Refutation of Products of Linear Polynomials

Jan Horáček and Martin Kreuzer

Faculty of Informatics and Mathematics
University of Passau, D-94030 Passau, Germany
Jan.Horacek@uni-passau.de, Martin.Kreuzer@uni-passau.de

Abstract. In this paper we consider formulas that are conjunctions of linear clauses, i.e., of linear equations. Such formulas are very interesting because they encode CNF constraints that are typically very hard for SAT solvers. We introduce a new proof system SRES that works with linear clauses and show that SRES is implicationally and refutationally complete. Algebraically speaking, linear clauses correspond to products of linear polynomials over a ring of Boolean polynomials. That is why SRES can certify if a product of linear polynomials lies in the ideal generated by some other such products, i.e., the SRES calculus decides the ideal membership problem. Furthermore, an algorithm for certifying inconsistent systems of the above shape is described. We also establish the connection with an another combined proof system R(lin).

Keywords: linear clauses, Boolean polynomials, algebraic proof systems, SAT solving

1 Introduction

In this paper we deal with a special case of the ideal membership problem: given a set of Boolean polynomials S which are products of linear polynomials and a Boolean polynomial f which is also a product of linear polynomials, decide whether $f \in \langle S \rangle$. Traditionally, one can use Gröbner bases to solve this problem. The assumption that the given polynomials are products of linear polynomials allows us introduce a new calculus, called SRES, that is tailored to tackle this problem. It needs only a lightweight version of the S-polynomials used Gröbner basis algorithms, namely the s-resolvents we introduced in [6]. These s-resolvents also generalize the classical resolution rule from propositional logic. Nevertheless, we consider SRES to be an algebraic proof system (in the sense of [5], [2] or [4]), and hence we mostly use the terminology of commutative algebra to describe it. The main difference in the algebraic approach is that we study assignments that yield `False`, i.e., are zeros of a system, in constrast to look for assignments that yield `True` in the SAT terminology.

From the propositional logic point of view, we would like to decide if the semantic implication $S \models f$ holds. Recall that a linear Boolean polynomial $x_{i_1} + \cdots + x_{i_j}$ corresponds to a linear XOR constraint $x_{i_1} \oplus \cdots \oplus x_{i_j}$. Such constraints are very difficult for SAT solvers because the truth value depends

genuinely on all variables, i.e., changing the value of one variable results in flipping the truth value of the XOR. Many hard formulas in CNF can be constructed by combining literals into linear XOR constraints, resulting in so-called linear clauses. For details, we refer the readers to [1] or [12]. Such formulas appear quite frequently in cryptography, where linear constraints are typically mixed with highly nonlinear ones, for instance in substitution permutation networks. Another application is to handle the bit-blasted version of ARX operations (Add-Roll-Xor) which is becoming a popular part of ARX ciphers. When converted to bit-wise operations, these conversions give XOR-heavy formulae with a handful of AND gates describing the carry chains of the adders.

The paper is organized as follows. In Section 2 we recall the definition of linear clauses and introduce several ways of describing them, in particular using products of linear Boolean polynomials. Our main data structure to describe them is the set of linear polynomials that appear in the linear clause.

In Section 3 we define the proof system SRES which incorporates and extends the s-resoution rule defined in [6]. We prove that the SRES proof system is implicationally and refutationally complete. Moreover, we establish a relation with other combined systems that use resolution and polynomial calculus (see [11]). In particular, we show that SRES simulates the proof system R(lin) defined in [7, 11].

In Section 4 we describe a concrete algorithm that searches for SRES-refutation proofs of inconsistent systems, called the SRES Refutation Algorithm, and compare it to other approaches to the problem at hand. Finally, in Section 5, we apply the SRES Refutation Algorithm to some examples and point out possible further improvements.

In the following we use some of the definitions and results given in [6], in particular the algorithm `Sres` given there. However, to aid the readers, we tried to make the paper as self-contained as possible and mention any overlaps explicitly.

2 Background

Let $\mathbb{F}_2 = \mathbb{Z}/2\mathbb{Z}$ be the binary field and $\mathbb{F}_2[x_1, \ldots, x_n]$ a polynomial ring over $\mathbb{F}_2$. The ring $\mathbb{B}_n = \mathbb{F}_2[x_1, \ldots, x_n]/F$ with $F = \langle x_1^2 + x_1, \ldots, x_n^2 + x_n \rangle$ is called the **ring of Boolean polynomials** in the indeterminates $x_1, \ldots, x_n$. Let $\mathbb{L}_n$ be the set of all linear polynomials in $\mathbb{F}_2[x_1, \ldots, x_n]$, i.e., the set of all polynomials of degree ≤ 1. (Here we use $\deg(0) = -1$.)

Throughout the paper we consider the obvious correspondences between the following types of objects, where $\ell_1, \ldots, \ell_k \in \mathbb{L}_n$:

(C1) A *set of linear polynomials* $H = \{\ell_1, \ldots, \ell_k\} \subseteq \mathbb{L}_n$.
(C2) A *Boolean polynomial* $h = \prod_{i=1}^{k} \ell_i$ which is a product of linear polynomials in $\mathbb{B}_n$.
(C3) A **linear clause** $(\ell_1 = 0) \vee \cdots \vee (\ell_k = 0)$.

Next we give an example of the above correspondence.

