, b(A) - 1)\} \vdash_{ResSV} \{(\square, 1)\}$ which proves the Theorem.

Lemma 5 *Given a formula $\mathcal{F} = \{(C_1, \infty), \dots, (C_j, \infty), (C_{j+1}, w_{j+1}), \dots, (C_p, w_p)\}$ where $\forall j < k \leqslant p$, $w_k \neq \infty$, if there exists an* ***ResSV*** *refutation $\mathcal{F} \vdash_{ResSV} \{(\Box, 1)\}$ of length l then there exits a* ***ResSV*** *refutation $\mathcal{F}' \vdash_{ResSV} \{(\Box, 1)\}$ of length $O(l)$ where*

$$\mathcal{F}' = \{(C_1, w_1), \dots, (C_j, w_j), (C_{j+1}, w_{j+1}), \dots, (C_p, w_p)\}$$

and $\forall 1 \leqslant k \leqslant j$, $w_k \neq \infty$.

Proof. For readability reasons, $\vdash$ denotes $\vdash_{ResSV}$. By Property 9, any $\mathcal{F} \vdash \{(\Box, 1)\}$ of length l can be rewritten into an equivalent refutation of length $e = O(l)$ in which resolution is restricted to its symmetric form. Let $\Pi = (\mathcal{F}_0; \mathcal{F}_1; \dots; \mathcal{F}_e)$ be that refutation where $\mathcal{F}_0 = \mathcal{F}$ and $(\Box, 1) \in \mathcal{F}_e$.

We are going to prove that for each $\mathcal{F}_i = \{(C_1, \infty), \dots, (C_j, \infty), (C_{j+1}, w_{j+1}), \dots, (C_p, w_p)\}$ there is an $\mathcal{F}'_i = \{(C_1, w_1), \dots, (C_j, w_j), (C_{j+1}, w_{j+1}), \dots, (C_p, w_p)\}$ such that $\forall 1 \leqslant k \leqslant j$ $w_k \neq \infty$ and $\mathcal{F}'_i \vdash \{(\Box, 1)\}$. We prove it by induction on i going in reverse order, from $i = e$ to $i = 0$.

Base case ($i = e$): we define,

$$\mathcal{F}'_e = \{(C_1, 1), \dots, (C_j, 1), (C_{j+1}, w_{j+1}), \dots, (C_p, w_p)\}$$

Since $(\Box, 1) \in \mathcal{F}_e$, then $(\Box, 1) \in \mathcal{F}'_e$. Thus, it satisfies the conditions.

Inductive step: Let $\mathcal{A} \subseteq \mathcal{F}_{i-1}$ and $\mathcal{C} \subseteq \mathcal{F}_i$ be the set of antecedent and consequent clauses in step $\mathcal{F}_{i-1}; \mathcal{F}_i$. Therefore, $\mathcal{A}; \mathcal{C}$ and $\mathcal{F}_{i-1} = (\mathcal{F}_i \backslash \mathcal{C}) \cup \mathcal{A}$. By induction hypothesis, there is a $\mathcal{C}' = \{(C, w) \mid (C, u) \in \mathcal{C}\} \subseteq \mathcal{F}'_i$ such that if $u \neq \infty$ then $w = u$, else w is finite. We define $\mathcal{F}'_{i-1}$ as $(\mathcal{F}'_i \backslash \mathcal{C}') \cup \mathcal{A}'$ for some $\mathcal{A}'$ satisfying:

1. $\mathcal{A}' = \{(C, w) \mid (C, u) \in \mathcal{A}\}$ such that if $u \neq \infty$ then $w = u$, else w is finite, and
2. there is a proof $\mathcal{A}'; \dots; \mathcal{C}'$

As a result, $\mathcal{F}'_{i-1}$ has the same clauses as $\mathcal{F}_{i-1}$ but with finite weight. Moreover, since there is a proof $\mathcal{A}'; \dots; \mathcal{C}'$ where $\mathcal{A}' \in \mathcal{F}'_{i-1}$ and $\mathcal{C}' \in \mathcal{F}'_i$, then there is a proof $\mathcal{F}'_{i-1}; \dots; \mathcal{F}'_i$ and, since by induction $\mathcal{F}'_i \vdash \{(\Box, 1)\}$, then $\mathcal{F}'_{i-1} \vdash \{(\Box, 1)\}$.

Next, we show how to obtain such $\mathcal{A}'$ for the different cases. If all clauses in $\mathcal{A}$ have finite weight then all clauses in $\mathcal{C}$ have also finite weight. As a consequence, $\mathcal{C}' = \mathcal{C}$. Then, $\mathcal{A}' = \mathcal{A}$ trivially satisfies the conditions. If some clause in $\mathcal{A}$ has infinite weight then we analyze each possible inference rule that can happen in the $\mathcal{F}_{i-1}; \mathcal{F}_i$ step:

