Logical Methods in Computer Science
Volume 21, Issue 2, 2025, pp. 6:1–6:25
https://lmcs.episciences.org/

Submitted Apr. 08, 2024
Published Apr. 21, 2025

FMPLEX: EXPLORING A BRIDGE BETWEEN FOURIER-MOTZKIN AND SIMPLEX

VALENTIN PROMIES [a], JASPER NALBACH [a], ERIKA ÁBRAHÁM [a], AND PAUL KOBIALKA [b]

[a] RWTH Aachen University, Germany
e-mail address: nalbach@cs.rwth-aachen.de, promies@cs.rwth-aachen.de, abraham@cs.rwth-aachen.de

[b] University of Oslo, Norway
e-mail address: paulkobi@ifi.uio.no

Abstract. In this paper we present a quantifier elimination method for conjunctions of *linear real arithmetic* constraints. Our algorithm is based on the *Fourier-Motzkin variable elimination* procedure, but by case splitting we are able to reduce the worst-case complexity from doubly to singly exponential. The adaption of the procedure for SMT solving has strong correspondence to the *simplex algorithm*, therefore we name it *FMplex*. Besides the theoretical foundations, we provide an experimental evaluation in the context of SMT solving. This is an extended version of the authors' work previously published at the fourteenth International Symposium on Games, Automata, Logics, and Formal Verification (GandALF 2023).

1. Introduction

Linear real arithmetic (LRA) is a powerful first-order theory with strong practical relevance. We focus on checking the satisfiability of *conjunctions* of LRA constraints, which is needed e.g. for solving quantifier-free LRA formulas using *satisfiability modulo theories (SMT) solvers*. This problem is known to be solvable in *polynomial* worst-case complexity but, surprisingly, the *ellipsoid* method [Kha80] proposed in 1980 by Khachiyan is still the only available algorithm that implements this bound. However, this method is seldom used in practice due to its high average-case effort. Instead, most approaches employ the *simplex* algorithm introduced by Dantzig in 1947, which is quite efficient in practice, despite a *singly exponential* worst case complexity. A third available solution is the *Fourier-Motzkin variable elimination (FM)* method, proposed in 1827 by Fourier [Fou27] and re-discovered in 1936 by Motzkin [Mot36]. In contrast to the other two approaches, FM admits quantifier elimination, but it has a *doubly exponential* worst case complexity, even though there have been various efforts to improve its efficiency by detecting and avoiding redundant computations (e.g. [Imb93, JMMT20]).

Key words and phrases: SMT solving, linear real arithmetic, Fourier-Motzkin elimination, simplex.

Jasper Nalbach and Valentin Promies were supported by the Deutsche Forschungsgemeinschaft (DFG) as part of AB 461/9-1 *SMT-ART*. Jasper Nalbach was also supported by the DFG RTG 2236 UnRAVeL .

LOGICAL METHODS IN COMPUTER SCIENCE
DOI:10.46298/LMCS-21(2:6)2025

In recent work [NPÁK23], we introduced a novel method, which is derived from the FM method, but which turns out to have striking resemblance to the simplex algorithm. This yields interesting theoretical insights into the relation of the two established methods and the nature of the problem itself. Here, we provide an extended version of [NPÁK23], and our contributions naturally include those of the original paper:

- The presentation of *FMplex*, a new variable elimination method based on a divide-and-conquer approach. We show that it does not contain certain redundancies Fourier-Motzkin might generate and lowers the overall complexity from *doubly* to *singly* exponential.
- An adaptation of FMplex for SMT solving, including methods to prune the search tree based on structural observations.
- A theorem formalizing connections between FMplex and the simplex algorithm.
- An implementation of the SMT adaptation and its experimental evaluation.

We extend our previous work by the following novel contributions:

- Additional and more detailed explanations, thereby improving readability and making some insights more explicit. In particular, we explain in Section 3 how the Fourier-Motzkin method still has advantages when only few variables are eliminated, and we added Section 4.3 discussing how to handle strict constraints.
- Extensive and reworked examples illustrating more aspects and intricacies of the algorithms. In particular, Definition 2.5 and Example 2.7 concerning redundancies, as well as Example 3.8 demonstrating the worst-case complexity are new. Moreover, Examples 4.5, 3.6 and 4.10 (Examples 2, 3, 4 in [NPÁK23]) have been extended and enriched with illustrations.
- Full and rigorous proofs for the central Theorems 4.2, 4.3 and 4.8 (Theorems 6, 7, 10 in [NPÁK23]). These proofs are not trivial, and they provide interesting insights into the problem's nature.
- We improved readability of some notation in Section 4.1 as well as some notation for matrices.

After recalling necessary preliminaries in Section 2, we introduce our FMplex method first for variable elimination in Section 3 and then for SMT solving in Section 4. We present related work and compare FMplex with other methods, first qualitatively in Section 5, and then experimentally in Section 6. We discuss future work and conclude the paper in Section 7.

2. Preliminaries

Let $\mathbb{R}$, $\mathbb{Q}$ and $\mathbb{N}$ denote the set of real, rational respectively natural ($0 \notin \mathbb{N}$) numbers. For $k \in \mathbb{N}$ we define $[k] := \{1, \dots, k\}$. Throughout this paper, we fix $n \in \mathbb{N}$, a set $X = \{x_1, \dots, x_n\}$ and a corresponding vector $\boldsymbol{x} = (x_1, \dots, x_n)^T$ of $\mathbb{R}$-valued variables.

Matrices. We use bold lower-case letters (e.g. $\boldsymbol{f}$) to denote vectors and upper case letters (e.g. A) for matrices. For any $m \in \mathbb{N}$ let $E \in \mathbb{Q}^{m \times m}$ be the identity matrix, and let $\mathbf{0} = (0 \ \cdots \ 0)^T \in \mathbb{Q}^{m \times 1}$ be the zero vector. The dimensions of E and $\mathbf{0}$ will be clear from the context. The i-th component of $\boldsymbol{f} \in \mathbb{Q}^{m \times 1} \cup \mathbb{Q}^{1 \times m}$ is denoted by f_i and the component-wise comparison to zero by $\boldsymbol{f} \geq 0$.

For $A \in \mathbb{Q}^{m \times n}$, $\boldsymbol{a}_{i,\text{-}} \in \mathbb{Q}^{1 \times n}$ and $\boldsymbol{a}_{\text{-},i} \in \mathbb{Q}^{m \times 1}$ denote the i-th row respectively column vector of A. Furthermore, $A[I]$ denotes the sub-matrix of A containing only the rows with indices from some $I \subseteq [m]$. That is, the rows of $A[I] \in \mathbb{Q}^{|I| \times n}$ are exactly the $\boldsymbol{a}_{i,\text{-}}$ with $i \in I$.

