New Results on Cutting Plane Proofs for Horn Constraint Systems

Hans Kleine Büning
Universität Paderborn, Paderborn, Germany
kbcsl@uni-paderborn.de

Piotr Wojciechowski
LDCSEE, West Virginia University, Morgantown, WV, USA
pwjociec@mix.wvu.edu

K. Subramani
LDCSEE, West Virginia University, Morgantown, WV, USA
k.subramani@mail.wvu.edu

Abstract

In this paper, we investigate properties of cutting plane based refutations for a class of integer programs called *Horn constraint systems (HCS).* Briefly, a system of linear inequalities $\mathbf{A} \cdot \mathbf{x} \geq \mathbf{b}$ is called a Horn constraint system, if each entry in $\mathbf{A}$ belongs to the set $\{0, 1, -1\}$ and furthermore there is at most one positive entry per row. Our focus is on deriving refutations i.e., proofs of unsatisfiability of such programs using cutting planes as a proof system. We also look at several properties of these refutations. Horn constraint systems can be considered as a more general form of propositional Horn formulas, i.e., CNF formulas with at most one positive literal per clause. Cutting plane calculus (CP) is a well-known calculus for deciding the unsatisfiability of propositional CNF formulas and integer programs. Usually, CP consists of a pair of inference rules. These are called the addition rule (ADD) and the division rule (DIV). In this paper, we show that cutting plane calculus is still complete for Horn constraints when every intermediate constraint is required to be Horn. We also investigate the lengths of cutting plane proofs for Horn constraint systems.

2012 ACM Subject Classification Theory of computation → Constraint and logic programming

Keywords and phrases Horn constraints, cutting planes, proof length

Digital Object Identifier 10.4230/LIPIcs.FSTTCS.2019.43

Funding This research was supported in part by the Air-Force Office of Scientific Research through Grant FA9550-19-1-017.

1 Introduction

This paper is concerned with the length of tree-like and Dag-like cutting plane refutations of Horn constraint systems (HCSs). HCSs are a type of polyhedral constraint system in which, each constraint is of the form $\mathbf{a} \cdot \mathbf{x} \geq b$, coefficients are limited to the set $\{0, 1, -1\}$, and each constraint has at most one variable with positive coefficient. HCSs find important applications in several problem domains [6].

A refutation of a system of constraints is a certificate that proves the infeasibility of that system. Associated with the concept of certificates is the concept of certifying algorithms. A certifying algorithm is any algorithm that, instead of simply returning **yes** or **no** to a feasibility query, provides a proof (certificate) that the returned response is correct [23]. Certifying algorithms for many combinatorial optimization problems have been studied in the literature. This is especially true for certifying algorithms that utilize properties of graphical structures [10, 17, 21].

Certifying algorithms rely on the presence of short certificates, both positive and negative. In case of Horn constraint systems, a satisfying assignment serves as a positive certificate and it is clearly succinct. In this paper, we focus on negative certificates or refutations of Horn constraint systems; in particular, we focus on cutting plane based refutations. Our primary interest is the length of the refutations. This helps identify what restrictions can be placed on cutting plane refutations of HCSs, while still guaranteeing the existence of short refutations. In particular, we focus on the length of both tree-like refutations and Dag-like refutations. We also investigate how the length of refutations changes when different inference rules are used. In this paper, we use two well-known inference rules known as the ADD rule and DIV rule [5] (see Section 2). Additionally we study the complexity of finding read-once refutations of Horn constraint systems when we allow for constraints to be multiplied by bounded coefficients.

The principal contributions of this paper are as follows:

1. Cutting plane tree-like refutations using only the ADD rule do not p-simulate cutting plane tree-like refutations using both the ADD rule and DIV rule for HCSs (see Theorem 9).
2. There exist HCSs for which Tree-like refutations using the ADD and DIV rules must be exponential in the size of the input HCS (see Theorem 10).
3. Dag-like refutations using the ADD and DIV rules are polynomial in the size of the input HCS (see Theorem 11).
4. Finding read-once refutations of HCSs is **NP-hard** even when we allow for constraints to be multiplied by coefficients bounded by a fixed constant (see Theorem 17).

Additionally, we derive interesting corollaries from the above results.

The rest of this paper is organized as follows: In Section 2, we introduce the problems being studied. Section 3 provides motivation for studying this problem and describe the related work in the literature. Our results for tree-like refutations are presented in Section 4. In Section 5, we give our results for Dag-like refutations. We examine read-once refutations with restricted multiplication in Section 6. We conclude in Section 7 by summarizing our contributions, and outlining avenues for future research.

2 Statement of Problems

In this section, we define the problems under consideration.

▶ **Definition 1.** *A system of constraints $\mathbf{A} \cdot \mathbf{x} \geq \mathbf{b}$ is said to be a Horn Constraint system (HCS) or a Horn polyhedron if*

1. *The entries in $\mathbf{A}$ belong to the set $\{0, 1, -1\}$.*
2. *Each row of $\mathbf{A}$ contains at most one positive entry.*
3. *$\mathbf{x}$ is a real valued vector.*
4. *$\mathbf{b}$ is an integral vector.*

In a constraint $\mathbf{a} \cdot \mathbf{x} \geq b_1$, b_1 is called the defining constant and in the constraint system $\mathbf{A} \cdot \mathbf{x} \geq \mathbf{b}$, $\mathbf{b}$ is referred to as the defining constant vector. The assumption that $\mathbf{b}$ is integral is *necessary* to maintain the *soundness* of the proof systems discussed in this paper.

We are interested in certificates of infeasibility; in particular, we are interested in cutting plane based refutations. In linear programs (systems of linear inequalities), we use the following rule:

$$ADD: \quad \frac{\sum_{i=1}^{n} a_i \cdot x_i \geq b_1 \qquad\qquad \sum_{i=1}^{n} a'_i \cdot x_i \geq b_2}{\sum_{i=1}^{n} (a_i + a'_i) \cdot x_i \geq b_1 + b_2} \tag{1}$$

We refer to Rule (1) as the **ADD rule**. This rule plays the same role as resolution in clausal formulas. It is easy to see that Rule (1) is sound in that any assignment satisfying the hypotheses **must** satisfy the consequent. Furthermore, the rule is **complete** in that if the original system is linear infeasible, then repeated application of Rule (1) will result in a contradiction of the form: $0 \geq b$, $b > 0$. The completeness of the ADD rule was established by Farkas [11], in a lemma that is famously known as Farkas' Lemma for systems of linear inequalities [27].