- *Split*: By definition of the split rule, $\mathcal{A} = \{(C, \infty)\}$, $\mathcal{C} = \{(C \vee x, \infty), (C \vee \overline{x}, \infty)\})$. Besides, $\mathcal{C}' = \{(C \vee x, u), (C \vee \overline{x}, v)\}$. Then, $\mathcal{A}' = \{(C, \max\{u, v\})\}$ satisfies the conditions.
- *Symmetric resolution*: By definition of the symmetric resolution rule, $\mathcal{A} = \{(C \vee x, \infty), (C \vee \overline{x}, \infty)\}$, $\mathcal{C} = \{(C, \infty), (C \vee x, \infty), (C \vee \overline{x}, \infty)\}$. Besides, $\mathcal{C}' = \{(C, u), (C \vee x, v), (C \vee \overline{x}, v')\}$. Then, $\mathcal{A}' = \{(C \vee x, u + v), (C \vee \overline{x}, u + v')\}$ satisfies the conditions.
- *Merge* with both antecedents having infinite weight: By definition of merge rule, $\mathcal{A} = \{(C, \infty), (C, \infty)\}$, $\mathcal{C} = \{(C, \infty)\}$. Besides, $\mathcal{C}' = \{(C, v)\}$. Then, $\mathcal{A}' = \{(C, v), (C, v)\}$ satisfies the conditions.
- *Merge* with one of the antecedents having finite weight: By definition of merge rule, $\mathcal{A} = \{(C, \infty), (C, u)\}$, $\mathcal{C} = \{(C, \infty)\}$. Besides, $\mathcal{C}' = \{(C, v)\}$:
  - if $0 < v \leqslant u$, then $\mathcal{A}' = \{(C, 1), (C, u)\}$ satisfies the conditions.

  - otherwise, $\mathcal{A}' = \{(C, v-u), (C, u)\}$. Note that u could be a negative weight coming from a virtual rule. In any case, $v - u > 0$ and $\mathcal{A}'$ satisfies the conditions.

- *Unmerge*: By definition of unmerge rule, $\mathcal{A} = \{(C, \infty)\}$, $\mathcal{C} = \{(C, \infty), (C, \infty)\}$. Besides, $\mathcal{C}' = \{(C, u), (C, v)\}$. Then, $\mathcal{A}' = \{(C, u+v)\}$ satisfies the conditions.

Theorem 20 *(**Circular** p-simulates **ResSV**) Consider a hard formula $\mathcal{H} = \{(C_1, \infty), \ldots, (C_p, \infty)\}$ and a MaxSAT proof $\mathcal{H} \cup \{(\overline{A}, 1)\} \vdash_{ResSV} \{(\square, 1)\}$ of length e. There is a SAT circular proof Π of A from $\mathcal{H}' = \{C_1, \ldots, C_p\}$ with proof system having symmetric resolution and split. The length of the circular proof is $O(e)$.*

Proof. From derivation $\mathcal{H} \cup \{(\overline{A}, 1)\} \vdash_{ResSV} \{(\square, 1)\}$ we need to build a pre-proof Π with a (possibly cyclic) graph $G(\Pi) = (J \cup I, E)$ and a flow $f(\cdot)$ that certifies that the pre-proof is indeed a circular proof. The graph must satisfy that $\mathcal{H}' \subset J$, $A \in J$; its inference nodes must be consistent with either symmetric resolution or split. Also, the flow $f(\cdot)$ must satisfy the balance conditions including that A has strictly positive balance.

First, by Lemma 5, there exists an $\mathcal{F} = \{(C_1, w_1), \ldots, (C_p, w_p)\}$ with $w_k \neq \infty$ for all $1 \leqslant k \leqslant p$, such that $\mathcal{F} \cup \{(\overline{A}, 1)\} \vdash_{ResSV} \{(\square, 1)\}$ with length $O(e)$ where resolution is restricted to its symmetric form. Moreover, by Property 11 and Property 9, $\mathcal{F} \cup \{(\overline{A}, 1)\} \vdash_{ResSV} \{(\square, 1)\}$ implies the existence of a derivation Γ,

$$\mathcal{F}_0 = (\mathcal{F} \cup \{(\overline{A}, 1)\} \cup \mathcal{B}); \mathcal{F}_1; \mathcal{F}_2; \ldots; (\mathcal{G} \cup \{(\square, 1)\} \cup \mathcal{B}); ...; \mathcal{F}_{m-1}; (\mathcal{G} \cup \{(A, 1), (\overline{A}, 1)\} \cup \mathcal{B}) = \mathcal{F}_m$$

where $\mathcal{B}$ is the set of clauses with positive weight added by the virtual rule in the original **ResSV** derivation, $m = O(e)$, and the only inference rules needed are split and symmetric resolution (along with the usual merge and unmerge).

First, we build the (acyclic) graph $G(\Gamma)$ along with a flow function $f(\cdot)$. Let $G_i(\Gamma) = (J_i \cup I_i, E_i)$ be the graph at step i, $b_i(C)$ be the balance of node $C \in J_i$ in $G_i(\Gamma)$, and let $merged(\mathcal{F}_i)$ be equivalent to $\mathcal{F}_i$ with no repeated clauses.

We will traverse the derivation Γ from $\mathcal{F}_0$ to $\mathcal{F}_m$ ensuring that, at each step i, $G_i(\Gamma)$ satisfies:

1. $\forall (C, w) \in \mathcal{F}_i$, $C \in J_i$
2. $\forall (C, w) \in merged(\mathcal{F}_i)$, $w = b_i(C)$
3. all nodes in J_i are different

We proceed by induction on the step i.