For $\boldsymbol{f} \in \mathbb{Q}^{1\times m}$, $\boldsymbol{f}A$ is a *linear combination* of the rows $i \in [m]$ of A with $f_i \neq 0$. We call A *linearly independent* if none of its rows is a linear combination of its other rows, and *linearly dependent* otherwise. The *rank of* A *rank*(A) is the maximal size of an $I \subseteq [m]$ so that $A_{[I]}$ is linearly independent.

Linear Constraints. Let $\boldsymbol{a} = (a_1, \ldots, a_n) \in \mathbb{Q}^{1\times n}$, $b \in \mathbb{Q}$ and $\sim \in \{=, \leq, <, \neq\}$ a *relation symbol*. We call $\boldsymbol{ax} := (a_1x_1 + \ldots + a_nx_n)$ a *linear term* and $\boldsymbol{ax} \sim b$ a *linear constraint*, which is *weak* if $\sim \in \{=, \leq\}$ and *strict* otherwise. A *system of linear constraints*, or short a *system*, is a non-empty finite set of linear constraints. In most parts of this paper, we only consider constraints of the form $\boldsymbol{ax} \leq b$. We can write every system $C = \{\boldsymbol{a}_{i,\text{-}}\, \boldsymbol{x} \leq b_i \mid i \in [m]\}$ of such constraints in *matrix representation* $A\boldsymbol{x} \leq \boldsymbol{b}$ with suitable $A \in \mathbb{Q}^{m\times n}$ and $\boldsymbol{b} \in \mathbb{Q}^{m\times 1}$. Conversely, every row $\boldsymbol{a}_{i,\text{-}}\, \boldsymbol{x} \leq b_i,\ i \in [m]$ of $A\boldsymbol{x} \leq \boldsymbol{b}$ is a linear constraint. Thus, the representations are mostly interchangeable; however, the matrix representation allows identical rows in contrast to the set notation. As the latter will play a role later on, we will stick to the matrix representation.

Variable Assignments. An *assignment* is a function $\alpha : Y \to \mathbb{R}$ with domain $dom(\alpha) = Y \subseteq X$. For an assignment α, a variable x_i and some $r \in \mathbb{R}$, the *extension* $\alpha[x_i \mapsto r]$ is the assignment with domain $dom(\alpha) \cup \{x_i\}$ such that $\alpha[x_i \mapsto r](x_j) = \alpha(x_j)$ for all $x_j \in dom(\alpha) \setminus \{x_i\}$ and $\alpha[x_i \mapsto r](x_i) = r$. For $Z \subseteq Y$, the *restriction* $\alpha|_Z$ is the assignment with domain Z such that $\alpha|_Z(x_i) = \alpha(x_i)$ for all $x_i \in Z$. We extend these notations to sets of assignments accordingly.

The standard *evaluation* of a linear term t under α is written $\alpha(t)$. We say that α *satisfies* (or is a solution of) a constraint $c = (\boldsymbol{ax} \sim b)$ if $\alpha(\boldsymbol{ax}) \sim b$ holds, and denote this fact by $\alpha \models c$. All solutions of c build its *solution set* $sol(c)$. Similarly, $\alpha \models (A\boldsymbol{x} \leq \boldsymbol{b})$ denotes that α is a common solution of all linear constraints in the system $A\boldsymbol{x} \leq \boldsymbol{b}$. A system is *satisfiable* if it has a common solution, and *unsatisfiable* otherwise. Note that each satisfiable system has also a rational-valued solution.

We will make use of the following two well-known results.

Theorem 2.1 (Farkas' Lemma [Far02])**.** *Let $A \in \mathbb{Q}^{m\times n}$ and $\boldsymbol{b} \in \mathbb{Q}^{m\times 1}$. Then the system $A\boldsymbol{x} \leq \boldsymbol{b}$ is satisfiable if and only if for all $\boldsymbol{f} \in \mathbb{Q}^{1\times m}$ with $\boldsymbol{f} \geq 0$ and $\boldsymbol{f}A = \boldsymbol{0}$ it holds $\boldsymbol{fb} \geq 0$.*

Theorem 2.2 (Fundamental Theorem of Linear Programming, as in [LY+84])**.** *Assume $A \in \mathbb{Q}^{m\times n}$ and $\boldsymbol{b} \in \mathbb{Q}^{m\times 1}$. Then $A\boldsymbol{x} \leq \boldsymbol{b}$ is satisfiable if and only if there exists a subset $I \subseteq [m]$ such that $A_{[I]}$ is linearly independent, $|I| = rank(A)$, and there exists an assignment $\alpha : X \to \mathbb{R}$ with $\alpha \models (A\boldsymbol{x} \leq \boldsymbol{b})$ and $\alpha \models (A_{[I]}\boldsymbol{x} = \boldsymbol{b}_{[I]})$.*

2.1. **Fourier-Motzkin Variable Elimination.** To eliminate any variable $x_j \in X$ from a system $A\boldsymbol{x} \leq \boldsymbol{b}$, the *Fourier-Motzkin* (FM) [Fou27, Mot36] method computes another system[1] $A'\boldsymbol{x} \leq \boldsymbol{b'}$ with $\boldsymbol{a'}_{\text{-},j} = 0$ and such that an assignment α is a solution of $A'\boldsymbol{x} \leq \boldsymbol{b'}$ if and only if there is $r \in \mathbb{Q}$ so that $\alpha[x_j \mapsto r]$ is a solution of $A\boldsymbol{x} \leq \boldsymbol{b}$. Graphically, the solution set of $A'\boldsymbol{x} \leq \boldsymbol{b'}$ is the projection of the solutions of $A\boldsymbol{x} \leq \boldsymbol{b}$ onto $X \setminus \{x_j\}$.

[1]Note that A and A' have the same number of columns, but the column $\boldsymbol{a}'_{\text{-},j}$ corresponding to x_j is zero. This presentation makes our later steps easier to formulate in terms of matrix multiplications.

The idea of the FM method is as follows. For each $i \in [m]$ with $a_{i,j} \neq 0$, the constraint $\boldsymbol{a}_{i,\text{-}}\, \boldsymbol{x} \leq b_i$ can be rewritten as either a *lower bound* or an *upper bound* on x_j:

$$x_j \sim \frac{1}{a_{i,j}}\big(b_i - \sum_{k\in[n]\setminus\{j\}} a_{i,k}x_k\big), \text{ where } \sim = \begin{cases} \geq & \text{if } a_{i,j} < 0 \\ \leq & \text{if } a_{i,j} > 0 \end{cases}$$

The term defining the bound is the same in both cases, and we denote it as $bnd_j(\boldsymbol{a}_{i,\text{-}}\, \boldsymbol{x} \leq b_i)$.