Example 1 Let $H = \{x_1 + x_2 + 1, x_1 + x_3\} \subseteq \mathbb{L}_3$. Then $h = (x_1 + x_2 + 1)(x_1 + x_3) = x_1x_3 + x_1x_2 + x_2x_3 + x_3$, and H (resp. h) coresponds to the propositional logic formula

$$(x_1 \oplus x_2 \oplus 1 = 0) \vee (x_1 \oplus x_3 = 0).$$

To simplify the notation, we view the sets in (C1) as polynomials in (C2), e.g., we speak of zeros of H instead of zeros of h. Furthermore, we always assume that the linear polynomials $\ell_1, \ldots, \ell_k$ appearing in (C1) are pairwise distinct, i.e., that the set H does not contain duplicates. The notion of linear clauses has been considered before in [7, 11]. Note that many difficult CNF instances can be naturally encoded in the form (C3). Thus we are targeting very hard formulas (see [10, Sec. 3]). Furthermore, we observe that two different subsets in (C1) may represent the same polynomial, as the following example shows.

Example 2 Consider the sets of linear polynomials $\{x_3, x_2 + x_3, x_1 + x_3 + 1\}$ and $\{x_3, x_1 + x_2, x_1 + x_3 + 1\}$ as in (C1). Then we have $x_3(x_2 + x_3)(x_1 + x_3 + 1) = x_3(x_1 + x_2)(x_1 + x_3 + 1)$ in $\mathbb{B}_3$, i.e. the corresponding polynomials in (C2) agree.

The subsets in (C1) are measured by degree and size. We recall here some definitions from [6] and [11].

Definition 3 Let $H = \{\ell_1, \ldots, \ell_k\} \subseteq \mathbb{L}_n$.

(1) The number $\deg(h) = \#H$ is called the **degree** of H.
(2) For $\ell \in \mathbb{L}_n$, we let $\operatorname{var}(\ell)$ be the set of indeterminates occurring in ℓ. We call the number $\operatorname{size}(H) = \#\operatorname{var}(\ell_1) + \cdots + \#\operatorname{var}(\ell_k)$ the **size** of H.

Our goal in this paper is to show that a given set of linear clauses is unsatisfiable. For this purpose, we define semantic implication as follows.

Definition 4 Let $F_1, \ldots, F_m, H \subseteq \mathbb{L}_n$. We say that $F_1, \ldots, F_m$ **semantically imply** H if every $\mathbb{F}_2$-rational common zero of the Boolean polynomials corresponding to $F_1, \ldots, F_m$ is a zero of H. In this case we write $F_1, \ldots, F_m \models H$.

From the algebraic point of view, we have $F_1, \ldots, F_m \models H$ if and only if $H \in \langle F_1, \ldots, F_m \rangle \subseteq \mathbb{B}_n$. Note that the zeros of an ideal in $\mathbb{B}_n$ are always $\mathbb{F}_2$-rational, because ideals in $\mathbb{B}_n$ correspond to ideals in $\mathbb{F}_2[x_1, \ldots, x_n]$ that contain the **field ideal** $\langle x_1^2 + x_1, \ldots, x_n^2 + x_n \rangle$. Using the newly defined terminology, we can say that our goal is to **refute** a given set of linear clauses, i.e. to prove that the corresponding sets of linear polynomials semantically imply $\{1\}$.

3 The SRES Proof System

Recall that a **proof system**, sometimes called a *Hilbert system* or *Hilbert calculus*, consists of a syntax, i.e. a set of rules which determine the set of well-formed formulas of the system, by a set of axioms, i.e. a set of formulas which are assumed to be tautologies, and by its set of rules of inference, i.e. by a set of rules which determine how one can get new tautologies from known ones.

For instance, the **Gröbner proof system** defined in [5] admits all formulas of the form $f(x_1,\ldots,x_n)=0$ where $f\in\mathbb{F}_2[x_1,\ldots,x_n]$. Its axioms are the **Boolean axioms** $x_i^2+x_i=0$ for $i=1,\ldots,n$, and its rules of inference are

$$\frac{f \qquad g}{f+g}\ (\mathcal{P}_a) \qquad \text{and} \qquad \frac{f}{x_i f}\ (\mathcal{P}_m)$$

where $f,g\in\mathbb{F}_2[x_1,\ldots,x_n]$ and x_i is an indeterminate.

In this section we continue to study the proof system based on s-resolution which was introduced in [6]. More precisely, we extend that system by other inference rules, and thus we define the new proof system called SRES. Let us begin by recalling the proof system in [6].

Definition 5 The proof system **s-resolve** is defined by the following parts:

(1) A formula is a set of linear polynomials $\{\ell_1,\ldots,\ell_s\}\subset\mathbb{L}_n$, where $s\in\mathbb{N}_+$.
(2) The axioms are the Boolean axioms given by $\{x_i,x_i+1\}$ for $i=1,\ldots,n$.
(3) There exists one rule of inference $(\mathcal{R})$, namely

$$\frac{\bigcup_{i=1}^{s}\{\ell_i\}\cup G \qquad\qquad \bigcup_{i=1}^{s}\{\ell_i+1\}\cup\tilde{G}}{\bigcup_{i=1}^{s-1}\{\ell_i+\ell_{i+1}+1\}\cup G\cup\tilde{G}}\ (\mathcal{R})$$

where $G,\tilde{G}\subseteq\mathbb{L}_n$ and $s\geq 1$. This rule is also called s**-resolution**. (For $s=1$, we let $\bigcup_{i=1}^{s-1}\{\ell_i+\ell_{i+1}+1\}=\emptyset$.) The result of an application of the s-resolution rule is called an s**-resolvent**.

Let us note that the Boolean axioms immediately imply the following remark.