Base case ($i = 0$). Then:

- $J_0 = \{C \mid (C, w) \in merged(\mathcal{F}_0)\}$
- $I_0 = \{d_C \mid C \in J_0\}$ (dummy inference nodes)
- $E_0 = \{(d_C, C) \mid C \in J_0, d_C \in I_0\}$
- $\forall (C, w) \in merged(\mathcal{F}_0), f(d_C) = w$

Inductive step: Let $\mathcal{A} \subseteq \mathcal{F}_i$ and $\mathcal{C} \subseteq \mathcal{F}_{i+1}$ be the antecedents and consequents of $\mathcal{F}_i; \mathcal{F}_{i+1}$ respectively. Note that, by induction hypothesis for every clause $(C, w) \in \mathcal{A}$ there is a node $C \in J_i$. The construction of $G_{i+1}(\Gamma)$ depends on the inference rule used:

- Split/Symmetric resolution:

- $J_{i+1} = J_i \cup \{C \mid (C,w) \in \mathcal{C} \wedge C \notin J_i\}$
- $I_{i+1} = I_i \cup \{i+1\}$
- $E_{i+1} = E_i \cup \{(C, i+1) \mid (C,w) \in \mathcal{A}\} \cup \{(i+1, C) \mid (C,w) \in \mathcal{C}\}$
- $f(i+1) = w$, where w is the common weight of all clauses in $\mathcal{A}$

As a result, $\forall (C,w) \in \mathcal{A}, b_{i+1}(C) = b_i(C) - w$ and $\forall (C,w) \in \mathcal{C}, b_{i+1}(C) = b_i(C) + w$. Since $\forall (C,w) \in \mathcal{A}$, its weight in $merged(\mathcal{F}_{i+1})$ is decreased by w wrt its weight in $merged(\mathcal{F}_i)$, and $\forall (C,w) \in \mathcal{C}$, its weight in $merged(\mathcal{F}_{i+1})$ is increased by w wrt its weight in $merged(\mathcal{F}_i)$, we can guarantee that $G_{i+1}(\Gamma)$ satisfies (1), (2) and (3).

- Merge/Unmerge: since $merged(\mathcal{F}_{i+1}) = merged(\mathcal{F}_i)$, we define $G_{i+1}(\Gamma)$ as $G_i(\Gamma)$ which, by induction hypothesis, satisfies (1), (2) and (3).

The result is that there is a node in $G_m(\Gamma)$ for all clauses in $merged(\mathcal{F}_m)$ and the weight of each of them corresponds to its balance. In particular, $\mathcal{H}' \in J_m$; $A \in J_m$ and $b_m(A) \geqslant 1$; $\forall (C,w) \in \mathcal{B}$, $b_m(C) \geqslant w$; and $\forall C \in \overline{A}$, $C \in J_m$ and $b_m(C) \geqslant 1$.

Let $G(\Pi) = (J_p \cup (I_p \backslash I_0), E_p \backslash E_0)$. Since $\forall (C,w) \in merged(\mathcal{F}_0)$, $d_C \in I_0$ and $f(d_C) = w$:

- $\forall (C,w) \in \mathcal{B}$, $b(C) = b_m(C) - w \geqslant 0$;
- $\forall C \in \mathcal{H}'$, $b(C)$ may become negative (but they are the hard clauses);
- $\forall C \in \overline{A}$, $b(C) = b_m(C) - 1 \geqslant 0$.

Moreover, balance $b(A)$ remains positive.

7.4 Refuting SAT by reducing it to Max2SAT

Ansótegui and Levy [1] propose a general method to refute a SAT formula by reducing SAT to Max2SAT. The reduction uses gadgets that transform a hard clause C into a (poly-size) set $\mathcal{G}$ of soft clauses that may include new variables. The arity of a gadget is the maximum size of its clauses. Clearly, reducing SAT to Max2SAT requires binary (i.e, arity 2) gadgets. Like any set of soft clauses, $\mathcal{G}$ has an associated cost function $\mathcal{G}(X,B)$ with X being an assignment to the variables in C and B being an assignment to the new variables. Gadgets satisfy that there is some $0 \leqslant \gamma$ such that:

- for the only truth assignment over X that violates $\mathcal{C}$, the best (i.e, minimum cost) extension over B is $\mathcal{G}(X,B) > \gamma$
- for every truth assignment that satisfies $\mathcal{C}$, the best (i.e, minimum cost) extension over B is $\mathcal{G}(X,B) = \gamma$

Example 3 *Some gadgets in the literature are:*

Garey's gadget *[15] for $x_1 \vee x_2 \vee x_3$ is the set of clauses $\{(x_i,1),(b,1),(\overline{x}_i \vee \overline{x}_j,1),(\overline{b} \vee x_i,1)\}$ with $1 \leqslant i < j \leqslant 3$. It is easy to see by inspection that this gadget has $\gamma = 3$.*

Garey's refined gadget *(Lemma 3 in [1]) for $x_1 \vee x_2 \vee x_3$ is the set of clauses $\{(x_1 \vee x_2,1),(x_1 \vee x_3,1),(\overline{x}_2 \vee \overline{x}_3,1),(b \vee \overline{x}_1,1),(\overline{b} \vee x_2,1),(\overline{b} \vee x_3,1)\}$. This gadget has $\gamma = 1$.*

Regular gadget *(Theorem 1 in [1]) for $x_1 \vee x_2 \vee \ldots \vee x_k$ is the following set of clauses, after replacing b_{k-1} by x_k,*

$$\begin{array}{rl} (x_i,1) & i = 1 \ldots k \\ (\overline{x}_i \vee \overline{b}_i,1) & i = 1 \ldots k-1 \\ (\overline{x}_{i+1} \vee b_i,1) & i = 1 \ldots k-2 \\ (b_i \vee \overline{b}_{i+1},1) & i = 1 \ldots k-2 \end{array}$$

This gadget has $\gamma = k - 1$.