Definition 2.3. For $A \in \mathbb{Q}^{m\times n}$, we define the following three index sets:

$$I_j^-(A) := \{i \in [m] \mid a_{i,j} < 0\} \quad I_j^+(A) := \{i \in [m] \mid a_{i,j} > 0\} \quad I_j^0(A) := \{i \in [m] \mid a_{i,j} = 0\}$$

$I_j^-(A)$, $I_j^+(A)$ and $I_j^0(A)$ indicate the rows of $A\boldsymbol{x} \leq \boldsymbol{b}$ which induce lower bounds, upper bounds and no bounds on x_j, respectively.

Due to the density of the reals, there exists a value for x_j that satisfies all bounds if and only if each lower bound is less than or equal to each upper bound. However, in general the involved bounds are symbolic, and thus their values depend on the values of other variables. Therefore, we cannot directly check whether the lower bounds are equal or below the upper bounds, but instead we express this by a new constraint set $A'\boldsymbol{x} \leq \boldsymbol{b'}$, which is defined by the following:

$$\{bnd_j(\boldsymbol{a}_{\ell,\text{-}}\, \boldsymbol{x} \leq b_\ell) \leq bnd_j(\boldsymbol{a}_{u,\text{-}}\, \boldsymbol{x} \leq b_u) \mid (\ell, u) \in I_j^-(A) \times I_j^+(A)\} \cup \{\boldsymbol{a}_{i,\text{-}}\, \boldsymbol{x} \leq b_i \mid i \in I_j^0(A)\}$$

Formulating this idea with respect to the matrix representation, the FM method applies the following transformation, using matrix multiplications. Recall that E is the identity matrix, and thus $\boldsymbol{e}_{i,\text{-}}$ is the i-th unit vector.

Definition 2.4 (Fourier-Motzkin Variable Elimination)**.** Let $A \in \mathbb{Q}^{m\times n}$, $\boldsymbol{b} \in \mathbb{Q}^{m\times 1}$, and $j \in [n]$. Let further $m' = |I_j^-(A)| \cdot |I_j^+(A)| + |I_j^0(A)|$ and $F \in \mathbb{Q}^{m'\times m}$ be a matrix consisting of exactly the following rows:

$$a_{u,j}^{-1} \cdot \boldsymbol{e}_{u,\text{-}} - a_{\ell,j}^{-1} \cdot \boldsymbol{e}_{\ell,\text{-}} \text{ for every pair } (\ell, u) \in I_j^-(A) \times I_j^+(A) \qquad \text{and}$$
$$\boldsymbol{e}_{i,\text{-}} \text{ for every } i \in I_j^0(A).$$

Then the *Fourier-Motzkin variable elimination* of x_j from the system $A\boldsymbol{x} \leq \boldsymbol{b}$ is defined as the system $FA\boldsymbol{x} \leq F\boldsymbol{b}$.

The FM method can also be used to check the consistency (i.e. satisfiability) of a system $A\boldsymbol{x} \leq \boldsymbol{b}$, by successively eliminating all variables $x_n, \ldots, x_1$. This yields intermediate systems $A^{(n-1)}\boldsymbol{x} \leq \boldsymbol{b}^{(n-1)}, \ldots, A^{(0)}\boldsymbol{x} \leq \boldsymbol{b}^{(0)}$, where the last one does not contain any variable, i.e. $A^{(0)} = 0$. Note that all entries of the transformation matrix F in the definition above are positive, and thus for any $k \in \{0, \ldots, n-1\}$ and any row i' in $A^{(k)}\boldsymbol{x} \leq \boldsymbol{b}^{(k)}$, there exists $\boldsymbol{f} \in \mathbb{Q}^{1\times m}$ such that $\boldsymbol{f} \geq 0$, $\boldsymbol{f}A = \boldsymbol{a}_{i',\text{-}}^{(k)}$ and $\boldsymbol{f}\boldsymbol{b} = b_{i'}^{(k)}$.We call this kind of linear combinations *conical combinations*. Now we know by Farkas' Lemma (Theorem 2.1) that, if $A^{(0)}\boldsymbol{x} \leq \boldsymbol{b}^{(0)}$ is unsatisfiable, then so is $A\boldsymbol{x} \leq \boldsymbol{b}$. On the other hand, if $A^{(0)}\boldsymbol{x} \leq \boldsymbol{b}^{(0)}$ is satisfiable, then it is satisfied by the empty assignment, which can be extended successively to a model of $A^{(1)}\boldsymbol{x} \leq \boldsymbol{b}^{(1)}, \ldots, A^{(n-1)}\boldsymbol{x} \leq \boldsymbol{b}^{(n-1)}$ and $A\boldsymbol{x} \leq \boldsymbol{b}$.

A major drawback of the Fourier-Motzkin variable elimination is its doubly exponential worst-case complexity in time and space w.r.t. the number of eliminated variables. Moreover, many of the generated rows are redundant because they are conical combinations of the other rows, i.e. they could be omitted without changing the solution set of the system.

Definition 2.5 (Redundancy)**.** Let $A \in \mathbb{Q}^{m\times n}$ and $\boldsymbol{b} \in \mathbb{Q}^{m\times 1}$. Let further $i \in [m]$ and $I := [m] \setminus \{i\}$. We call $\boldsymbol{a}_{i,\text{-}}\boldsymbol{x} \leq b_i$ *redundant* in $A\boldsymbol{x} \leq \boldsymbol{b}$, if $sol(A\boldsymbol{x} \leq \boldsymbol{b}) = sol(A_{[I]}\boldsymbol{x} \leq \boldsymbol{b}_{[I]})$.

Redundancies might already be contained in the input system, or they are introduced by the projection operation. While removing all redundancies is expensive, there are efficient methods, e.g. Imbert's acceleration theorems [Imb90, Imb93, JMMT20], for removing some redundancies of the latter type. The following lemma describes this kind of redundancy.