Farkas' lemma along with the fact that linear programs must have basic feasible solutions establishes that the linear programming problem is in the complexity class **NP** $\cap$ **coNP**. Farkas' lemma is one of several lemmata that consider pairs of linear systems in which exactly one element of the pair is feasible. These lemmas are collectively referred to as "Theorems of the Alternative" [25].

▶ **Definition 2.** *A* linear refutation *is a sequence of applications of the ADD rule that results in a contradiction of the form* $0 \geq b$, $b \geq 1$.

In general, applying the ADD rule to an infeasible system $\mathbf{A} \cdot \mathbf{x} \geq \mathbf{b}$, could result in a contradiction of the form $0 \geq b$, $b > 0$. However, in case of Horn systems (with integral defining constants), we must have $b \geq 1$ (see [6]).

When studying integer feasibility, we typically use an additional rule. This is referred to as the **DIV rule** and is described as follows:

$$DIV: \quad \frac{\sum_{i=1}^{n} a_{ij} \cdot x_i \geq b_j \qquad\qquad d \in \mathbb{Z}^+ : \frac{a_{ij}}{d} \in \mathbb{Z},\ i = 1 \ldots n}{\sum_{i=1}^{n} \frac{a_{ij}}{d} \cdot x_i \geq \left\lceil \frac{b_j}{d} \right\rceil} \tag{2}$$

Rule (2) corresponds to dividing a constraint by a common divisor d of the left-hand coefficients and then rounding the right-hand side. Since each $\frac{a_{ij}}{d}$ is an integer this inference preserves integer solutions but does necessarily preserve linear solutions. However, for systems of Horn constraints the DIV rule preserves linear feasibility, since in Horn polyhedra, linear feasibility implies integer feasibility [6].

▶ **Definition 3.** *An* integer refutation *is a sequence of applications of the ADD and DIV rules that results in a contradiction of the form* $0 \geq b$, $b \geq 1$.

Note that for systems of Horn constraints, an integer refutation still proves linear infeasibility.

We now formally define the types of refutations discussed in this paper.

▶ **Definition 4.** *A* ***Dag-like*** *refutation is a refutation in which each constraint, can be used any number of times. This applies to constraints present in the original system and those derived as a result of previous applications of the inference rules.*

▶ **Definition 5.** *A* ***tree-like*** *refutation is a refutation in which each derived constraint, can be used at most once. However, if a derived constraint needs to be reused, then it can be re-derived.*

▶ **Definition 6.** *A* ***read-once*** *refutation is a refutation in which each constraint can be used at most once. This applies to constraints present in the original system and those derived as a result of previous applications of the inference rules.*

For both tree-like and Dag-like refutations, we are interested in the length of the refutation. We measure the length of a refutation in terms of the number of inferences.

▶ **Definition 7.** *The* ***length*** *of a refutation is the number of inferences (either the ADD rule or the DIV rule) in the refutation.*

We use $|S|$ to denote the length of proof S. Using this definition of length, we can now define the concept of p-simulation.

▶ **Definition 8.** *Let S and S' be two proof systems. S* ***p-simulates*** *S' over a set of formulas F, if there exists a polynomial $p(n)$ such that for every formula f in F, there exists a proof S_f of f under proof system S such that $|S_f| \leq p(|S'_f|)$ where S'_f is the shortest proof of f under proof system S'.*

This paper examines both tree-like refutations and Dag-like refutations using only the ADD rule as well as these types of refutations using both the ADD and DIV rules.

3 Motivation and Related Work

Horn constraint systems generalize difference constraint systems. Recall that a difference constraint is a relationship of the form: $x_i - x_j \geq b_{ij}$. A conjunction of difference constraints is called a Difference constraint system (DCS). It is well-known that a DCS is feasible if and only if it has an integral solution (as long as the vector of defining constants is integral). This is because the constraint matrix $\mathbf{A}$ of a DCS is Totally unimodular (TU) [27]. If $\mathbf{A}$ is TU and $\mathbf{b}$ is integral, then all the extreme points of the polyhedron $\mathbf{A} \cdot \mathbf{x} \geq \mathbf{b}$ are integral. Horn constraint matrices are *not TU*; however, it is known that if $\mathbf{A} \cdot \mathbf{x} \geq \mathbf{b}$ is feasible, then it has a minimal element, which is integral [32]. Horn constraint systems have been used as domains in abstract interpretation [2, 9]. Horn systems also find applications in declarative programming [16, 22]. The applications of Horn constraints to program verification has been discussed extensively in [3, 20]. Recently, Horn clauses have been utilized to solve the satisfiability problem for general CNF systems through MAXSAT resolution [4].

This paper is concerned with negative certificates. Assume that we are given a linear constraint system $\mathbf{P}:\ \mathbf{A} \cdot \mathbf{x} \geq \mathbf{b}$ (not necessarily Horn). Any satisfying assignment to the system serves as a positive certificate which asserts the feasibility of $\mathbf{P}$. In order to certify the infeasibility of a linear system, we typically resort to Farkas' lemma [11]. As per Farkas' lemma, it suffices to provide a non-negative m-vector $\mathbf{y}$, such that $\mathbf{y} \cdot \mathbf{A} = \mathbf{0}$, $\mathbf{y} \cdot \mathbf{b} < \mathbf{0}$. This vector $\mathbf{y}$ is called the Farkas witness of the infeasibility of $\mathbf{P}$.

It is important to note that the absence of a Farkas witness guarantees linear feasibility but not integer feasibility. For establishing integer infeasibility, additional inference rules are required. One such inference system is the cutting plane calculus introduced by Gomory [12]. Gomory proposed cutting planes mainly as an algorithmic approach to solve integer programs and was less concerned with proofs and proof lengths. One of the first papers to use cutting planes as a propositional proof system is [8]. The connection between resolution and cutting planes is explored in [14]. In [13], it is shown that there exist tautologies (the pigeonhole principle) for which the number of resolution steps must be exponential in the size of the input. Exponential lower bounds for cutting plane proofs are detailed in [5] and [26]. It is unlikely that succinct certificates of infeasibility exist for integer programming, since this would mean that integer programming lies in the complexity class $\mathbf{NP} \cap \mathbf{coNP}$.

In this paper, we focus on deriving bounds on the lengths of cutting plane proofs in restricted cutting plane systems. It is to be noted that placing restrictions on the type or number of inferences that can be applied could cause the proof system to become **incomplete**. There are several reasons to consider restricted proof systems, viz.