The Max2SAT encoding of the SAT formula $\mathcal{F}$ is the weighted formula $\mathcal{G}_F$ obtained by applying the gadget to every clause in $\mathcal{F}$ of arity higher than 2. It is easy to see that $\mathcal{F}$ is unsatisfiable iff $MaxSAT(\mathcal{G}_F) > m\gamma$, where m is the number of times the gadget is applied. Therefore, a *gadget-based refutation* of a SAT formula $\mathcal{F}$ is a derivation $\mathcal{G}_{\mathcal{F}} \vdash_{Res} (\square, m\gamma + 1)$. The authors show that using the Regular gadget there is a polynomial refutation of PHP.

The paper reports some preliminary empirical results in which different gadgets and different MaxSAT solvers are tested over PHP. The experiment shows that: *i*) reducing PHP to $Max2SAT$ is effective, *ii*) the MaxSAT solver of choice is not very relevant, and *iii*) the gadget of choice is extremely relevant. Regarding this experiment we make two claims:

1. The efficiency of using gadgets in the PHP depends mostly on whether gadgets are covering or not. We say that a gadget is *covering* if for all $l \in C$ we have that $(l, 1) \in \mathcal{G}$. Note that Garey's and Regular gadgets are covering, while Garey's refined gadget is not.

2. The efficiency of using gadgets in the PHP does not require gadgets to have arity 2.

To test our hypothesis we define two covering non-binary gadgets. The first one is a **dummy gadget** that just adds clauses transforming $x_1 \vee x_2 \vee \ldots \vee x_k$ into

$$\{(x_1 \vee x_2 \vee \ldots \vee x_k, 1), (x_1, 1), (x_2, 1), \ldots, (x_k, 1),\\ (\overline{x}_1 \vee x_2, 1), (\overline{x}_1 \vee \overline{x}_2, 1), (\overline{x}_2 \vee x_3, 1), (\overline{x}_2 \vee \overline{x}_3, 1), \ldots, (\overline{x}_k \vee x_1, 1), (\overline{x}_k \vee \overline{x}_1, 1)\}$$

The second gadget is a ternary version of the regular gadget, called **ternary regular**, obtained by applying MaxSAT resolution to each pair

$$(\overline{x}_i \vee \overline{b}_i, 1), (\overline{x}_{i+1} \vee b_i, 1)$$

which produces, after replacing b_{k-1} by x_k,

$$\begin{array}{rl} (x_i, 1) & i = 1 \ldots k \\ (\overline{x}_i \vee x_{i+1} \vee \overline{b}_i, 1) & i = 1 \ldots k-1 \\ (x_i \vee \overline{x}_{i+1} \vee b_i, 1) & i = 1 \ldots k-2 \\ (\overline{x}_i \vee \overline{x}_{i+1}, 1) & i = 1 \ldots k-1 \\ (b_i \vee \overline{b}_{i+1}, 1) & i = 1 \ldots k-2 \end{array}$$

Figure 7.4 reports the solving time (note the log scale) of the MaxSAT encoding of the PHP with different gadgets using algorithm RC2 [19]. We can clearly identify three groups of gadgets with nearly identical performance. In the first group we have the three non-covering gadgets: Trevisan [26], Garey's refined and refined regular (Theorem 2 in [1]). In this set there is a quick exponential growth. In the second group we have gadgets for which we have first transformed the k-arity clauses into ternary using the usual trick (see Lemma 5 in [1]). All gadgets in this set are covering and they are much more efficient than the first non-covering group. Interestingly, our dummy gadget also falls into this group although it is non-binary. These results support our claim on the importance of the unit clauses and the irrelevance of all clauses being binary. The third group contains the most efficient gadgets. Besides being covering, they have in common that they apply directly to k-ary clauses. In this group we have the original binary regular gadget, our ternary version of regular, and our dummy gadget. Their performance is nearly identical, so we believe that the reason is that they do not need the kSAT to 3SAT intermediate step. These results further support our claim on the importance of the unit clauses and the irrelevance of all clauses being binary.