Lemma 2.6 (Redundancy by Construction)**.** *Let $A \in \mathbb{Q}^{m\times n}, \boldsymbol{b} \in \mathbb{Q}^{m\times 1}$ and $F \in \mathbb{Q}^{m'\times m}$. Let furthermore $A' = FA$, $\boldsymbol{b'} = F\boldsymbol{b}$ and $i \in [m']$. If there exists $\boldsymbol{r} \in \mathbb{Q}^{1\times m'}$ with $\boldsymbol{r} \geq 0$, $r_i = 0$ and $\boldsymbol{r}F = \boldsymbol{f}_{i,\text{-}}$ (i.e. the i-th row of $A'\boldsymbol{x} \leq \boldsymbol{b'}$ is a conical combination $\boldsymbol{r}FA\boldsymbol{x} \leq \boldsymbol{r}F\boldsymbol{b}$ ($\boldsymbol{r}A'\boldsymbol{x} \leq \boldsymbol{r}\boldsymbol{b'}$) of the other rows), then that row is redundant in $A'\boldsymbol{x} \leq \boldsymbol{b'}$.*

Example 2.7. Consider the system on the left of Figure 1 on which we applied Fourier-Motzkin to eliminate x_1 and x_2 to obtain the system on the right, whose first row is redundant by construction. The matrix F from Lemma 2.6 is encoded in the row labels. Note that we omit the eliminated variables and corresponding zero-columns to save space.

$$\begin{array}{l} c_1 \\ c_2 \\ c_3 \\ c_4 \end{array}\!\!\begin{bmatrix} 1 & 1 & 1 \\ 1 & -1 & 1 \\ -1 & 1 & 1 \\ -1 & -1 & 1 \end{bmatrix} \cdot \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} \leq \begin{bmatrix} 0 \\ 0 \\ 1 \\ 1 \end{bmatrix} \quad \xrightarrow{\text{elim. } x_1 \text{ and } x_2} \quad \begin{array}{r} c_1+c_2+c_3+c_4 \\ c_1+c_4 \\ c_2+c_3 \end{array}\!\!\begin{bmatrix} 4 \\ 2 \\ 2 \end{bmatrix} \cdot \begin{bmatrix} x_3 \end{bmatrix} \leq \begin{bmatrix} 2 \\ 1 \\ 1 \end{bmatrix}$$

FIGURE 1. Fourier-Motzkin elimination steps from Example 2.7.

3. FMplex as Variable Elimination Procedure

The FM method encodes that none of the lower bounds on some variable x_j in a system $A\boldsymbol{x} \leq \boldsymbol{b}$ is larger than any of its upper bounds. In our *FMplex* method, we do not consider all lower-upper bound combinations at once. Instead, we *split the problem into a set of sub-problems* by a case distinction either on *which of the lower bounds are the largest* or, alternatively, on *which of the upper bounds are the smallest*. For the case distinction on lower bounds, we create one sub-problem for each lower bound on x_j, and in this sub-problem we consider solutions where that lower bound is maximal among all lower bounds, and at the same time not larger than any of the upper bounds. The idea is that, if we find a solution for one of the sub-problems, it is also a solution of the original system. On the other hand, any solution of $A\boldsymbol{x} \leq \boldsymbol{b}$ will satisfy one of the sub-problems, since one of the lower bounds has to be maximal for that assignment. The case distinction for upper bounds works analogously.

Asymptotically, these sub-problems are significantly smaller than the systems produced by FM, so that in total our approach produces *at most exponentially* many constraints after iterated application, in contrast to the doubly exponential effort of the FM method.

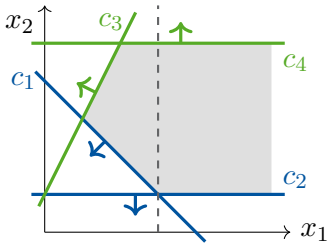
Formally, if there are no upper or no lower bounds on x_j, then there is no need for case splitting and we follow FM using $\exists x_j.\ A\boldsymbol{x} \leq \boldsymbol{b} \equiv A_{[I_j^0(A)]}\boldsymbol{x} \leq \boldsymbol{b}_{[I_j^0(A)]}$. Otherwise, for the sub-problem when designating $i \in I_j^-(A)$ as a largest lower bound, we encode that no other

lower bound is larger than the bound induced by row i, and no upper bound is below this bound. Using the set notation for systems, we obtain

$$\{bnd_j(\boldsymbol{a}_{i',\text{-}}\,\boldsymbol{x} \le b_{i'}) \le bnd_j(\boldsymbol{a}_{i,\text{-}}\,\boldsymbol{x} \le b_i) \mid i' \in I_j^-(A) \setminus \{i\}\}$$
$$\cup\, \{bnd_j(\boldsymbol{a}_{i,\text{-}}\,\boldsymbol{x} \le b_i) \le bnd_j(\boldsymbol{a}_{i',\text{-}}\,\boldsymbol{x} \le b_{i'}) \mid i' \in I_j^+(A)\} \cup \{\boldsymbol{a}_{i',\text{-}}\,\boldsymbol{x} \le b_{i'} \mid i' \in I_j^0(A)\}.$$

Example 3.1. We eliminate x_2 from the system $A\boldsymbol{x} \le \boldsymbol{b}$ consisting of the lower-bounding constraints c_1 and c_2, and the upper-bounding c_3 and c_4, specified below along with a graphical depiction.

$$\begin{matrix} c_1 \\ c_2 \\ c_3 \\ c_4 \end{matrix}\!\!\begin{bmatrix} -1 & -1 \\ 0 & -2 \\ -2 & 1 \\ 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} \le \begin{bmatrix} -4 \\ -2 \\ 1 \\ 5 \end{bmatrix}$$



The lower bounds $I_2^-(A) = \{1, 2\}$ on x_2 are blue, the upper bounds $I_2^+(A) = \{3, 4\}$ are green. The solution set is the gray area and the dashed line indicates the split into two sub-problems, namely the cases that c_1 resp. c_2 is a largest lower bound on x_2 and not larger than any upper bound on x_2. The encoding of the c_1-case is given by

$$bnd_2(c_2) \le bnd_2(c_1) \quad \wedge \quad bnd_2(c_1) \le bnd_2(c_3) \quad \wedge \quad bnd_2(c_1) \le bnd_2(c_4),$$

where the first constraint expresses that c_1 gives the greatest lower bound and the other two ensure that it does not exceed the upper bounds. In normal form, this formula evaluates to $(x_1 \le 3) \wedge (-3x_1 \le -3) \wedge (-x_1 \le 1)$ satisfied by any $x_1 \in [1, 3]$, on the left of the dashed line. The case for c_2 evaluates to $(-x_1 \le -3) \wedge (-2x_1 \le 0) \wedge (0 \le 4)$ and is satisfiable on the right of the dashed line. The disjunction of the two formulas then defines exactly those values for x_1 which allow a solution of the initial system, i.e. $x_1 \in [1, 3]$ or $x_1 \in [3, \infty)$.