Remark 6 By applying 2-resolution to the axioms $\{x_i,x_i+1\}$ and $\{x_i+1,x_i\}$, we obtain that $\{0\}$ is a tautology in the s-resolve proof system.

In [6] we showed that s-resolve is correct in the sense of the following definition.

Definition 7 Let $(\mathcal{I})$ be a rule of inference of a proof system. We say that the rule of inference $(\mathcal{I})$ is **correct** if

$$\frac{F_1 \qquad F_2 \qquad \ldots \qquad F_m}{H}\ (\mathcal{I})$$

for some formulas $F_1,\ldots,F_m,H$ implies $F_1,\ldots,F_m\models H$.

For $s\geq 3$, the s-resolution rule depends e.g. on the numbering of the linear polynomials. (2-resolvents are unique because there is only one way how to form the linear polynomial $\ell_1+\ell_2+1$.) However, considered as a Boolean polynomial, the s-resolvent is uniquely determined. The next example is a point in case.

Example 8 Resolving two sets $F = \{x_1 + 1, x_1 + x_3, x_1 + x_2 + 1, x_2 + x_3 + 1\}$ and $G = \{x_1, x_1 + x_3 + 1, x_1 + x_2, x_2 + x_3\}$ with the numbering $\ell_1 = x_1 + 1$, $\ell_2 = x_1 + x_3$, $\ell_3 = x_1 + x_2 + 1$, $\ell_4 = x_2 + x_3 + 1$ yields the 4-resolvent $R_1 = \{x_3, x_2 + x_3, x_1 + x_3 + 1\}$. If we swap the last two polynomials in G, 4-resolution with the numbering $\ell_1 = x_1 + 1$, $\ell_2 = x_1 + x_3$, $\ell_4 = x_1 + x_2 + 1$, $\ell_3 = x_2 + x_3 + 1$ yields $R_2 = \{x_3, x_1 + x_2, x_1 + x_3 + 1\}$. Both R_1 and R_2 correspond to the same Boolean polynomial, as we saw in Example 2.

Next we enrich s-resolve with a new rule of inference as follows.

Definition 9 The proof system **SRES** is defined by the following parts.

(1) The syntax agrees with the syntax of s-resolve, i.e. a formula is a set of linear polynomials $\{\ell_1, \ldots, \ell_s\} \subset \mathbb{L}_n$, where $s \in \mathbb{N}_+$.
(2) The axioms are the Boolean axioms $\{x_i, x_i + 1\}$ for $i = 1, \ldots, n$.
(3) The rules of inference consist of s-resolution ($\mathcal{R}$) and the following **weakening rule** ($\mathcal{W}$):

$$\frac{H}{H \cup \{\ell\}} \ (\mathcal{W})$$

for $H \subseteq \mathbb{L}_n$ and $\ell \in \mathbb{L}_n$.

Since we trivially have $H \models H \cup \{\ell\}$, the proof system SRES is correct with respect to semantic implication. Next we recall the definition of a proof.

Definition 10 Let PS be a proof system. A PS-**proof** of a formula H from the **initial premises** $F_1, \ldots, F_m$ in the proof system PS is a sequence of formulas $\pi = (G_1, \ldots, G_k)$ such that $G_k = H$ and each of the formulas G_i is of one of the following forms:

(1) $G_i \in \{F_1, \ldots, F_m\}$
(2) G_i is one of the axioms of PS.
(3) G_i is obtained from some of the formulas G_j with $j < i$ by applying one of the rules of inference of the proof system PS.

If a formula H has a PS-proof from $\{F_1, \ldots, F_m\}$, we write $F_1, \ldots, F_m \vdash_{\mathrm{PS}} H$, or simply $F_1, \ldots, F_m \vdash H$ if no confusion can arise.

Since the proof system SRES is correct, our goal to refute $\{F_1, \ldots, F_m\}$ can now be expressed by asking that we should prove $F_1, \ldots, F_m \vdash \{1\}$. Notice that the intended semantics is that an unsatisfiable formula corresponds to polynomial equations $\ell_{i,1} \cdots \ell_{i,s} = 0$ for $i \in \{1, \ldots, m\}$ which has no solution. Equivalently, a tautology is an equation which holds for all points of $\mathbb{F}_2^n$. A proof can be seen as a sequence of applications of rules to axioms or previously proved tautologies.

Next we extend the capabilities of our proof system by deriving the following further rules of inference.

Definition 11 Let F, G be subsets of $\mathbb{L}_n$, and let $\ell, \ell_1, \ell_2 \in \mathbb{L}_n$.

(1) The rule $(\mathcal{U})$ defined by

$$\frac{H \cup \{1\}}{H} \ (\mathcal{U})$$

is called **unit cancellation**.

(2) The rule $(\mathcal{MP})$ defined by

$$\frac{\{\ell\} \qquad H \cup \{\ell + 1\}}{H} \ (\mathcal{MP})$$

is called **modus ponens**.

(3) The rule $(\mathcal{A})$ defined by

$$\frac{F \cup \{\ell_1\} \qquad H \cup \{\ell_2\}}{F \cup H \cup \{\ell_1 + \ell_2\}} \ (\mathcal{A})$$

is called the **addition rule**.

Proposition 12 The rules $(\mathcal{U})$, $(\mathcal{MP})$, and $(\mathcal{A})$ are correct. Furthermore, any proof derived using $(\mathcal{U}), (\mathcal{MP})$ and $(\mathcal{A})$ can be rewritten into an SRES-proof.