1. Restricted proofs tend to be "short" (polynomial in the size of the input). For instance, read-once refutations are at most linear in the size of the input. Read-once and literal-once refutations have been discussed extensively for boolean formulas in [15] and [31]. This is in stark contrast to general resolution. In general resolution a refutation could have exponentially many steps [13].
2. In certain cases, the presence of restricted proofs can be determined in polynomial time. For instance, we have shown that the problem of read-once refutations is in **P** for difference constraints [29] and Unit Two Variable Per Inequality (UTPVI) constraints [30].

In recent work, we showed that the problem of finding read-once refutations under the ADD and DIV rules in Horn constraint systems is **NP-hard** [19]. In this paper, we extend these results by studying the lengths of tree-like and Dag-like refutations under the ADD and DIV rules. We also examine read-once refutations when we allow for limited use of multiplication.

4 Tree-like Refutations

In this section, we examine the length of tree-like refutations for HCSs.

First, we compare tree-like proofs using only the ADD rule to tree-like proofs using both the ADD and DIV rules.

▶ **Theorem 9.** *Tree-like proofs using only the ADD rule do not p-simulate tree-like proofs using both the ADD rule and DIV rule for HCSs.*

Proof. Consider the following HCS:

$$\left.\begin{array}{rcl} -x_1 - x_2 - x_3 - \ldots - x_n & \geq & 1 \\ x_1 - x_2 - x_3 - \ldots - x_n & \geq & 0 \\ x_2 - x_3 - \ldots - x_n & \geq & 0 \end{array}\right| \begin{array}{rcl} x_3 - \ldots - x_n & \geq & 0 \\ & \vdots & \\ x_n & \geq & 0 \end{array} \qquad (3)$$

We will show that any tree-like refutation of System 3 that uses only the ADD rule has at least $(2^n - 1)$ inferences. This will be done by induction on the number of variables. Let $\mathbf{H}_n$ be System 3.

If $n = 1$, then $\mathbf{H}_1$ consists of the constraints $-x_1 \geq 1$ and $x_1 \geq 0$. This system has the following refutations using only the ADD rule:

1. Apply the ADD rule to $-x_1 \geq 1$ and $x_1 \geq 0$ to get $0 \geq 1$.

This is a contradiction. Thus, System $\mathbf{H}_1$ has a refutation with $1 = 2^1 - 1$ inference. Note that this is the shortest refutation of System $\mathbf{H}_1$, thus any refutation of this system must use at least 1 inference as desired.

Now assume that when $n = k$ any refutation System $\mathbf{H}_k$ uses at least $(2^k - 1)$ inferences. Let us look at System $\mathbf{H}_{k+1}$. Without the constraint $x_{k+1} \geq 0$, System $\mathbf{H}_{k+1}$ can be satisfied by setting $\mathbf{x} = (0, 0, 0, \ldots, 0, -1)$. Thus, any refutation of System $\mathbf{H}_{k+1}$ must use $x_{k+1} \geq 0$.

Let R_{k+1} be a refutation of $\mathbf{H}_{k+1}$. Since the addition of constraints is associative we can assume without loss of generality that R_{k+1} consists of the following:

1. A derivation of a constraint of the form $-c \cdot x_{k+1} \geq 1$ for some constant c from System $\mathbf{H}_{k+1} \setminus \{x_{k+1} \geq 0\}$.
2. An additional c applications of the ADD rule (using the constraint $x_{k+1} \geq 0$) to derive the the constraint $0 \geq 1$.

Note that System $\mathbf{H}_{k+1} \setminus \{x_{k+1} \geq 0\}$ can be constructed from System $\mathbf{H}_k$ by adding $-x_{k+1}$ to every constraint. This means that any tree-like derivation of $-c \cdot x_{k+1} \geq 1$ for some constant c from System $\mathbf{H}_{k+1} \setminus \{x_{k+1} \geq 0\}$ corresponds to a refutation of System $\mathbf{H}_k$.

Thus, by the inductive hypothesis, this derivation must use at least $(2^k - 1)$ inferences. Since every constraint in $\mathbf{H}_{k+1} \setminus \{x_{k+1} \geq 0\}$ has a $-x_{k+1}$ term, we must have that $c \geq 2^k$ in the resultant constraint.

Thus, to derive the constraint $0 \geq 1$ an additional $c \geq 2^k$ inferences of the form "ADD $-c \cdot x_{k+1} \geq 1$ and $x_{k+1} \geq 0$ to get $-(c-1) \cdot x_{k+1} \geq 1$" are needed. Since we desire a tree-like refutation, we cannot shorten this refutation by reusing already derived constraints. Thus, any refutation of System $\mathbf{H}_{k+1}$ needs to use a total of $2^k + 2^k - 1 = 2^{k+1} - 1$ inferences.

However, System 3 has the following tree-like refutation using both the ADD and DIV rules:

1. Apply the ADD rule to $-x_1 - x_2 - x_3 - \ldots - x_n \geq 1$ and $x_1 - x_2 - x_3 - \ldots - x_n \geq 0$ to get
$-2 \cdot x_2 - 2 \cdot x_3 - \ldots - 2 \cdot x_n \geq 1$.
2. Apply the DIV rule with $d = 2$ to $-2 \cdot x_2 - 2 \cdot x_3 - \ldots - 2 \cdot x_n \geq 1$ to get $-x_2 - x_3 - \ldots - x_n \geq 1$.
3. Apply the ADD rule to $-x_2 - x_3 - \ldots - x_n \geq 1$ and $x_2 - x_3 - \ldots - x_n \geq 0$ to get $-2 \cdot x_3 - \ldots - 2 \cdot x_n \geq 1$.
4. Apply the DIV rule with $d = 2$ to $-2 \cdot x_3 - \ldots - 2 \cdot x_n \geq 1$ to get $-x_3 - \ldots - x_n \geq 1$.
5. $\vdots$
6. Apply the ADD rule to $-x_n \geq 1$ and $x_n \geq 0$ to get $0 \geq 1$.

This refutation has only $(2 \cdot n - 1)$ inferences. Thus, the tree-like refutation of System 3 that uses only the ADD rule is exponentially longer than the tree-like refutation using both the ADD and DIV rules. ◀

Despite the fact that the DIV rule can result in much shorter tree-like refutations for systems of Horn constraints, there are still systems of Horn constraints with exponentially long refutations.