The construction for the case $i \in I_j^+(A)$ designating i as smallest upper bound is analogous. Like for FM, the constructed constraints $bnd_j(\boldsymbol{a}_{i',\text{-}}\,\boldsymbol{x} \le b_{i'}) \le bnd_j(\boldsymbol{a}_{i,\text{-}}\,\boldsymbol{x} \le b_i)$ are equivalent to linear combinations of the system $A\boldsymbol{x} \le \boldsymbol{b}$. However, combinations of two lower bounds or two upper bounds use negative coefficients, meaning that these linear combinations are not necessarily *conical*. Nevertheless, we can formulate the sub-problems with respect to the matrix representation, resulting in the following transformation:

Definition 3.2 (Restricted Projection)**.** Let $A \in \mathbb{Q}^{m\times n}$, $\boldsymbol{b} \in \mathbb{Q}^{m\times 1}$ and $j \in [n]$.

- If $I_j^-(A) \neq \emptyset$ and $I_j^+(A) \neq \emptyset$, then for any $i \in I_j^-(A) \cup I_j^+(A)$ we fix $F \in \mathbb{Q}^{(m-1)\times m}$ arbitrarily but deterministically to consist of exactly the following rows:

  $$\frac{1}{a_{i,j}} \cdot \boldsymbol{e}_{i,\text{-}} - \frac{1}{a_{i',j}} \cdot \boldsymbol{e}_{i',\text{-}} \text{ for every } i' \in I_j^-(A) \setminus \{i\},$$
  $$-\frac{1}{a_{i,j}} \cdot \boldsymbol{e}_{i,\text{-}} + \frac{1}{a_{i',j}} \cdot \boldsymbol{e}_{i',\text{-}} \text{ for every } i' \in I_j^+(A) \setminus \{i\}, \qquad \text{and} \qquad \boldsymbol{e}_{i',\text{-}} \text{ for every } i' \in I_j^0(A).$$

  Then the *restricted projection* $P_{j,i}(A\boldsymbol{x} \le \boldsymbol{b})$ of x_j from the system $A\boldsymbol{x} \le \boldsymbol{b}$ with respect to the row i is defined as the system $FA\boldsymbol{x} \le F\boldsymbol{b}$. We call F the *projection matrix* corresponding to $P_{j,i}(A\boldsymbol{x} \le \boldsymbol{b})$.

- If $I_j^-(A) = \emptyset$ or $I_j^+(A) = \emptyset$, then we define the projection matrix $F \in \mathbb{Q}^{|I_j^0(A)| \times m}$ to have exactly one row $\boldsymbol{e}_{i',\text{-}}$ for each $i' \in I_j^0(A)$, and define $P_{j,\perp}(A\boldsymbol{x} \leq \boldsymbol{b})$ as $FA\boldsymbol{x} \leq F\boldsymbol{b}$.

We call this a restricted projection because it defines exactly the part of the projection, where the respective bound is a greatest lower or least upper bound. Crucially, the solutions of the restricted projections for all lower (or all upper) bounds of a variable exactly cover the projection of the entire solution set, as is demonstrated by the following lemma.

Lemma 3.3. *Let $A \in \mathbb{Q}^{m\times n}$, $\boldsymbol{b} \in \mathbb{Q}^{m\times 1}$, $j \in [n]$. If $I_j^-(A) \neq \emptyset$ and $I_j^+(A) \neq \emptyset$, then*

$$sol(A\boldsymbol{x} \leq \boldsymbol{b})|_{X\setminus\{x_j\}} = \bigcup_{i\in I_j^-(A)} sol(P_{j,i}(A\boldsymbol{x} \leq \boldsymbol{b})) = \bigcup_{i\in I_j^+(A)} sol(P_{j,i}(A\boldsymbol{x} \leq \boldsymbol{b})).$$

Otherwise ($I_j^-(A) = \emptyset$ or $I_j^+(A) = \emptyset$), it holds $sol(A\boldsymbol{x} \leq \boldsymbol{b})|_{X\setminus\{x_j\}} = sol(P_{j,\perp}(A\boldsymbol{x} \leq \boldsymbol{b}))$.

Proof. The case $I_j^-(A) = \emptyset$ or $I_j^+(A) = \emptyset$ follows from the correctness of FM. We show the equality for $I_j^-(A)$, the case for $I_j^+(A)$ is analogous.

$\supseteq$ **:** Let $i \in I_j^-(A)$ and $\alpha \models P_{j,i}(A\boldsymbol{x} \leq \boldsymbol{b})$, then for all $\ell \in I_j^-(A)$, $u \in I_j^+(A)$ it holds

$$\alpha(bnd_j(\boldsymbol{a}_{\ell,\text{-}}\ \boldsymbol{x} \leq b_\ell)) \leq \alpha(bnd_j(\boldsymbol{a}_{i,\text{-}}\ \boldsymbol{x} \leq b_i)) \leq \alpha(bnd_j(\boldsymbol{a}_{u,\text{-}}\ \boldsymbol{x} \leq b_u)).$$

Thus, $\alpha[x_j \mapsto \alpha(bnd_j(\boldsymbol{a}_{i,\text{-}}\ \boldsymbol{x} \leq b_i))] \models A\boldsymbol{x} \leq \boldsymbol{b}$.

$\subseteq$ **:** Let $\alpha \models A\boldsymbol{x} \leq \boldsymbol{b}$. Then, by Fourier-Motzkin, for all $\ell \in I_j^-(A)$ and $u \in I_j^+(A)$ it holds $\alpha(bnd_j(\boldsymbol{a}_{\ell,\text{-}}\ \boldsymbol{x} \leq b_\ell)) \leq \alpha(bnd_j(\boldsymbol{a}_{u,\text{-}}\ \boldsymbol{x} \leq b_u))$. Let $i \in I_j^-(A)$ so that

$$\alpha(bnd_j(\boldsymbol{a}_{i,\text{-}}\ \boldsymbol{x} \leq b_i)) = \max\{\alpha(bnd_j(\boldsymbol{a}_{\ell,\text{-}}\ \boldsymbol{x} \leq b_\ell)) \mid \ell \in I_j^-(A)\},$$

then $\alpha \models P_{j,i}(A\boldsymbol{x} \leq \boldsymbol{b})$. □

Definition 3.4 (FMplex Variable Elimination)**.** For $A \in \mathbb{Q}^{m\times n}$, $\boldsymbol{b} \in \mathbb{Q}^{m\times 1}$, $j \in [n]$ and $* \in \{-,+\}$, we define

$$\texttt{FMP}_j^*(A\boldsymbol{x} \leq \boldsymbol{b}) = \begin{cases} \{P_{j,i}(A\boldsymbol{x} \leq \boldsymbol{b}) \mid i \in I_j^*(A)\} & \text{if } I_j^-(A) \neq \emptyset \text{ and } I_j^+(A) \neq \emptyset \\ \{P_{j,\perp}(A\boldsymbol{x} \leq \boldsymbol{b})\} & \text{otherwise.} \end{cases}$$

The FMplex elimination defines a set of restricted projections which can be composed to the full projection according to Lemma 3.3. Lifting this from sets to logic naturally results in the following theorem which demonstrates the usage of our method.