Proof. The correctness of $(\mathcal{U})$ follows from the observation that multiplying a Boolean polynomial by 1 does not change its zeros. Using Remark 6 we apply 1-resolution on $H \cup \{1\}$ and $\{0\}$, and we get H. To prove modus ponens it suffices to use $(\mathcal{R})$ with $s = 1$ and $G = \emptyset$. Finally, we show the correctness of $(\mathcal{A})$. Using the weakening rule, we infer $F \cup \{\ell_1, \ell_2 + 1\}$ from $F \cup \{\ell_1\}$ and $H \cup \{\ell_2, \ell_1 + 1\}$ from $H \cup \{\ell_2\}$. Then, using 2-resolution, we get $F \cup H \cup \{\ell_1 + \ell_2\}$. □

The following remark and the subsequent proposition will become important later in the proof of Proposition 18.

Remark 13 Let $\pi = (H_1, \ldots, H_k)$ be an SRES-proof of H_k from $F_1, \ldots, F_m$ that uses only the s-resolution rule. Let ℓ be an element of one of the sets H_i. Then ℓ lies in the $\mathbb{F}_2$-vector space generated by the linear polynomials contained in the union $\bigcup_{i=1}^{m} F_i$.

Proposition 14 Let $F \subseteq \mathbb{L}_n$ and let $i \in \{1, \ldots, n\}$. For $a \in \{0, 1\}$, let $F(x_i \mapsto a)$ denote the set which is obtained by substituting $x_i \mapsto a$ into the linear polynomials contained in F. Then the following claims hold true for $i \in \{1, \ldots, n\}$.

(1) $F, \{x_i\} \vdash_{\text{SRES}} F(x_i \mapsto 0)$
(2) $F(x_i \mapsto 0), \{x_i\} \vdash_{\text{SRES}} F$
(3) $F, \{x_i + 1\} \vdash_{\text{SRES}} F(x_i \mapsto 1)$
(4) $F(x_i \mapsto 1), \{x_i + 1\} \vdash_{\text{SRES}} F$

Proof. First we prove (1). Let $\ell \in F$ be such that x_i occurs in ℓ. The polynomial $x_i + \ell$ corresponds to substituting $x_i = 0$ into ℓ. This addition can be derived in SRES using Proposition 12. The claims (2)–(4) follow analogously from Proposition 12. □

The next example illustrates this proposition.

Example 15 Let $F_1 = \{x_1 + x_2, x_1\}$ and $F_2 = \{x_1 + 1\}$. By Proposition 12, there exists an SRES-proof of the set $\{x_2 + 1, 1\}$ from F_1, F_2 which corresponds to the substitution of $x_1 = 1$ into F_1. On the other hand, we can "backtrack" the substitution, i.e. we have $F_2, \{x_2 + 1, 1\} \vdash_{\mathrm{SRES}} F_1$.

The following example shows an important advantage of the SRES proof system over the Gröbner proof system defined in [5]. More precisely, it shows that the data structures (C1)–(C3) efficiently store dense linear clauses.

Example 16 Consider the sets

$$\begin{aligned} F_1 &= \{x_1 + x_2,\ \ldots,\ x_n + x_{n+1}\} \\ F_2 &= \{x_1 + x_2 + 1\} \\ F_3 &= \{x_2 + x_3 + 1\} \\ \vdots &= \ \vdots \\ F_{n+1} &= \{x_n + x_{n+1} + 1\} \end{aligned}$$

On one hand, it is easy to see that the system is inconsistent because substituting $F_2, \ldots, F_{n+1}$ into F_1 gives us 1. On the other hand, the input for the Gröbner basis algorithm is assumed to be expanded (i.e., not in the form of products of linear polynomials). We can always find $n \in \mathbb{N}$ such that expanding F_1 to the polynom $f_1 = \prod_{i=1}^{n}(x_i + x_{i+1})$, which has 2^n terms, exceeds the available memory, and hence any Gröbner basis algorithm can not be applied. Notice that we have $\mathrm{size}(F_1) = 2n$.

One workaround would be to introduce new indeterminates to break the long product, but it does not help too much because the Gröbner basis algorithm substitutes the new indeterminates back, and thus recovers the expanded polynomial F_1 again.

However, by Proposition 14, there exists a short SRES refutation that corresponds to the substitution of $F_2, \ldots, F_{n+1}$ into F_1.

The following definition provides further useful properties of proof systems.

Definition 17 (1) A proof system is called **implicationally complete** if for every formula H and every set of formulas $\{F_1, \ldots, F_m\}$ such that we have $F_1, \ldots, F_m \models H$, there exists a proof of H from $F_1, \ldots, F_m$ in this proof system.

(2) A proof system is called **refutationally complete** if for every inconsistent set of formulas $\{F_1, \ldots, F_m\}$, i.e. for $F_1, \ldots, F_m \models \{1\}$, there exists a proof of $\{1\}$ from $F_1, \ldots, F_m$ in this proof system.

The proof of the following proposition is inspired by the corresponding proof for the classical resolution calculus (see for instance [3, Th. 4.1.5]) and by the proof given in [11, Th. 5.1])

Proposition 18 The proof system SRES is implicationally and refutationally complete.

Proof. First we show that SRES is implicationally complete. Let $F_1, \ldots, F_m, H \subseteq \mathbb{L}_n$ be sets of linear polynomials such that $F_1, \ldots, F_m \models H$. We want to prove that $F_1, \ldots, F_m \vdash_{\mathrm{SRES}} H$. We proceed by induction on the number $k = \mathrm{size}(F_1) + \cdots + \mathrm{size}(F_m) + \mathrm{size}(H)$.