▶ **Theorem 10.** *For every positive integer n, there exists an HCS with n variables for which every tree-like cutting plane proof using both the ADD and DIV rules is exponential in the size of the system.*

Proof. We show this by introducing the variable x_0 to System (3). Specifically, we examine the following system of Horn constraints.

$$
\begin{array}{rcl|rcl}
x_0 - x_1 - x_2 - x_3 - \ldots - x_n & \geq & 1 & x_3 - \ldots - x_n & \geq & 0 \\
x_1 - x_2 - x_3 - \ldots - x_n & \geq & 0 & & \vdots & \\
x_2 - x_3 - \ldots - x_n & \geq & 0 & x_n & \geq & 0 \\
 & & & -x_0 - x_i & \geq & 0
\end{array}
\tag{4}
$$

We will examine how the minimum length of a tree-like proof using both the ADD rule and DIV rule depends on the choice of x_i in the constraint $-x_0 - x_i \geq 0$.

By construction, the constraint $x_0 - x_1 - x_2 - x_3 - \ldots - x_n \geq 1$ is the only constraint in System (4) with the term x_0. Thus, any constraint derived from $x_0 - x_1 - x_2 - x_3 - \ldots - x_n \geq 1$ must have the term x_0 until x_0 is canceled from the constraint. This means that, the DIV rule cannot be applied (with $d > 1$) to any constraint derived from $x_0 - x_1 - x_2 - x_3 - \ldots - x_n \geq 1$ until x_0 is eliminated. There are two ways to eliminate x_0 from this constraint:

Type 1. Derive the constraint $-x_0 \geq 0$ from the remaining constraints and then eliminate x_0. In this case, the only constraint with $-x_0$ is the constraint $-x_0 - x_i \geq 0$. Thus, we must derive the constraint $x_i \geq 0$. To do this we need to eliminate $x_{i+1} \ldots x_n$ from $x_i - x_{i+1} - \ldots - x_n \geq 0$ using the constraints where those variables have positive coefficients.

Type 2. Eliminate it by using the constraint $-x_0 - x_i \geq 0$ and then eliminate the extra copy of x_i. Note that in this case the DIV rule cannot be applied until this extra copy is eliminated. To do this without deriving the constraint $x_i \geq 0$ (since this would make it the equivalent of a Type 1 refutation), we need to cancel the variables $x_1 \dots x_{i-1}$.

We will refer to these as Type 1 and Type 2 tree-like refutations. We will show that any tree-like refutation of system (4) requires $\min(2 \cdot n + 2^{n-i}, 2 \cdot (n-i) + 2^i + 1)$ inferences.

First consider the case where $x_i = x_n$ in System (4). This results in the following HCS.

$$\begin{array}{rcl|rcl} x_0 - x_1 - x_2 - x_3 - \ldots - x_n & \geq & 1 & x_3 - \ldots - x_n & \geq & 0 \\ x_1 - x_2 - x_3 - \ldots - x_n & \geq & 0 & & \vdots & \\ x_2 - x_3 - \ldots - x_n & \geq & 0 & x_n & \geq & 0 \\ & & & -x_0 - x_n & \geq & 0 \end{array}$$

In this case, System (4) has the following Type 1 tree-like refutation.

1. Apply the ADD rule to $x_n \geq 0$ and $-x_0 - x_n \geq 0$ to get $-x_0 \geq 0$.
2. Apply the ADD rule to $-x_0 \geq 0$ and $x_0 - x_1 - x_2 - x_3 - \ldots - x_n \geq 1$ to get $-x_1 - x_2 - x_3 - \ldots - x_n \geq 1$.

We have now derived System (3) with n variables. From Theorem 9, System (3) has a tree-like refutation of length $(2 \cdot n - 1)$. Thus, if $x_i = x_n$, System (4) has a refutation of length $(2 \cdot n + 1)$ as desired.

However, consider the case where $x_i = x_1$ in System 4. This results in the following HCS.

$$\begin{array}{rcl|rcl} x_0 - x_1 - x_2 - x_3 - \ldots - x_n & \geq & 1 & x_3 - \ldots - x_n & \geq & 0 \\ x_1 - x_2 - x_3 - \ldots - x_n & \geq & 0 & & \vdots & \\ x_2 - x_3 - \ldots - x_n & \geq & 0 & x_n & \geq & 0 \\ & & & -x_0 - x_1 & \geq & 0 \end{array}$$

In this case, System (4) has the following Type 2 tree-like refutation.

1. Apply the ADD rule to $-x_0 - x_1 \geq 0$ and $x_0 - x_1 - x_2 - x_3 \ldots x_n \geq 1$ to get $-2 \cdot x_1 - x_2 - x_3 \ldots - x_n \geq 1$.
2. Apply the ADD rule to $x_1 - x_2 - x_3 - \ldots - x_n \geq 0$ and $-2 \cdot x_1 - x_2 - x_3 \ldots - x_n \geq 1$ to get $-x_1 - 2 \cdot x_2 - 2 \cdot x_3 - \ldots - 2 \cdot x_n \geq 1$.
3. Apply the ADD rule to $x_1 - x_2 - x_3 - \ldots - x_n \geq 0$ and $-x_1 - 2 \cdot x_2 - 2 \cdot x_3 \ldots - 2 \cdot x_n \geq 1$ to get $-3 \cdot x_2 - 3 \cdot x_3 - \ldots - 3 \cdot x_n \geq 1$.
4. Apply the DIV rule with $d = 3$ to $-3 \cdot x_2 - 3 \cdot x_3 - \ldots - 3 \cdot x_n \geq 1$ to get $-x_2 - x_3 - \ldots - x_n \geq 1$.

This results in System (3) with $(n-1)$ variables. This means that completing the refutation will take an additional $(2 \cdot n - 3)$ inferences. Thus, in this case, System (4) has a refutation of length $(2 \cdot n + 1)$ as desired.

Now we consider System (4). As stated previously, any tree-like refutation must be Type 1 or Type 2. Thus, we need to calculate the minimum number of inferences used by each type of refutation.