Theorem 3.5. *Let $A \in \mathbb{Q}^{m\times n}$, $\boldsymbol{b} \in \mathbb{Q}^{m\times 1}$, and $j \in [n]$. Then*

$$\exists x_j.\ A\boldsymbol{x} \leq \boldsymbol{b} \quad \equiv \bigvee_{S\in \texttt{FMP}_j^-(A\boldsymbol{x}\leq\boldsymbol{b})} S \quad \equiv \bigvee_{S\in \texttt{FMP}_j^+(A\boldsymbol{x}\leq\boldsymbol{b})} S.$$

Proof. This immediately follows from Lemma 3.3. □

For eliminating multiple variables, we iteratively apply $\texttt{FMP}^-$ or $\texttt{FMP}^+$ to each restricted projection resulting from the previous elimination step. Note that we can choose the next variable to be eliminated as well as the bound type independently in every branch.

Example 3.6. We continue Example 3.1, where we eliminated x_2, and next eliminate x_1:

$$\exists x_1.\ \exists x_2.\ A\boldsymbol{x} \leq \boldsymbol{b} \quad \equiv \quad \exists x_1. \bigvee_{S\in \texttt{FMP}_2^-(A\boldsymbol{x}\leq\boldsymbol{b})} S \quad \equiv \quad \exists x_1.\ (x_1 \leq 3 \wedge -3x_1 \leq -3 \wedge -x_1 \leq 1)$$
$$\vee\ \exists x_1.\ (-x_1 \leq -3 \wedge -2x_1 \leq 0 \wedge 0 \leq 4)$$

We eliminate the two quantifiers for x_1 separately, using

$$\texttt{FMP}_1^-(\ x_1 \le 3 \wedge -3x_1 \le -3 \wedge -x_1 \le 1\ ) = \{\ (-1 \le 1 \wedge 1 \le 3),\ (1 \le -1 \wedge -1 \le 3)\ \}$$

and $\texttt{FMP}_1^-(\ -x_1 \le -3 \wedge -2x_1 \le 0 \wedge\ 0 \le 4\ ) = \{\ (0 \le 4)\ \}$, giving us the final result

$$\exists x_1.\ \exists x_2.\ A\boldsymbol{x} \le \boldsymbol{b} \quad \equiv \quad \big(\ (-1 \le 1 \wedge 1 \le 3) \quad \vee \quad (1 \le -1 \wedge -1 \le 3)\ \big) \quad \vee \quad (0 \le 4).$$

Note that in any of the systems, we could use $\texttt{FMP}^+$ instead of $\texttt{FMP}^-$. For example, the first disjunct obtained from eliminating x_2 contains two lower bounds on x_1, but only one upper bound. Thus, one can reduce the number of resulting disjuncts, by instead computing

$$\texttt{FMP}_1^+(\ x_1 \le 3 \wedge -3x_1 \le -3 \wedge -x_1 \le 1\ ) = \{(1 \le 3 \wedge -1 \le 3)\}.$$

We analyze the complexity in terms of the number of new rows (or constraints) that are constructed during the elimination of all variables:

Theorem 3.7 (Complexity of FMP)**.** *Let $A \in \mathbb{Q}^{m\times n}$, and $\boldsymbol{b} \in \mathbb{Q}^{m\times 1}$. When eliminating n variables from $A\boldsymbol{x} \le \boldsymbol{b}$ by repeated application of $\texttt{FMP}^-$ or $\texttt{FMP}^+$, at most $\mathcal{O}(n \cdot m^{n+1})$ new rows are constructed.*

Proof. Let $N(m, n)$ be the maximal number of constructed rows when eliminating n variables from a system of m rows. Applying $\texttt{FMP}^-$ or $\texttt{FMP}^+$ once leads to at most $m-1$ restricted projections, each of which contains at most $m-1$ rows and $n-1$ variables. This implies $N(m, n) \le (m-1) \cdot ((m-1) + N(m-1, n-1))$, and repeating this reasoning yields

$$N(m,n) \le \sum_{i=1}^{k}(m-i) \cdot \prod_{j=1}^{i}(m-j) \le n \cdot m^{n+1},\ \text{where } k = \min(n, m). \qquad \square$$

The following example illustrates that this exponential complexity can occur in practice, thus the bound is strong.

Example 3.8. We use $\texttt{FMP}^-$ to eliminate $x_1, \ldots, x_n$ in that order from the constraint set

$$\{\ -x_j - x_{n+1} \le 0 \mid j \in [n]\ \} \cup \{\ -x_j - 2x_{n+1} \le 0 \mid j \in [n]\ \} \cup \{\ x_1 + \ldots + x_{n+1} \le -1\ \}.$$

Any intermediate system where the first $k \in \{0, ..., n\}$ variables have been eliminated ($k = 0$ gives the initial system) has the following form: There are two lower bounds $-x_j - x_{n+1} \le 0$ and $-x_j - 2x_{n+1} \le 0$ on each of the remaining variables $x_{k+1}, \ldots, x_n$, and there is one constraint $x_{k+1} + \ldots + x_n + ax_{n+1} \le -1$ which is the only upper bound on all those variables. This upper bound is the result of combining $(x_1 + \ldots + x_{n+1} \le -1)$ with the chosen lower bounds and thus, $a < 0$ holds if $k > 1$. Moreover, there are constraints $(x_{n+1} \le 0)$ and/or $(-x_{n+1} \le 0)$ resulting from combining the two lower bounds.

Accordingly, each intermediate system has two branches for the next elimination, each of which requires the computation of two constraints. Thus, the total number of generated constraints is $2 \cdot (2 + 4 + \ldots + 2^n) = 2 \cdot (2^{n+1} - 2)$.