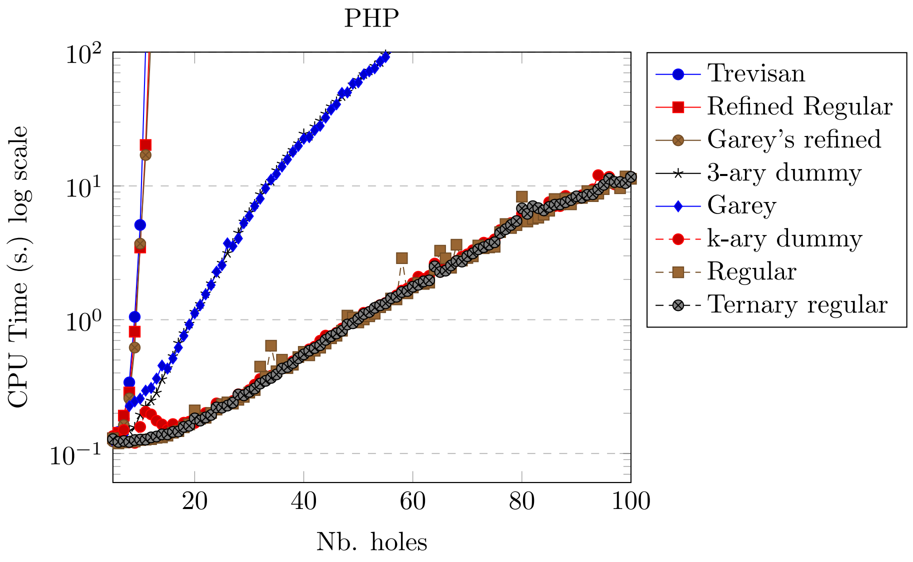


Figure 7: Solving time (in seconds) of PHP as a function of the number of holes with different gadgets. Note the logarithmic scale.

8. Related Work

In this Section we review and discuss two related works that have influenced the research presented in this paper.

8.1 Soft Probing

ResSV contains three rules that provide increasing refutational power. While increasing the power is a desirable feature, having more rules to choose from makes the automatization more difficult. Therefore, one practical challenge is to use split and virtual in a controlled but potentially useful way. Soft Probing is a technique that was used as a pre-process in the MiniMaxSAT solver [17] to extract an initial lower bound from MaxSAT formulas. It can be seen as a simple, yet efficient implementation of this idea. In the original paper, the technique is presented algorithmically and very briefly. Next, we show how it fits into the context of this paper.

Consider the following theorem,

Theorem 21 *Let $\mathcal{F}$ be a weighted MaxSAT formula. If there is a unary (i.e, made exclusively of unit clauses) formula $\mathcal{U}$ such that if $(l, w) \in \mathcal{U}$ and $k = \sum_{(l,w)\in\mathcal{U}} w$ then $(\bar{l}, u) \notin \mathcal{U}$, and,*

1. $\mathcal{F} \cup \mathcal{U} \vdash_{Res} \mathcal{G} \cup \{(\Box, k)\}$
2. $\mathcal{G} \cup \overline{\mathcal{U}} \vdash_{Res} \{(\Box, k')\}$

Then, $\mathcal{F} \vdash_{ResSV} \{(\Box, k')\}$.

Proof. From $\mathcal{F}$ we apply the virtual rule with every unit clause in $\mathcal{U}$ obtaining $\mathcal{F} \cup \mathcal{U} \cup \mathcal{U}^-$ with $\mathcal{U}^- = \{(l, -w) \mid (l, w) \in \mathcal{U}\}$. Then, we use the first proof in the theorem obtaining