First we find the minimum length of a Type 1 tree-like refutation. In the general case for System (4), this refutation has the following form:

1. Apply the ADD rule to

   $$x_i - x_{i+1} - \ldots - x_n \geq 0 \text{ and } x_{i+1} - x_{i+2} \ldots - x_n \geq 0$$

   to get

   $$x_i - 2 \cdot x_{i+2} - \ldots - 2 \cdot x_n \geq 0.$$

2. Apply the ADD rule to $x_i - 2 \cdot x_{i+2} - \ldots - 2 \cdot x_n \geq 0$ and 2 copies of $x_{i+2} - x_{i+3} \ldots - x_n \geq 0$ to get
$$x_i - 4 \cdot x_{i+3} - \ldots - 4 \cdot x_n \geq 0.$$
3. For each $r = 3 \ldots (n - i - 1)$, apply the ADD rule to
$$x_i - 2^{r-1} \cdot x_{i+r} - \ldots - 2^{r-1} \cdot x_n \geq 0$$
and 2^{r-1} copies of $x_{+r} - x_{i+r+1} \ldots - x_n \geq 0$ to get
$$x_i - 2^r \cdot x_{i+r+1} - \ldots - 2^r \cdot x_n \geq 0.$$
4. Apply the ADD rule to $x_i - 2^{n-i-1} \cdot x_n$ and 2^{n-i-1} copies of $x_n \geq 0$ to get $x_i \geq 0$.
5. Apply the ADD rule to $x_i \geq 0$ and $-x_0 - x_i \geq 0$ to get $-x_0 \geq 0$.
6. Apply the ADD rule to $-x_0 \geq 0$ and $x_0 - x_1 - x_2 - x_3 - \ldots - x_n \geq 1$ to get
$$-x_1 - x_2 - x_3 - \ldots - x_n \geq 1.$$

This takes a minimum of $(2^{n-i} + 1)$ inferences. We have now derived System (3) with n variables. From Theorem 9, System (3) has a tree-like refutation of length $(2 \cdot n - 1)$. Thus, the minimum length of a Type 1 tree-like refutation of System (4) is $(2 \cdot n + 2^{n-i})$ as desired.

Now we find the minimum length of a Type 2 tree-like refutation. In the general case for System (4), this refutation has the following form:

1. Apply the ADD rule to
$$-x_0 - x_i \geq 0 \text{ and } x_0 - x_1 - x_2 - x_3 \ldots - x_n \geq 1$$
to get
$$-x_1 - x_2 - \ldots - 2 \cdot x_i - \ldots - x_n \geq 1$$
2. Apply the ADD rule to
$$x_1 - x_2 - x_3 - \ldots - x_n \geq 0 \text{ and } -x_1 - x_2 - \ldots - 2 \cdot x_i - \ldots - x_n \geq 1$$
to get
$$-2 \cdot x_2 - 2 \cdot x_3 - \ldots - 3 \cdot x_i - \ldots - 2 \cdot x_n \geq 1.$$
3. Apply the ADD rule to $-2 \cdot x_2 - 2 \cdot x_3 - \ldots - 3 \cdot x_i - \ldots - 2 \cdot x_n \geq 1$ and 2 copies of $x_2 - x_3 - \ldots - x_n \geq 0$ to get
$$-4 \cdot x_3 - 4 \cdot x_4 - \ldots - 5 \cdot x_i - \ldots - 4 \cdot x_n \geq 1.$$
4. For each $r = 3 \ldots i - 1$, apply the ADD rule to
$$-2^{r-1} \cdot x_r - 2^{r-1} \cdot x_{r+1} - \ldots - (2^{r-1} + 1) \cdot x_i - \ldots - 2^{r-1} \cdot x_n \geq 1$$
and 2^{r-1} copies of $x_r - x_{r+1} - \ldots - x_n \geq 0$ to get
$$-2^r \cdot x_{r+1} - 2^r \cdot x_{r+2} - \ldots - (2^r + 1) \cdot x_i - \ldots - 2^r \cdot x_n \geq 1.$$
5. Apply the ADD rule to $-(2^{i-1} + 1) \cdot x_i - 2^{i-1} \cdot x_{i+1} - \ldots - 2^{i-1} \cdot x_n \geq 1$ and $(2^{i-1} + 1)$ copies of $x_i - x_{i+1} \ldots - x_n \geq 0$ to get
$$-(2^i + 1) \cdot x_{i+1} - (2^i + 1) \cdot x_{i+2} - \ldots - (2^i + 1) \cdot x_n \geq 1.$$

6. Apply the DIV rule with $d = (2^i + 1)$ to

$$-(2^i + 1) \cdot x_{i+1} - (2^i + 1) \cdot x_{i+2} - \ldots - (2^i + 1) \cdot x_n \geq 1$$

to get

$$-x_{i+1} - x_{i+2} - \ldots - x_n \geq 1.$$

This takes a minimum of $(2^i + 2)$ inferences. We have now derived System (3) with $(n - i)$ variables. This means that completing the refutation will take an additional $(2 \cdot n - i - 1)$ inferences. Thus, the minimum length of a Type 2 tree-like refutation of System (4) is $(2 \cdot (n - i) + 2^i + 1)$ as desired.

Thus, when $x_i = x_{\frac{n}{2}}$ any tree-like refutation of System 4 must have a length of at least $(n + 2^{\frac{n}{2}} + 1)$.

$$\begin{array}{rcl|rcl}
x_0 - x_1 - x_2 - x_3 - \ldots - x_n & \geq & 1 & x_3 - \ldots - x_n & \geq & 0 \\
x_1 - x_2 - x_3 - \ldots - x_n & \geq & 0 & \vdots & & \\
x_2 - x_3 - \ldots - x_n & \geq & 0 & x_n & \geq & 0 \\
 & & & -x_0 - x_{\frac{n}{2}} & \geq & 0
\end{array}$$

Thus, any tree-like refutation of this system that uses both the ADD rule and the DIV rule must be exponential in the size of the system. ◀

5 Dag-like Refutations

In this section, we examine the lengths of Dag-like refutations of systems of Horn constraints.

Unlike tree-like refutations, Dag-like refutations of HCSs are guaranteed to be polynomially sized.

▶ **Theorem 11.** *Dag-like cutting plane proofs of Horn constraint systems are polynomial in the size of the constraint system even when restricted to using only the ADD rule.*

Proof. Let $\mathbf{H}$ be an unsatisfiable system of Horn constraints with m constraints over n variables. If $\mathbf{H}$ has no positive absolute constraints (constraints of the form $x_i \geq c$), then let $\mathbf{D} \subseteq \mathbf{H}$ be the set of difference constraints in $\mathbf{H}$. If $\mathbf{x}$ is a assignment to the variables in $\mathbf{H}$ that satisfies every constraint in $\mathbf{D}$, then for some positive constant M, $(\mathbf{x} - M \cdot \mathbf{1})$ is a satisfying assignment to $\mathbf{H}$. Thus, $\mathbf{D}$ must be unsatisfiable. It is easy in this case $\mathbf{H}$ has a refutation of length at most m since $\mathbf{D}$ is an infeasible system of difference constraints.

If $\mathbf{H}$ has a positive absolute constraint $x_i \geq c$, then we can construct the system $\mathbf{H}'$ by removing the constraint $x_i \geq c$ and summing it with every constraint with the literal $-x_i$. This process is repeated, until a feasible system is derived or an infeasible system of difference constraints is constructed.