Let us consider the case $k = 0$, i.e. the case when all linear polynomials in F_i and H are constants 0 or 1. If all sets $F_1, \ldots, F_m$ are equal to $\{0\}$, there is trivially an SRES-proof of $\{0\}$. All the semantic implicants of $\{0\}$ of size 0 can be derived from $\{0\}$ by the weakening rule. If there exists a set $F_i = \{1\}$, then there is trivially an SRES-proof that refutes $F_1, \ldots, F_m$, and hence we can derive any linear clause by the weakening rule and the unit cancellation rule. Finally, if $F_i = \{0, 1\}$, then F_i is simplified to $\{0\}$ by unit cancellation, and thus we can use the previous argument.

Now assume that the claim holds for some $k \geq 0$ and consider a semantic implication

$$F_1, \ldots, F_m \models H$$

in which the sum of sizes of $F_1, \ldots, F_m, H$ is at most $k+1$. Choose $i \in \{1, \ldots, n\}$ such that x_i occurs in $F_1 \cup \cdots \cup F_m \cup H$. Given $a \in \{0, 1\}$, let $F(x_i \mapsto a)$ denote the set which is obtained by substituting $x_i \mapsto a$ into the linear polynomials contained in F.

By Proposition 14, we have $F_j, \{x_i\} \vdash_{\mathrm{SRES}} F_j(x_i \mapsto 0)$ for $j = 1, \ldots, m$. Moreover, we clearly have $F_1(x_i \mapsto 0), \ldots, F_m(x_i \mapsto 0) \models H(x_i \mapsto 0)$. By the induction hypothesis, there is an SRES-proof of $H(x_i \mapsto 0)$ from $F_1(x_i \mapsto 0), \ldots, F_m(x_i \mapsto 0)$. Altogether, we get

$$\{x_i\}, F_1, \ldots, F_m \vdash_{\mathrm{SRES}} H(x_i \mapsto 0).$$

Furthermore, by Proposition 14, we have $H(x_i \mapsto 0), \{x_i\} \vdash_{\mathrm{SRES}} H$. All in all, there is an SRES-proof π_1 of H from $\{x_i\}, F_1, \ldots, F_m$. Analogously, we get an SRES-proof π_2 of H from $\{x_i + 1\}, F_1, \ldots, F_m$.

Next we modify the derivations of π_1 (resp. π_2) such that they start from $\{x_i, x_i + 1\}, F_1, \ldots, F_m$ instead of $\{x_i\}, F_1, \ldots, F_m$ (resp. $\{x_i + 1\}, F_1, \ldots, F_m$). Note that the same rules of inference can be applied, and thus one can rewrite the proof π_1 into an SRES-proof α_1 from $\{x_i, x_i + 1\}, F_1, \ldots, F_m$ of either H or $H \cup \{x_i\}$. Similarly, we rewrite π_2 into an SRES-proof α_2 from $\{x_i, x_i + 1\}, F_1, \ldots, F_m$ of H or $H \cup \{x_i + 1\}$. If the proof α_1 or α_2 ends with H, we are done. Otherwise, we have $H \cup \{x_i\}, H \cup \{x_i + 1\} \vdash_{\mathrm{SRES}} H$ by a single step of the 1-resolution rule, which concludes the proof. □

The SRES proof system is in fact a combined proof system in the sense of [8, Sec.7.1]. For instance, R(lin) in [11] is an another example of a combined proof

systems that combines resolution with polynomial calculus. By Proposition 12 we immediately get that SRES efficiently **simulates** the system R(lin). This means that there exists a polynomial-time algorithm that translates any R(lin)-proof of H from $F_1, \ldots, F_m$ to an SRES-proof of H from $F_1, \ldots, F_m$.

On the other hand, SRES cannot simulate the rule $(\mathcal{P}_a)$ of the Gröbner proof system because addition of two products of linear polynomials does not have to be a product of linear polynomials in $\mathbb{B}_n$, e.g., $x_1 \cdot x_2 + 1$ cannot be written as a product of linear polynomials in $\mathbb{B}_3$.

4 The SRES Refutation Algorithm

In [6] the authors introduced an algorithm that finds refutation proofs for unsatisfiable formulas in CNF using s-resolution. In this section we generalize this result to formulas that are conjuctions of linear clauses and show that it is refutationally complete in the sense of Definition 17. We recall the following algorithm for computing the s-resolvent of two polynomials which is described in [6].

Algorithm 1 `Sres` (s-Resolution of Two Polynomials)
Input: Sets $F, G \subseteq \mathbb{L}_n$. We assume that $\#\{\ell \in F \mid \ell + 1 \in G\} \geq 1$.
Output: A set $R \subseteq \mathbb{L}_n$ such that R is the s-resolvent of F and G.

```
1: Write {ℓ ∈ F | ℓ + 1 ∈ G} as {ℓ_1, ..., ℓ_s}.
2: F' := {ℓ ∈ F | ℓ + 1 ∉ G}
3: G' := {ℓ ∈ G | ℓ + 1 ∉ F}
4: R := F' ∪ G'
5: if s = 1 and R = ∅ then
6:    return {1}
7: else
8:    for i = 1, ..., s − 1 do
9:       if ℓ_i + ℓ_{i+1} ∉ R then
10:         R := R ∪ {ℓ_i + ℓ_{i+1} + 1}
11:      else
12:         return {0}
13:      end if
14:   end for
15: end if
16: return R
```

Algorithm 2 extends two polynomials by the weakening rule in all possible ways such that s-resolution can be applied. More precisely, this set of extensions is defined as follows.