$$\mathcal{G} \cup \mathcal{U}^- \cup \{(\Box, k)\}$$

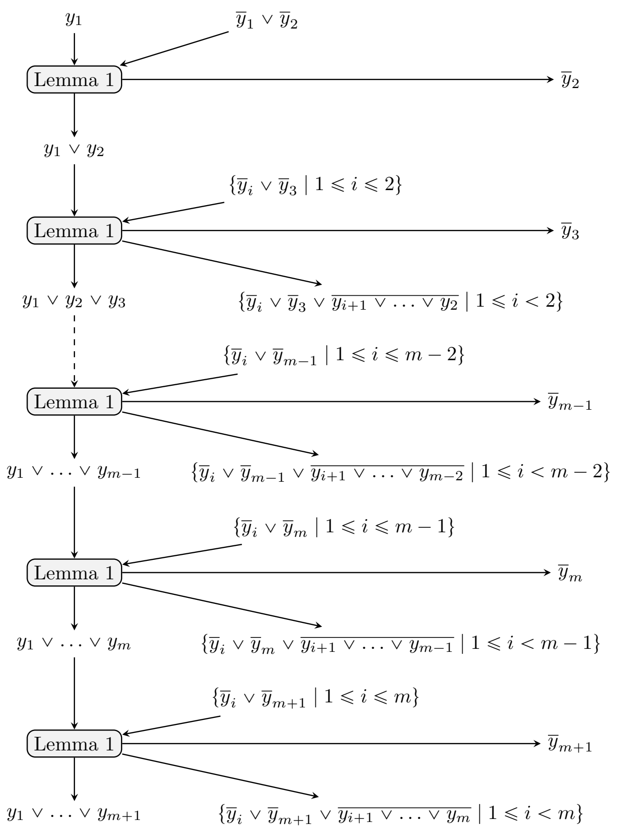

y_1
$\overline{y}_1 \vee \overline{y}_2$
Lemma 1
$\overline{y}_2$
$y_1 \vee y_2$
$\{\overline{y}_i \vee \overline{y}_3 \mid 1 \leqslant i \leqslant 2\}$
Lemma 1
$\overline{y}_3$
$y_1 \vee y_2 \vee y_3$
$\{\overline{y}_i \vee \overline{y}_3 \vee \overline{y_{i+1} \vee \ldots \vee y_2} \mid 1 \leqslant i < 2\}$
$\{\overline{y}_i \vee \overline{y}_{m-1} \mid 1 \leqslant i \leqslant m-2\}$
Lemma 1
$\overline{y}_{m-1}$
$y_1 \vee \ldots \vee y_{m-1}$
$\{\overline{y}_i \vee \overline{y}_{m-1} \vee \overline{y_{i+1} \vee \ldots \vee y_{m-2}} \mid 1 \leqslant i < m-2\}$
$\{\overline{y}_i \vee \overline{y}_m \mid 1 \leqslant i \leqslant m-1\}$
Lemma 1
$\overline{y}_m$
$y_1 \vee \ldots \vee y_m$
$\{\overline{y}_i \vee \overline{y}_m \vee \overline{y_{i+1} \vee \ldots \vee y_{m-1}} \mid 1 \leqslant i < m-1\}$
$\{\overline{y}_i \vee \overline{y}_{m+1} \mid 1 \leqslant i \leqslant m\}$
Lemma 1
$\overline{y}_{m+1}$
$y_1 \vee \ldots \vee y_{m+1}$
$\{\overline{y}_i \vee \overline{y}_{m+1} \vee \overline{y_{i+1} \vee \ldots \vee y_m} \mid 1 \leqslant i < m\}$

Figure 8: Derivation graph corresponding to hole j. For clarity purposes, we rename each variable x_{ij}, $1 \leqslant i \leqslant m+1$ to y_i. All clauses have cost 1.

Using the split rule, we transform $(\Box, k)$ into $\mathcal{U} \cup \overline{\mathcal{U}}$ obtaining

$$\mathcal{G} \cup \mathcal{U}^- \cup \mathcal{U} \cup \overline{\mathcal{U}}$$

Then we eliminate $\mathcal{U}^- \cup \mathcal{U}$ which cancel each other and use the second proof of the theorem to obtain

$$\{(\Box, k')\}$$

This Theorem gives a **Res** condition to identify a **ResSV** derivation that produces an increment in the lower bound. Soft Probing applies this theorem iteratively for every literal l in the formula. At each step, $\mathcal{U}$ is restricted to $\{(l, w)\}$ and it only considers unit propagation (which can be implemented efficiently) for the two derivations.

Now, a natural question arises: how powerful is **ResSV** when restricted to the use of this Theorem? Interestingly enough, it is sufficient for refuting the PHP and $SPHP$ in polynomial time. The following property shows that both problems satisfy the conditions of the previous theorem.

Property 13 *Consider the PHP and $SPHP$ problems and let $\mathcal{U} = \{(x_{11}, 1), (x_{12}, 1), \ldots, (x_{1m}, 1)\}$.*

- *There is a proof $PHP \cup \mathcal{U} \vdash_{Res} PHP \cup \{(\Box, m)\}$*

- *There is a proof* $PHP \cup \overline{\mathcal{U}} \vdash_{Res} \{(\Box, 1)\}$
- *There is a proof* $SPHP \cup \mathcal{U} \vdash_{Res} \mathcal{G} \cup \{(\Box, m)\}$
- *There is a proof* $\mathcal{G} \cup \overline{\mathcal{U}} \vdash_{Res} \{(\Box, 1)\}$

Proof. First, we prove the $SPHP$ case. The first refutation of $SPHP$ is as follows. First, for each hole j and $\{(x_{1j}, 1)\}$ there is a derivation of $\{(\overline{x}_{ij}, 1) \mid 2 \leqslant i \leqslant m+1\}$ (see Figure 8). Then, for each pigeon $i > 1$ and $\{(\overline{x}_{ij}, 1) \mid 1 \leqslant j \leqslant m\}$, there is a derivation of $\{(\Box, 1)\}$ (see Figure 5 (left)). Therefore, concatenating the previous derivations we get,

$$SPHP \cup \mathcal{U} \vdash_{Res} \mathcal{G} \cup \{(\Box, m)\}$$

where clause $\{(x_{11} \vee x_{12} \vee \ldots \vee x_{1m}, 1)\} \in \mathcal{G}$. Figure 5 (left) shows the derivation graph of the second refutation,

$$\{(x_{11} \vee x_{12} \vee \ldots \vee x_{1m}, 1)\} \cup \overline{\mathcal{U}} \vdash_{Res} \{(\Box, 1)\}$$

which completes the proof.

Let us now prove the PHP case. Since $PHP \vdash_{Res} PHP \cup SPHP$ by unmerging each hard clause $(C, \infty) \in PHP$ into $(C, \infty), (C, 1)$ and we have proved that $SPHP \cup \mathcal{U} \vdash_{Res} \mathcal{G} \cup \{(\Box, m)\}$, then $PHP \cup \mathcal{U} \vdash_{Res} PHP \cup \{(\Box, m)\}$. Since we have proved that $\{(x_{11} \vee x_{12} \vee \ldots \vee x_{1m}, 1)\} \cup \overline{\mathcal{U}} \vdash_{Res} \{(\Box, 1)\}$, unmerging $(x_{11} \vee x_{12} \vee \ldots \vee x_{1m}, \infty) \in PHP$ into $(x_{11} \vee x_{12} \vee \ldots \vee x_{1m}, \infty), (x_{11} \vee x_{12} \vee \ldots \vee x_{1m}, 1)$ completes the proof.