Note that if an infeasible system of difference constraints is constructed, then the system has a read-once refutation using only the ADD rule. This is a linearly sized proof of infeasibility. To obtain this system, we eliminated at most $(n - 1)$ constraints of the form $x_i \geq c$. Each of these eliminations took at most m applications of the ADD rule. Since the refutation of the resultant DCS is read-once, it cannot use more than m inferences. Thus, any refutation discovered by this process has length at most $m \cdot n$. ◀

Note that the refutation generated this way has the following properties:

1. Every intermediate constraint is a Horn constraint.
2. Only the ADD rule is used.

Thus, we have the following corollaries:

▶ **Corollary 12.** *Every infeasible system of Horn constraints has a refutation where every intermediate constraint is Horn.*

Proof. This follows immediately from the fact that in the refutation generated by Theorem 11, every intermediate constraint is Horn. ◀

▶ **Corollary 13.** *If a Dag-like refutation of length n exists for an unsatisfiable Horn constraint system H, then there exists a polynomial p such that there exists Dag-like refutation of length $p(n)$, even when all intermediate constraints are Horn.*

Proof. Let D be a Dag-like refutation of H of length n. Let $H_D \subseteq H$ be the set of constraints in H used by D. Thus, H_D is an infeasible HCS. By Theorem 11, H_D has a polynomially sized Dag-like refutation D' of length n' where every intermediate constraint is Horn. Thus, there exists a polynomial p such that $n' \leq p(|H_D|) \leq p(n)$ since $|H_D| \leq n$. ◀

▶ **Corollary 14.** *Dag-like refutations using only the ADD rule p-simulate Dag-like refutations using both the ADD rule and DIV rule for HCSs.*

Proof. Let H be an arbitrary HCS. Let D be the shortest Dag-like refutation of H that uses both the ADD rule and the DIV rule, and let n be the length of D. Let $H_D \subseteq H$ be the set of constraints in H used by D. Thus, H_D is an infeasible HCS. By Theorem 11, H_D has a polynomially sized Dag-like refutation D' of length n' using only the ADD rule. Thus, there exists a polynomial p such that $n' \leq p(|H_D|) \leq p(n)$ since $|H_D| \leq n$. ◀

6 Restricted Read-once Refutations

In this section, we examine the complexity of finding read-once refutations of HCSs when we allow for the multiplication of constraints by bounded coefficients. To accomplish this, we introduce an inference rule known as the **MUL rule**. This rule is described as follows:

$$MUL: \quad \frac{\sum_{i=1}^{n} a_{ij} \cdot x_i \geq b_j \qquad\qquad c_j \in \mathbb{Z}^+}{\sum_{i=1}^{n} c_j \cdot a_{ij} \cdot x_i \geq c_j \cdot b_j} \tag{5}$$

Rule (5) corresponds to multiplying a constraint by a positive integer multiplier. Note that, with unrestricted use of the ADD and MUL rules, any infeasible system of Horn constraints has a read-once refutation. This refutation is constructed as follows:

1. Let $\mathbf{H}$ be an infeasible HCS, and let T be a tree-like refutation of H that uses only the ADD rule.
2. For each constraint $l_j \in \mathbf{H}$, let c_j be the number of times l_j is used in T. If $c_j > 0$, apply the MUL rule, with multiplier c_j to l_j.
3. Use the ADD rule to sum together all the constraints generated with the MUL rule.

Thus, we examine the problem of finding read-once refutations in HCSs when we only allow for restricted use of the MUL rule.

▶ **Definition 15.** *An r-restricted read-once refutation using the ADD and MUL rules of an HCS $\mathbf{H}$, is a read-once refutation of $\mathbf{H}$ such that:*

1. *The MUL rule is only applied with with multiplier $c \leq r$.*
2. *The MUL rule is only applied to constraints in $\mathbf{H}$.*

First we show that, for any fixed constant r, the problem of finding an r-restricted read-once refutation using the ADD and MUL rules is **NP-hard**. To accomplish this, we utilize a reduction from the set packing problem.

▶ **Definition 16.** *The **set packing** problem is the following: Given a set S, m subsets $S_1, \ldots, S_m$ of S, and an integer k, does $\{S_1, \ldots, S_m\}$ contain k mutually disjoint sets.*

This problem is known to be **NP-complete** [18].

▶ **Theorem 17.** *Let r be any positive integer. Finding r-restricted read-once refutations using the ADD and MUL rules is **NP-hard** for HCSs.*

Proof. Consider an instance of the set packing problem and let h be the integer such that $2^h \leq r < 2^{h+1}$. We construct the system of Horn constraints $\mathbf{H}$ as follows:

1. For each $x_i \in S$:
   a. Create the variables x_i and w_i.
   b. For each $g = 1 \ldots h$, create the variables $y_{i,(2\cdot g-1)}$ and $y_{i,(2\cdot g)}$. Also create the constraints $y_{i,(2\cdot g-1)} - y_{i,(2\cdot g+1)} - y_{i,(2\cdot g+2)} \geq 0$ and $y_{i,(2\cdot g)} - y_{i,(2\cdot g+1)} - y_{i,(2\cdot g+2)} \geq 0$.
   c. Create the constraints $x_i - y_{i,1} - y_{i,2} \geq 0$, $y_{i,(2\cdot h-1)} - w_i \geq 0$, $y_{i,(2\cdot h)} - w_i \geq 0$, and $w_i \geq 0$.
2. For $j = 1 \ldots k$, create the variable v_j.
3. For each subset S_l, $l = 1 \ldots m$, and each $j = 1 \ldots k$ create the constraints $v_j - \sum_{x_i \in S_l} x_i \geq 0$.
4. Finally create the constraint $-v_1 - \ldots - v_k \geq 1$.

We will show that $\mathbf{H}$ has an r-restricted read-once refutation using the ADD and MUL rules if and only if the sets S_1, …, S_m have a packing of size k. First we assume that $\mathbf{H}$ has a r-restricted read-once refutation R using the ADD and MUL rules.