Definition 19 Let $F, G \subseteq \mathbb{L}_n$. The set of all pairs $K \subseteq \mathbb{L}_n \times \mathbb{L}_n$ such that the following two conditions hold for all $(F', G') \in K$

(1) $\#\{\ell \in F' \mid \ell + 1 \in G'\} \geq 1$

(2) $F' \cup G' \subseteq F \cup G \cup \{\ell + 1 \mid \ell \in F \cup G\}$

is called the **set of expansions** for F, G.

Condition (1) encodes the fact that there exists at least one $\ell \in \mathbb{L}_n$ in F' and G' on which we can s-resolve. Condition (2) restrains F', G' to contain linear polynomials ℓ or $\ell + 1$ such that $\ell \in F \cup G$. Note that the set of expansions for F, G is unique.

Algorithm 2 produces the set of expansions in a brute-force way, i.e., Condition (1) is implemented in Step 10, and Condition (2) is fulfilled because of the two foreach loops in Steps 3, 4.

Algorithm 2 `AllExpansions` (All Possible Expansions to s-Resolution)
Input: Sets $F, G \subseteq \mathbb{L}_n$.
Output: The set of expansions for F, G.

1: $\{\ell_1, \ldots, \ell_k\} := \{\ell \in F \mid \ell \notin G, \ell + 1 \notin G\}$
2: $\{\ell'_1, \ldots, \ell'_{k'}\} := \{\ell \in G \mid \ell \notin F, \ell + 1 \notin F\}$
3: **foreach** $A \in \{\ell_1, \ell_1 + 1, 1\} \times \cdots \times \{\ell_k, \ell_k + 1, 1\}$ **do**
4: **foreach** $B \in \{\ell'_1, \ell'_1 + 1, 1\} \times \cdots \times \{\ell'_{k'}, \ell'_{k'} + 1, 1\}$ **do**
5: Write $A = (a_1, \ldots, a_k)$.
6: Write $B = (b_1, \ldots, b'_k)$.
7: $F' := F \cup \bigcup_{i=1}^{k} \{a_i\}$
8: $G' := G \cup \bigcup_{i=1}^{k'} \{b_i\}$
9: Minimize the representation of F' and G' by applying unit cancellation, i.e., remove the element 1 from F', G'.
10: **if** $\#\{\ell \in F' \mid \ell + 1 \in G'\} \geq 1$ **then**
11: $K := K \cup \{(F', G')\}$
12: **end if**
13: **end foreach**
14: **end foreach**
15: **return** K

The next example shows the generation of the pairs in Algorithm 2.

Example 20 Let $F = \{x_1, x_2, x_3 + 1\}$ and $G = \{x_1 + 1, x_2, x_4\}$. Then we may extend F to F itself, to $\{x_1, x_2, x_3 + 1, x_4\}$, or to $\{x_1, x_2, x_3 + 1, x_4 + 1\}$. Similarly, we may extend G to G, to $\{x_1 + 1, x_2, x_4, x_3 + 1\}$, or to $\{x_1 + 1, x_2, x_4, x_3\}$. Altogether, nine pairs are constructed.

Next we process the pairs computed by Algorithm 2 in increasing order with respect to the following ordering relation.

Definition 21 Let $F, G, F_1, F_2, G_1, G_2 \subseteq \mathbb{L}_n$.

(1) We write $F \preceq G$ if we have $\deg(F) < \deg(G)$, or if we have $\deg(F) = \deg(G)$ and $\text{size}(F) < \text{size}(G)$.

(2) Let $\#\{\ell \in F_1 \mid \ell + 1 \in G_1\} \geq 1$ and $\#\{\ell \in F_2 \mid \ell + 1 \in G_2\} \geq 1$. We write $(F_1, G_1) \trianglelefteq (F_2, G_2)$ if $\texttt{Sres}(F_1, G_1) \preceq \texttt{Sres}(F_2, G_2)$.

Note that the size of the s-resultants can be determined without actually executing `Sres` (see [6, Alg. 8]). Now we are ready to present Algorithm 3 for constructing SRES-refutations. We assume that all sets occurring in the algorithm do not contain duplicates, i.e., that removal of duplicates is applied whenever possible. This is the classical assumption on sets in programming languages such as `python`.

Algorithm 3 `SRES Refute` (SRES Refutation Algorithm)
Input: Subsets $F_1, \ldots, F_m \subseteq \mathbb{L}_n$ such that F_i does not contain any duplicate elements.
Output: `False` if $F_1, \ldots, F_m \models \{1\}$, `True` otherwise.
Require: Algorithms 1, 2.

$S := \{F_1, \ldots, F_m\}$
if $\{1\} \in S$ **then**
  return `False`
end if
Apply unit simplification and cancellation on S.
$\{Q_1, \ldots, Q_k\} := S \cup \bigcup_{i=1}^{n} \{x_i, x_i + 1\}$
Let P be the list containing all pairs computed by $\texttt{AllExpansions}(Q_i, Q_j)$ for $1 \leq i < j \leq k$.
while $P \neq \emptyset$ **do**
  Let (F, G) be a minimum of P w.r.t. $\trianglelefteq$, and remove (F, G) from P.
  $R := \texttt{Sres}(F, G)$
  if $R = \{1\}$ **then**
    return `False`
  else if R is not a subset of any $Q \in S$ **then**
    Remove all Q from S with $R \subsetneq Q$ and all pairs (Q_1, Q_2) from P such that $R \subsetneq Q_1$ or $R \subsetneq Q_2$.
    Append all pairs in $\texttt{AllExpansions}(R, G)$ for $G \in S, R \neq G$ to the list P.
    $S := S \cup \{R\}$
  end if
end while
return `True`

Proposition 22 Algorithm 3 is correct, refutationally complete, and finite. In particular, the algorithm returns `False` if and only if $F_1, \ldots, F_m \models \{1\}$.