8.2 OSAC

Weighted Constraint Satisfaction Problems (WCSPs) are optimization problems defined by a network of local cost functions defined over discrete variables. Thus, MaxSAT can be seen as a particular type of WCSP where the local cost functions are the clauses and variables are boolean[12]. WCSP solvers compute lower bounds by enforcing *local consistency*. This is achieved by moving costs around the network using two equivalence preserving operations: *projection* and *extension*. WCSP projection is similar to MaxSAT symmetric resolution and WCSP extension is similar to split. The main difference is that in the WCSPs movements are restricted to pre-defined subsets of variables (i.e, the scopes of the original cost functions), while in **ResSV** the proof system gives complete freedom on the variables involved in the clauses. This freedom is needed to guarantee completeness, which is not a problem in the WCSP context where local consistency is not used as a stand-alone algorithm, but only as a heuristic.

Optimal Soft Arc Consistency OSAC [11] introduced the idea of allowing weights to become negative during the process. As in our case, it is shown that the lower bound is valid (i.e, sound) as long as all the weights are positive at the end of the process. Interestingly, OSAC can be enforced with a linear program. Solving the linear program produces the optimal lower bound is obtained (optimal with respect to the pre-defined scopes on which costs can be moved to).

Thus, OSAC is reminiscent to a **ResSV** proof restricting new clauses to pre-defined (and of bounded size) sets of variables. Interestingly, the efficiency of **ResSV** on the SPHP problem does not rely on the size of the clauses which is as high as the number of pigeons and holes, and therefore unbounded.

9. Conclusions and Future Work

Several approaches for MaxSAT solving have been proposed in the last years and most of the comparisons have been done empirically. In this paper we set some basic definitions for a proof complexity approach, which we believe may be a very useful complement. From a descriptive point of view, our theoretical approach provides a framework to explain under a common language some related work such as circular proofs (Section 7.3), soft probing (Section 8.1) or dual rail (Section 7.2). Because proof systems break inferences into different rules, a proof complexity approach facilitates the understanding of the advantages and limitations of each different rule (the very recent work of [14] already gives some support to this claim). Our paper covers a first analysis of three inference rules: resolution, split and virtual, with split and virtual being original from our work. We show that the addition of each rule makes the proof system stronger.

We expect this work to motivate other MaxSAT practitioners to use our framework to analyze their contributions. In our future work we want to explore the relationship between certifying lower bounds with search algorithms and proof systems. This idea, which has shed so much light to the SAT case would be very beneficial also for MaxSAT.

Acknowledgements

We are thankful to Carlos Ansótegui and the anonymous referees for many helpful comments.

References

[1] Carlos Ansótegui and Jordi Levy. Reducing SAT to max2sat. In Zhi-Hua Zhou, editor, *Proceedings of the Thirtieth International Joint Conference on Artificial Intelligence, IJCAI 2021, Virtual Event / Montreal, Canada, 19-27 August 2021*, pages 1367–1373. ijcai.org, 2021.

[2] Albert Atserias. On sufficient conditions for unsatisfiability of random formulas. *J. ACM*, 51(2):281–311, 2004.

[3] Albert Atserias and Massimo Lauria. Circular (yet sound) proofs. *CoRR*, abs/1802.05266, 2018.

[4] Albert Atserias and Massimo Lauria. Circular (yet sound) proofs. In Mikolás Janota and Inês Lynce, editors, *Theory and Applications of Satisfiability Testing - SAT 2019 - 22nd International Conference, SAT 2019, Lisbon, Portugal, July 9-12, 2019, Proceedings*, volume 11628 of *Lecture Notes in Computer Science*, pages 1–18. Springer, 2019.

[5] 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.

[6] Maria Luisa Bonet, Sam Buss, Alexey Ignatiev, João Marques-Silva, and António Morgado. Maxsat resolution with the dual rail encoding. In Sheila A. McIlraith and Kilian Q. Weinberger, editors, *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence, (AAAI-18), the 30th innovative Applications of Artificial Intelligence (IAAI-18), and the 8th AAAI Symposium on Educational Advances in Artificial Intelligence (EAAI-18), New Orleans, Louisiana, USA, February 2-7, 2018*, pages 6565–6572. AAAI Press, 2018.

[7] Maria Luisa Bonet, Sam Buss, Alexey Ignatiev, António Morgado, and João Marques-Silva. Propositional proof systems based on maximum satisfiability. *Artif. Intell.*, 300:103552, 2021.

[8] Maria Luisa Bonet and Jordi Levy. Equivalence between systems stronger than resolution. In Luca Pulina and Martina Seidl, editors, *Theory and Applications of Satisfiability Testing - SAT 2020 - 23rd International Conference, Alghero, Italy, July 3-10, 2020, Proceedings*, volume 12178 of *Lecture Notes in Computer Science*, pages 166–181. Springer, 2020.

[9] Maria Luisa Bonet, Jordi Levy, and Felip Manyà. Resolution for max-sat. *Artif. Intell.*, 171(8-9):606–618, 2007.