Interestingly, every leaf of this elimination contains a constraint $ax_{n+1} \le -1$, with $a < 0$, i.e. a lower bound on x_{n+1} that implies $x_{n+1} > 0$. If any of the greatest lower bounds chosen to create a leaf had the form $-x_j - 2x_{n+1} \le 0$, then that resulting leaf contains $x_{n+1} \le 0$ and is thus unsatisfiable. Consequently, the only satisfiable leaf is obtained by choosing the lower bound $-x_j - x_{n+1} \le 0$ for all $j \in \{0, \ldots, k\}$.

While still exponential, this bound is considerably better than the theoretical doubly exponential worst-case complexity of the FM method. Shortly speaking, FMplex trades one exponential step at the cost of the result being a decomposition into multiple partial

projections. However, there are systems for which FMplex produces strictly more rows than the FM method: assume a system of m constraints, with each $m/2$ upper and lower bounds on a variable x_1, then $\texttt{FMP}^-$ (or $\texttt{FMP}^+$) constructs $(m/2) \cdot (m-1)$ new constraints in the elimination of x_1, while FM would only compute $(m/2)^2$. Here, the advantage of FMplex becomes apparent only for the elimination of multiple variables, when the split into sub-problems prevents the doubly exponential growth.

Like FM, FMplex keeps redundancies from the input throughout the algorithm, thus there might be identical rows in the same or across different sub-problems. But in contrast to FM, FMplex does not introduce any redundancies by construction in the sense of Lemma 2.6.

Theorem 3.9. *Let $A \in \mathbb{Q}^{m \times n}$, $\boldsymbol{b} \in \mathbb{Q}^{m \times 1}$ and $k \in [m]$. Assume $(A^{(0)}\boldsymbol{x} \leq \boldsymbol{b}^{(0)}) = (A\boldsymbol{x} \leq \boldsymbol{b})$ and for all $j \in [k]$, let $(A^{(j)}\boldsymbol{x} \leq \boldsymbol{b}^{(j)}) \in \texttt{FMP}_j^-(A^{(j-1)}\boldsymbol{x} \leq \boldsymbol{b}^{(j-1)}) \cup \texttt{FMP}_j^+(A^{(j-1)}\boldsymbol{x} \leq \boldsymbol{b}^{(j-1)})$. Let $F^{(1)}, \ldots, F^{(k)}$ be the respective projection matrices, and $F = F^{(k)} \cdot \ldots \cdot F^{(1)}$. Then F is linearly independent.*

Proof. By definition, $F^{(k)}$, …, $F^{(1)}$ are linearly independent. Thus, F is also linearly independent as a product of linearly independent matrices. □

4. FMplex as Satisfiability Checking Procedure

A formula is satisfiable if and only if eliminating all variables (using any quantifier elimination method such as FM or FMplex) yields a tautology. However, FMplex applies case splitting such that satisfiability of any of the cases implies the satisfiability of the original problem. Therefore, we do not need to compute the whole projection at once, but we can explore the decomposition using a depth-first search. In this section, we first present a basic version of our algorithm to exploit this observation. Later, we examine how the search tree can be pruned, resulting in two additional variants. The resulting search tree has the original system as root, and each node has as children the systems resulting from restricted projections. The original system is satisfiable if and only if there is a leaf whose constraints are all tautologies. An example is depicted in Figure 3.

An important observation is that we can decide independently for each node of the search tree which variable to eliminate next and whether to branch on lower or on upper bounds. This freedom of choice is formalized in the following definition.

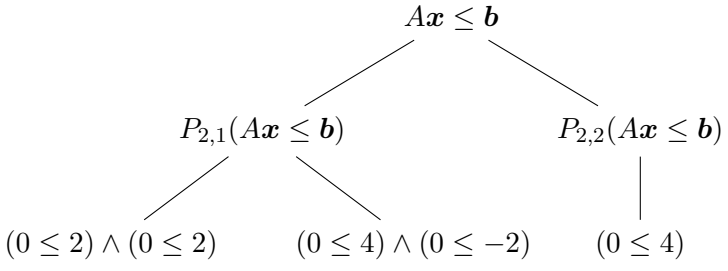


FIGURE 3. The search tree corresponding to Example 3.6. The very first leaf (bottom left) is already satisfiable, meaning that the rest would not need to be computed.

Definition 4.1 (Branch Choices)**.** The set of *branch choices* for a system $A\boldsymbol{x} \leq \boldsymbol{b}$ is

$$\begin{aligned} \mathit{branch\_choices}(A\boldsymbol{x} \leq \boldsymbol{b}) \;=\; & \{\; (x_j, I_j^*(A)) \mid * \in \{-,+\} \;\wedge\; j \in [n] \;\wedge\; I_j^-(A) \neq \emptyset \neq I_j^+(A) \;\} \\ & \cup\; \{\; (x_j, \{\bot\}) \mid j \in [n] \;\wedge\; (I_j^-(A) = \emptyset \vee I_j^+(A) = \emptyset) \;\}. \end{aligned}$$

For an initial input $\widehat{A}\boldsymbol{x} \leq \widehat{\boldsymbol{b}}$ with $\widehat{m}$ rows, we define the depth-first search using the recursive method $\texttt{FMplex}(\widehat{A}\boldsymbol{x} \leq \widehat{\boldsymbol{b}}; A\boldsymbol{x} \leq \boldsymbol{b}, F)$ in Algorithm 1 where $A\boldsymbol{x} \leq \boldsymbol{b}$ is the currently processed sub-problem in the recursion tree. The original system $\widehat{A}\boldsymbol{x} \leq \widehat{\boldsymbol{b}}$ with $\widehat{m}$ is not used in the algorithm itself, but included for illustrational purposes. We track the relation of $A\boldsymbol{x} \leq \boldsymbol{b}$ to $\widehat{A}\boldsymbol{x} \leq \widehat{\boldsymbol{b}}$ in terms of linear combinations using the parameter F. The initial call is defined as $\texttt{FMplex}(\widehat{A}\boldsymbol{x} \leq \widehat{\boldsymbol{b}}) = \texttt{FMplex}(\widehat{A}\boldsymbol{x} \leq \widehat{\boldsymbol{b}}; \widehat{A}\boldsymbol{x} \leq \widehat{\boldsymbol{b}}, E)$. We allow that $A\boldsymbol{x} \leq \boldsymbol{b}$ contains identical rows when they are obtained in different ways (which is reflected by F). We need to keep these duplicates for proving the results of this section.