Proof. Finiteness follows from the fact that there are only finitely many linear polynomials in $\mathbb{L}_n$. Hence the sets in S are finite, and so is the set $P \subseteq \mathbb{L}_n \times \mathbb{L}_n$.

The algorithm is correct because the SRES inference rules semantically imply their results.

It remains to show that the algorithm is refutationally complete. Assume that the ideal generated by the polynomials corresponding to the input sets has no

common zero. By Proposition 18, we know that there exists an SRES-refutation of $F_1, \ldots, F_m$. The Boolean axiom is incorporated in Step 6, and weakening takes place in the function `AllExpansions` such that all possible choices to form an s-resolvent are created for all possible $s \in \mathbb{N}_+$. The s-resolution rule is then applied by calling `Sres`. Any set Q that is a proper superset of some set already occurring in S is ignored because all possible s-resolvents that would be created using Q are at that time already in P. Thus the algorithm sequentially creates all SRES-derivations from $F_1, \ldots, F_m$. It arranges the computation such that "smaller" sets in terms of size are preferred. Since Proposition 18 shows that there exists an SRES-refutation, the set $\{1\}$ is eventually discovered by the algorithm. □

The list P in Algorithm 3 can be implemented as a min-heap such that the minimal pair is always easily found and extracted. The number of pairs in P may be huge. Thus it is convenient to compute the s-resolvents only in Step 10. The next example indicates how the pairs can be stored. The s-resolvent is created only if its size is minimal.

Example 23 Let $F_1 = \{x_1 + 1\}$ and $F_2 = \{x_1, x_2\}$. Algorithm `AllExpansions` outputs (F_1, F_2), $(F_1 \cup \{x_2 + 1\}, F_2)$, $(F_1 \cup \{x_2\}, F_2)$. The pair $(F_1 \cup \{x_2\}, F_2)$ can be stored as a tuple $(1, \{x_2\}, 2, \emptyset, 1)$ with the meaning "`Sres` of $F_1 \cup \{x_2\}$ and $F_2 \cup \emptyset$ has size 1".

Recall that we need to know the sizes of all s-resultants in P in order to select the minimum. Thus the chosen format is very convenient because the size of the s-resolvent can be predicted as in [6, Alg. 8] without computing the actual s-resolvents.

Furthermore, one can form only 2-resolvents in Algorithm 3 since 2-resolution is enough to simulate R(lin) which is implicationally and refutationally complete. However, "extra" linear clauses coming from s-resolution steps with $s \geq 3$ may come in handy, and the refutation can be found faster in Algorithm 3.

5 Examples and Future Directions

In this section we give some examples of the SRES proof system. Initial experiments on refuting CNF formulae using SRES can be found in [6, Sec. 9]. Those experiments were focused on comparing resolution with SRES based on CNF benchmarks coming from [10]. These formulae are hard for classical resolution, but there may exist short refutations in other axiomatic systems of propositional calculus [13].

In the following example taken from [5], we compare SRES with algebraic systems such as the Gröbner proof system and the Nullstellensatz system [4, Def. 2.1]. Let $P = \mathbb{F}_2[x_1, \ldots, x_n]$. A **Nullstellensatz proof** of a polynomial $h \in P$ from polynomials $f_1, \ldots, f_m \in P$ is a tuple of polynomials $(p_1, \ldots, p_m, r_1, \ldots, r_n) \in P^{m+n}$ such that the equation

$$\sum_{i=1}^{m} p_i f_i + \sum_{j=1}^{n} r_j(x_j^2 + x_j) = h$$

holds in P. The degree of the proof is defined as

$$\max\big\{ \max_i \big(\deg(p_i) + \deg(f_i) \big), \max_j \big(\deg(r_j) + 2 \big) \big\}.$$

Example 24 Consider the polynomials $f_1 = x_1 + x_1x_2$ and $f_2 = x_2 + x_2x_3$ in $\mathbb{F}_2[x_1, x_2, x_3]$. They encode two implications $x_1 \to x_2$ and $x_2 \to x_3$. (E.g., if $x_1 = 1$, then x_2 is constrained to be 1.) Let us write a proof of $h = x_1 + x_1x_3$, i.e., the implication $x_1 \to x_3$, in the Gröbner proof system.

Firstly, we multiply f_2 by x_1, and we get $g_1 = x_1x_2 + x_1x_2x_3$. The addition $f_1 + g_1$ gives us $x_1 + x_1x_2x_3$. Then we compute $g_2 = x_3 \cdot f_1 = x_1x_3 + x_1x_2x_3$. Finally, we get $g_1 + g_2 = x_1 + x_1x_3$.