Consider the constraint $x_i - y_{i,1} - y_{i,2} \geq 0$. Assume that R applies the MUL rule to this constraint with multiplier c_i. This results in the constraint $c_i \cdot x_i - c_i \cdot y_{i,1} - c_i \cdot y_{i,2} \geq 0$ To eliminate the variables $y_{i,1}$ and $y_{i,2}$, R must also do the following:

1. Apply the MUL rule with multiplier c_i to each the constraints $y_{i,1} - y_{i,3} - y_{i,4} \geq 0$ and $y_{i,2} - y_{i,3} - y_{i,4} \geq 0$. Then apply the ADD rule to generate the constraint $c_i \cdot x_i - 2 \cdot c_i \cdot y_{i,3} - 2 \cdot c_i \cdot y_{i,4} \geq 0$.
2. Apply the MUL rule with multiplier $2 \cdot c_i$ to each the constraints $y_{i,3} - y_{i,5} - y_{i,6} \geq 0$ and $y_{i,4} - y_{i,5} - y_{i,6} \geq 0$. Then apply the ADD rule to generate the constraint $c_i \cdot x_i - 4 \cdot c_i \cdot y_{i,5} - 4 \cdot c_i \cdot y_{i,6} \geq 0$.
3. Apply the MUL rule with multiplier $4 \cdot c_i$ to each the constraints $y_{i,5} - y_{i,7} - y_{i,8} \geq 0$ and $y_{i,6} - y_{i,7} - y_{i,8} \geq 0$. Then apply the ADD rule to generate the constraint $c_i \cdot x_i - 8 \cdot c_i \cdot y_{i,7} - 8 \cdot c_i \cdot y_{i,8} \geq 0$.
4. Apply the MUL rule with multiplier $2^{h-1} \cdot c_i$ to each the constraints $y_{i,(2\cdot h-1)} - w_i \geq 0$ and $y_{i,(2\cdot h)} - w_i \geq 0$. Then apply the ADD rule to generate the constraint $c_i \cdot x_i - 2^h \cdot c_i \cdot w_i \geq 0$.
5. Apply the MUL rule with multiplier $2^h \cdot c_i$ to the constraint $w_i \geq 0$. Then apply the ADD rule to generate the constraint $c_i \cdot x_i \geq 0$.

Since R is an r-restricted read-once refutation, we have that $2^h \cdot c_i \leq r < 2 \cdot 2^h$. Thus, $c_i = 1$.

By construction, $x_i - y_{i,1} - y_{i,2} \geq 0$ is the only constraint in $\mathbf{H}$ where x_i has positive coefficient. Thus, for each x_i, R can only use one constraint where x_i has negative coefficient. Otherwise, R will be unable to cancel every instance of $-x_i$.

Note that, by construction, the only constraint in $\mathbf{H}$ with positive defining constant is $-v_1 - \ldots - v_k \geq 1$. Thus, this constraint must by used in R. For each v_i, R must use a constraint corresponding to one of the sets $S_1, \ldots, S_m$. Recall, that x_i can be used by at most one of these constraints, thus the sets chosen for each v_j must be mutually disjoint and no set can be chosen multiple times. This can only happen if the sets $S_1, \ldots, S_m$ have a packing of size k.

Now assume that the sets $S_1, \ldots, S_m$ have a packing of size k. Assume without loss of generality that this packing is the sets $S_1, \ldots, S_k$. H has the following r-restricted read-once refutation using the ADD and MUL rules.

1. Start with the constraint $-v_1 - \ldots - v_k \geq 1$.
2. For each v_j, apply the ADD rule to the constraint $v_j - \sum_{x_i \in S_j} x_i \geq 0$ and the result of the previous application of the ADD rule.
3. Let L be the constraint derived through this process. Since the sets $S_1, \ldots, S_k$ are mutually disjoint, every variable x_i in L has coefficient -1.
4. For each x_i in L, derive the constraint $x_i \geq 0$, using the method detailed previously. Then use the constraint $x_i \geq 0$ to eliminate $-x_i$ from L. ◀

Note that the construction used in Theorem 17 assumes that r is a fixed constant. Let n be the number of variables in the HCS $\mathbf{H}$ constructed in the proof of Theorem 17. By construction, we have that $n = k + (2 \cdot \log_2 r + 2) \cdot |S|$. We can assume without loss of generality that $k \leq |S|$ (we cannot pack more than k non-empty subsets of S into S). Thus, $n \leq |S| \cdot (2 \cdot \log_2 r + 3)$. Let c be an arbitrary positive integer. If we choose $r = 2^{\frac{|S|^{c-1}-3}{2}}$, then $|S|^{c-1} = 2 \cdot (\log_2 r) + 3$ and $n^{c-1} \leq (2 \cdot (\log_2 r) + 3)^c$. This means that $\log_2 r$ is in $O(n^{1-\frac{1}{c}})$. This leads to the following corollary of Theorem 17.

▶ **Corollary 18.** *Let c be an arbitrary positive integer. Finding r-restricted read-once refutations using the ADD and MUL rules is* ***NP-hard*** *for HCSs, even when r is $O\left(2^{(n^{1-\frac{1}{c}})}\right)$.*

7 Conclusion

In this paper, we studied refutability in Horn constraint systems under various cutting plane based proof systems. Horn constraint systems generalize difference constraint systems and find applications in a number of domains, especially program verification. We looked at both tree-like and Dag-like refutations. Both these proof systems are complete for Horn constraint systems (assuming that the defining constants are integers). For both types of refutations we looked at refutations using only the ADD rule as well as refutations using both the ADD and DIV rules. We showed there exist HCSs with exponentially sized tree-like refutations even when both the ADD and DIV rules are allowed. We also showed that every HCS has a polynomially sized Dag-like refutation even when restricted to only the ADD rule. It follows that cutting plane Dag-like refutations using only the ADD rule p-simulate cutting plane Dag-like refutations using the ADD rule and DIV rule for HCSs. Additionally, we established that if a cutting plane refutation of length n exists for an unsatisfiable Horn constraint system, then there exists a cutting plane proof of length $p(n)$, even when all intermediate constraints are Horn. We also showed that, when we allow for constraints to be multiplied by bounded coefficients, the problem of finding a read-once refutation of an HCS is **NP-complete**. Our results are important because they are the first step towards the design of efficient procedures in constraint solvers [7, 24, 28].

From our perspective, the following problems are worth pursuing:

1. Can we find the shortest tree-like or Dag-like proofs of an unsatisfiable Horn constraint system, if only the ADD rule is permitted?
2. Can we find the shortest tree-like or Dag-like proofs of an unsatisfiable Horn constraint system, if both the ADD and the DIV rules are permitted?

It is worth noting that in the case of Horn clauses the problem of finding the shortest resolution proof is **NP-hard** [1]. Even worse, the problem is hard to linearly approximate [1]. However, these negative results do not directly apply to tree-like (or Dag-like) proofs in Horn constraint systems.

References

1 M. Alekhnovich, S. Buss, S. Moran, and T. Pitassi. Minimum Propositional Proof Length is NP-Hard to Linearly Approximate. In *Mathematical Foundations of Computer Science (MFCS)*, pages 176–184. Springer-Verlag, 1998. Lecture Notes in Computer Science.