[10] Joshua Buresh-Oppenheim and Toniann Pitassi. The complexity of resolution refinements. *J. Symb. Log.*, 72(4):1336–1352, 2007.

[11] Martin C. Cooper, Simon de Givry, Martí Sánchez-Fibla, Thomas Schiex, Matthias Zytnicki, and T. Werner. Soft arc consistency revisited. *Artif. Intell.*, 174(7-8):449–478, 2010.

[12] Simon de Givry, Javier Larrosa, Pedro Meseguer, and Thomas Schiex. Solving max-sat as weighted CSP. In Francesca Rossi, editor, *Principles and Practice of Constraint Programming - CP 2003, 9th International Conference, CP 2003, Kinsale, Ireland, September 29 - October 3, 2003, Proceedings*, volume 2833 of *Lecture Notes in Computer Science*, pages 363–376. Springer, 2003.

[13] Jan Elffers and Jakob Nordström. Divide and conquer: Towards faster pseudo-boolean solving. In *Proceedings of the Twenty-Seventh International Joint Conference on Artificial Intelligence, IJCAI-18*, pages 1291–1299. International Joint Conferences on Artificial Intelligence Organization, 7 2018.

[14] Yuval Filmus, Meena Mahajan, Gaurav Sood, and Marc Vinyals. Maxsat resolution and subcube sums. *Electronic Colloquium on Computational Complexity (ECCC)*, 27:82, 2020.

[15] M. R. Garey, David S. Johnson, and Larry J. Stockmeyer. Some simplified np-complete graph problems. *Theor. Comput. Sci.*, 1(3):237–267, 1976.

[16] Armin Haken. The intractability of resolution. *Theoretical Computer Science*, 39:297 – 308, 1985. Third Conference on Foundations of Software Technology and Theoretical Computer Science.

[17] Federico Heras, Javier Larrosa, and Albert Oliveras. Minimaxsat: An efficient weighted max-sat solver. *J. Artif. Intell. Res.*, 31:1–32, 2008.

[18] Alexey Ignatiev, António Morgado, and João Marques-Silva. On tackling the limits of resolution in SAT solving. In Serge Gaspers and Toby Walsh, editors, *Theory and Applications of Satisfiability Testing - SAT 2017 - 20th International Conference, Melbourne, VIC, Australia, August 28 - September 1, 2017, Proceedings*, volume 10491 of *Lecture Notes in Computer Science*, pages 164–183. Springer, 2017.

[19] Alexey Ignatiev, António Morgado, and João Marques-Silva. RC2: an efficient maxsat solver. *J. Satisf. Boolean Model. Comput.*, 11(1):53–64, 2019.

[20] Javier Larrosa and Federico Heras. Resolution in max-sat and its relation to local consistency in weighted csps. In Leslie Pack Kaelbling and Alessandro Saffiotti, editors, *IJCAI-05, Proceedings of the Nineteenth International Joint Conference on Artificial Intelligence, Edinburgh, Scotland, UK, July 30 - August 5, 2005*, pages 193–198. Professional Book Center, 2005.

[21] Javier Larrosa, Federico Heras, and Simon de Givry. A logical approach to efficient max-sat solving. *Artif. Intell.*, 172(2-3):204–233, 2008.

[22] Javier Larrosa and Emma Rollon. Augmenting the power of (partial) maxsat resolution with extension. In *The Thirty-Fourth AAAI Conference on Artificial Intelligence, AAAI 2020, The Thirty-Second Innovative Applications of Artificial Intelligence Conference, IAAI 2020, The Tenth AAAI Symposium on Educational Advances in Artificial Intelligence, EAAI 2020, New York, NY, USA, February 7-12, 2020*, pages 1561–1568. AAAI Press, 2020.

[23] Javier Larrosa and Emma Rollon. Towards a better understanding of (partial weighted) maxsat proof systems. In *Theory and Applications of Satisfiability Testing - SAT 2020 - 23rd International Conference, SAT 2019, Italy, July 3-10, 2020, Proceedings*, Lecture Notes in Computer Science. Springer, 2020.

[24] António Morgado, Federico Heras, Mark H. Liffiton, Jordi Planes, and João Marques-Silva. Iterative and core-guided maxsat solving: A survey and assessment. *Constraints*, 18(4):478–534, 2013.

[25] J. A. Robinson. A machine-oriented logic based on the resolution principle. *J. ACM*, 12(1):23–41, January 1965.

[26] Luca Trevisan, Gregory B. Sorkin, Madhu Sudan, and David P. Williamson. Gadgets, approximation, and linear programming. *SIAM J. Comput.*, 29(6):2074–2097, 2000.

[27] Marc Vinyals, Jan Elffers, Jesús Giráldez-Cru, Stephan Gocht, and Jakob Nordström. In between resolution and cutting planes: A study of proof systems for pseudo-boolean SAT solving. In Olaf Beyersdorff and Christoph M. Wintersteiger, editors, *Theory and Applications of Satisfiability Testing - SAT 2018 - 21st International Conference, SAT 2018, Held as Part of the Federated Logic Conference, FloC 2018, Oxford, UK, July 9-12, 2018, Proceedings*, volume 10929 of *Lecture Notes in Computer Science*, pages 292–310. Springer, 2018.