Note that the maximal degree appearing in the proof is 3. On the other hand, there exists a Nullstellensatz proof of the maximal degree $\mathcal{O}(\log(n))$ of $x_1 \to x_n$ from $x_1 \to x_2, \ldots, x_{n-1} \to x_n$ (see [5]). In our case, the Nullstellensatz proof is the tuple $(1 + x_3, x_1, 0, 0, 0)$ because of the equality

$$(1 + x_3)(x_1 + x_1x_2) + x_1(x_2 + x_2x_3) = x_1 + x_1x_3.$$

Now we write the polynomials f_1, f_2 as $F_1 = \{x_1, 1 + x_2\}$ and $F_2 = \{x_2, 1 + x_3\}$. Resolving on x_2 yields $H = \{x_1, 1+x_3\}$ which correponds to the polynomial h. Note that the maximal degree in the proof is now 2, and the SRES proof is simpler than the Gröbner proof.

Further typical examples that are easy for SRES and difficult for resolution [13] are inconsistent systems of linear equations. By Proposition 12, SRES simulates the addition rule, and thus one can use Gaußian elimination to produce SRES-refutations for such systems. Recall the encoding into linear clauses in Example 16 is very efficient for SRES, but multiplying out the products causes an exponential blowup. A similar problem emerges in the next example in the case of CNF encodings.

Example 25 Consider the linear system

$$f_1 = x_1 + \cdots + x_n, \;\; f_2 = x_1 + \cdots + x_n + 1$$

over $\mathbb{F}_2$. On one hand, encoding f_1 and f_2 in CNF suffers from introducing either exponentially many auxiliary variables or exponentially many new clauses. On the other hand, by the weakening rule and 2-resolution we get $f_1(f_2 + 1) + (f_1 + 1)f_2 = 1$ in $\mathbb{B}_n$ immediately.

Let us conclude this section and this paper with a few remarks about the problem that is solved by Algorithm 3. The traditional way in Computer Algebra is to use Gröbner basis algorithms. As we saw in Example 16, they have serious drawbacks in our setting and tend to run out of memory quickly.

Another approach is to use a SAT solver, e.g. a version of the DPLL algorithm. However, the DPLL approach needs a huge number of clauses to describe XOR constraints. All in all, the classical DPLL algorithm can be extended to linear clauses in a rather straightforward way, as for instance done in [7], but appears to be not very efficient.

The new approach introduced here, namely the SRES proof system and the SRES Refutation Algorithm, offers the prospect of a new and tailor-made way to treat systems of linear clauses. Besides finding possible optimizations of the implementation, we may enhance it further by combining it with a suitable conflict-learning mechanism such as the ones which lie at the heart of modern SAT solvers. Recall that conflict clauses in SAT solvers can be generated from 1-resolvents of a cut in the implication graph. Thus, s-resolution may generalize conflict learning in the case of linear clauses or help to improve a separate XOR-reasoning module such as one introduced in [9].

Acknowledgments. The authors thank Jan Krajíček and Iddo Tzameret for fruitful discussions on combined proof systems. We thank the anonymous reviewers for their many insightful comments. This work was financially supported by the DFG project "Algebraische Fehlerangriffe" [KR 1907/6-2].

References

1. Baumgartner, P., and Massacci, F. The taming of the (X)OR. In *Computational LogicCL 2000*. Springer, 2000, pp. 508–522.
2. Berkholz, C. The relation between polynomial calculus, Sherali-Adams, and sum-of-squares proofs. In *LIPIcs-Leibniz International Proceedings in Informatics* (2018), vol. 96, Leibniz-Zentrum fuer Informatik, Schloss Dagstuhl.
3. Büning, H. K., and Lettmann, T. *Propositional logic: deduction and algorithms*, vol. 48. Cambridge University Press, 1999.
4. Buss, S., Impagliazzo, R., Krajíček, J., Pudlák, P., Razborov, A. A., and Sgall, J. Proof complexity in algebraic systems and bounded depth frege systems with modular counting. *Computational Complexity 6*, 3 (1996), 256–298.
5. Clegg, M., Edmonds, J., and Impagliazzo, R. Using the Groebner basis algorithm to find proofs of unsatisfiability. In *Proceedings of the 28th Annual ACM Symposium on the Theory of Computing* (New York, USA, 1996), STOC '96, ACM Press, pp. 174–183.
6. Horáček, J., and Kreuzer, M. On conversions from CNF to ANF. *(submitted)* (2018).
7. Itsykson, D., and Sokolov, D. Lower bounds for splittings by linear combinations. In *International Symposium on Mathematical Foundations of Computer Science* (2014), Springer, pp. 372–383.
8. Krajicek, J. Proof complexity (book draft). Available at https://www.karlin.mff.cuni.cz/~krajicek/prfdraft2.pdf, 2018.
9. Laitinen, T., Junttila, T., and Niemelä, I. Conflict-driven xor-clause learning. In *Theory and Applications of Satisfiability Testing – SAT 2012* (Berlin, Heidelberg, 2012), A. Cimatti and R. Sebastiani, Eds., Springer Berlin Heidelberg, pp. 383–396.

10. Lauria, M., Elffers, J., Nordström, J., and Vinyals, M. CNFgen: A generator of crafted benchmarks. In *Theory and Applications of Satisfiability Testing – SAT 2017* (Cham, 2017), S. Gaspers and T. Walsh, Eds., Springer International Publishing, pp. 464–473.
11. Raz, R., and Tzameret, I. Resolution over linear equations and multilinear proofs. *Annals of Pure and Applied Logic 155*, 3 (2008), 194 – 224.
12. Soos, M., Nohl, K., and Castelluccia, C. Extending SAT solvers to cryptographic problems. In *International Conference on Theory and Applications of Satisfiability Testing* (2009), Springer, pp. 244–257.
13. Urquhart, A. Hard examples for resolution. *J. ACM 34*, 1 (Jan. 1987), 209–219.