2 Alexey Bakhirkin and David Monniaux. Combining Forward and Backward Abstract Interpretation of Horn Clauses. In *Static Analysis - 24th International Symposium, SAS 2017, New York, NY, USA, August 30 - September 1, 2017, Proceedings*, pages 23–45, 2017.

3 Nikolaj Bjørner, Arie Gurfinkel, Kenneth L. McMillan, and Andrey Rybalchenko. Horn Clause Solvers for Program Verification. In *Fields of Logic and Computation II - Essays Dedicated to Yuri Gurevich on the Occasion of His 75th Birthday*, pages 24–51, 2015.

4 Maria Luisa Bonet, Sam Buss, Alexey Ignatiev, Joao Marques-Silva, and Antonio Morgado. MaxSAT Resolution With the Dual Rail Encoding. In *AAAI Conference on Artificial Intelligence*, 2018. URL: `https://www.aaai.org/ocs/index.php/AAAI/AAAI18/paper/view/16782/16235`.

5 Maria Luisa Bonet, Toniann Pitassi, and Ran Raz. Lower Bounds for Cutting Planes Proofs with Small Coefficients. *J. Symb. Log.*, 62(3):708–728, 1997.

6 R. Chandrasekaran and K. Subramani. A combinatorial algorithm for Horn programs. *Discrete Optimization*, 10:85–101, 2013.

7 Alessandro Cimatti, Alberto Griggio, and Roberto Sebastiani. Computing Small Unsatisfiable Cores in Satisfiability Modulo Theories. *J. Artif. Intell. Res. (JAIR)*, 40:701–728, 2011.

8 W. Cook, C. R. Coullard, and Gy. Turan. On the complexity of Cutting-Plane Proofs. *Discrete Applied Mathematics*, 18:25–38, 1987.

9 Patrick Cousot and Radhia Cousot. Abstract Interpretation: A Unified Lattice Model for Static Analysis of Programs by Construction or Approximation of Fixpoints. In *POPL*, pages 238–252, 1977.

10 Marcel Dhiflaoui, Stefan Funke, Carsten Kwappik, Kurt Mehlhorn, Michael Seel, Elmar Schömer, Ralph Schulte, and Dennis Weber. Certifying and repairing solutions to large LPs how good are LP-solvers? In *SODA*, pages 255–256, 2003.

11 Gyula Farkas. Über die Theorie der Einfachen Ungleichungen. *Journal für die Reine und Angewandte Mathematik*, 124(124):1–27, 1902.

12 R. E. Gomory. Outline of an algorithm for integer solutions to linear programs. *Bulletin of the American Mathematical Society*, 64:275–278, 1958.

13 A. Haken. The intractability of resolution. *Theoretical Computer Science*, 39(2-3):297–308, August 1985.

14 John N. Hooker. Generalized Resolution and Cutting Planes. *Annals of Operations Research*, 12(1-4):217–239, 1988.

15 K. Iwama and E. Miyano. Intractability of Read-Once Resolution. In *Proceedings of the 10th Annual Conference on Structure in Complexity Theory (SCTC '95)*, pages 29–36, Los Alamitos, CA, USA, June 1995. IEEE Computer Society Press.

16 Joxan Jaffar and Michael Maher. Constraint Logic Programming: A Survey. *The Journal of Logic Programming*, s 19–20:503–581, October 1994. doi:10.1016/0743-1066(94)90033-7.

17 Haim Kaplan and Yahav Nussbaum. Certifying algorithms for recognizing proper circular-arc graphs and unit circular-arc graphs. *Discrete Applied Mathematics*, 157(15):3216–3230, 2009.

18 Richard M. Karp. Reducibility among combinatorial problems. In R. E. Miller and J. W. Thatcher, editors, *Complexity of Computer Computations*, pages 85–103, New York, 1972. Plenum Press.

19 Hans Kleine Büning, Piotr J. Wojciechowski, and K. Subramani. On the application of restricted cutting plane systems to Horn constraint systems. In *The 12th International Symposium on Frontiers of Combining Systems, London, United Kingdom,, September 4-6, 2019, Proceedings*, pages 149–164, 2019.

20 Anvesh Komuravelli, Nikolaj Bjørner, Arie Gurfinkel, and Kenneth L. McMillan. Compositional Verification of Procedural Programs using Horn Clauses over Integers and Arrays. In *Formal Methods in Computer-Aided Design, FMCAD 2015, Austin, Texas, USA, September 27-30, 2015.*, pages 89–96, 2015.

21 Dieter Kratsch, Ross M. McConnell, Kurt Mehlhorn, and Jeremy Spinrad. Certifying algorithms for recognizing interval graphs and permutation graphs. In *SODA*, pages 158–167, 2003.

22 Kung-Kiu Lau and Mario Ornaghi. Specifying Compositional Units for Correct Program Development in Computational Logic. In *Program Development in Computational Logic: A Decade of Research Advances in Logic-Based Program Development*, pages 1–29. Springer, 2004.

23 R. M. McConnell, K. Mehlhorn, S. Näher, and P. Schweitzer. Certifying algorithms. *Computer Science Review*, 5(2):119–161, 2011.

24 Microsoft Research. *Z3: An efficient SMT solver.* URL: http://research.microsoft.com/projects/z3/.

25 G. L. Nemhauser and L. A. Wolsey. *Integer and Combinatorial Optimization.* John Wiley & Sons, New York, 1999.

26 Pavel Pudlák. Lower Bounds for Resolution and Cutting Plane Proofs and Monotone Computations. *J. Symb. Log.*, 62(3):981–998, 1997. doi:10.2307/2275583.

27 A. Schrijver. *Theory of Linear and Integer Programming.* John Wiley and Sons, New York, 1987.

28 SRI International. *Yices: An SMT solver.* URL: http://yices.csl.sri.com/.

29 K. Subramani. Optimal Length Resolution Refutations of Difference Constraint Systems. *Journal of Automated Reasoning (JAR)*, 43(2):121–137, 2009.

30 K. Subramani and Piotr Wojciechowki. A Polynomial Time Algorithm for Read-Once Certification of Linear Infeasibility in UTVPI Constraints. *Algorithmica*, 81(7):2765–2794, 2019.

31 Stefan Szeider. NP-completeness of refutability by literal-once resolution. In *Automated Reasoning, First International Joint Conference, IJCAR 2001, Siena, Italy, June 18-23, 2001, Proceedings*, pages 168–181, 2001.

32 A.F. Veinott and G.B. Dantzig. Integral Extreme points. *SIAM Review*, 10:371–372, 1968.