Solutions. If a trivially satisfiable node is found, the algorithm constructs an assignment starting with the empty assignment and extending it with values for all eliminated variables in reverse elimination order. For every variable x_j, the assignment α is extended by assigning to x_j a value that us not below any lower bound and not above any upper bound on x_j, when the bounds are evaluated under α. By the semantics of the projection, the value of the designated (largest lower or smallest upper) bound on x_j is suitable.

Algorithm 1: $\texttt{FMplex}(\widehat{A}\boldsymbol{x} \leq \widehat{\boldsymbol{b}}; A\boldsymbol{x} \leq \boldsymbol{b}, F)$

Data : $\widehat{A} \in \mathbb{Q}^{\widehat{m}\times n},\ \widehat{\boldsymbol{b}} \in \mathbb{Q}^{\widehat{m}}$
Input : $A \in \mathbb{Q}^{m\times n},\ \boldsymbol{b} \in \mathbb{Q}^m,\ F \in \mathbb{Q}^{m\times\widehat{m}}$ s.t. $F\widehat{A} = A$ and $F\widehat{\boldsymbol{b}} = \boldsymbol{b}$
Output : (SAT, α) with $\alpha \models A\boldsymbol{x} \leq \boldsymbol{b}$, or (UNSAT, K) where the rows $K \subseteq [\widehat{m}]$ of $\widehat{A}\boldsymbol{x} \leq \widehat{\boldsymbol{b}}$ build a minimal unsatisfiable subset of constraints $\{\widehat{\boldsymbol{a}}_{k,-}\boldsymbol{x} \leq \widehat{b_k} \mid k \in K\}$, or *PARTIAL-UNSAT*

```
1  if A = 0 ∧ b ≥ 0 then return (SAT, α : ∅ → ℝ)                    // trivial sat
2  if ∃i ∈ [m]. a_{i,-} = 0 ∧ b_i < 0 ∧ f_{i,-} ≥ 0 then             // global conflict
3  |  return (UNSAT, {i' | f_{i,i'} ≠ 0})
4  choose (x_j, V) ∈ branch_choices(Ax ≤ b)
5  foreach i ∈ V do
6  |  compute A'x ≤ b' := P_{j,i}(Ax ≤ b) with projection matrix F'
7  |  switch FMplex (Âx ≤ b̂; A'x ≤ b', F' · F) do
8  |  |  case (UNSAT, K) do return (UNSAT, K)
9  |  |  case (SAT, α) do return (SAT, α[x_j ↦ r]) for a suitable r ∈ ℚ, e.g.
   |  |   r = bnd_j(a_{i,-}x ≤ b_i)
10 |  |  case PARTIAL-UNSAT do continue
11 return PARTIAL-UNSAT
```

Conflicts. We distinguish inconsistencies in $A\boldsymbol{x} \leq \boldsymbol{b}$ by the following notions: We call a row i of $A\boldsymbol{x} \leq \boldsymbol{b}$ a *conflict* if it is of the form $\boldsymbol{a}_{i,\text{-}} = \boldsymbol{0}$ with $b_i < 0$. We call the conflict *global* if $\boldsymbol{f}_{i,\text{-}} \geq 0$ and *local* otherwise. In case of a global conflict, Farkas' Lemma allows to deduce the unsatisfiability of $\widehat{A}\boldsymbol{x} \leq \widehat{\boldsymbol{b}}$, thus stopping the search before the whole search tree is generated. Then a set of conflicting rows K of the input system corresponding to $\boldsymbol{f}_{i,\text{-}}$ is returned. In particular, the set $\{\widehat{\boldsymbol{a}}_{j,\text{-}}\ \boldsymbol{x} \leq \widehat{b}_j \mid f_{i,j} \neq 0\}$ is a minimal unsatisfiable subset of the constraints in $\widehat{A}\boldsymbol{x} \leq \widehat{\boldsymbol{b}}$. In case of a local conflict, the current node is unsatisfiable, but we need to continue the search in the other branches. The algorithm returns *PARTIAL-UNSAT* to indicate that $A\boldsymbol{x} \leq \boldsymbol{b}$ is unsatisfiable, but the unsatisfiability of $\widehat{A}\boldsymbol{x} \leq \widehat{\boldsymbol{b}}$ cannot be derived. This approach, formalized in Algorithm 1, guarantees that the initial call will never return *PARTIAL-UNSAT*; we always find either a global conflict or a solution.

The correctness and completeness of `FMplex` follows from Theorem 3.5 and Theorem 4.2. As an immediate consequence of Theorem 3.5, `FMplex` returns *SAT* if and only if the input is satisfiable. On the other hand, if the input is unsatisfiable, then the initial call to Algorithm 1 will always return *UNSAT* and never *PARTIAL-UNSAT*, as is shown in Theorem 4.2.

Theorem 4.2. *Let $\widehat{A} \in \mathbb{Q}^{\widehat{m}\times n}$, and $\widehat{\boldsymbol{b}} \in \mathbb{Q}^{\widehat{m}\times 1}$. Then $\widehat{A}\boldsymbol{x} \leq \widehat{\boldsymbol{b}}$ is unsatisfiable if and only if the call `FMplex`$(\widehat{A}\boldsymbol{x} \leq \widehat{\boldsymbol{b}})$ to Algorithm 1 terminates with a global conflict.*

Proof. We show the two directions of the equivalence separately:

$\Leftarrow$: Assume the call `FMplex`$(\widehat{A}\boldsymbol{x} \leq \widehat{\boldsymbol{b}}; A\boldsymbol{x} \leq \boldsymbol{b}, F)$ in the recursion tree returns *UNSAT*, i.e. $A\boldsymbol{x} \leq \boldsymbol{b}$ contains a global conflict in some row i. Then $\boldsymbol{a}_{i,\text{-}} = \boldsymbol{f}_{i,\text{-}}\widehat{A} = 0$, $b_i = \boldsymbol{f}_{i,\text{-}} \cdot \widehat{\boldsymbol{b}} < 0$ and $\boldsymbol{f}_{i,\text{-}} \geq 0$, thus Farkas' Lemma (Theorem 2.1) implies that $\widehat{A}\boldsymbol{x} \leq \widehat{\boldsymbol{b}}$ is unsatisfiable.

$\Rightarrow$: Assume $\widehat{A}\boldsymbol{x} \leq \widehat{\boldsymbol{b}}$ is unsatisfiable. Then there is a minimal subset $\widehat{K} \subseteq [\widehat{m}]$ with size $|\widehat{K}| \leq n+1$ such that the corresponding rows $\{\widehat{\boldsymbol{a}}_{\boldsymbol{i},\text{-}}\boldsymbol{x} \leq \widehat{b}_i \mid i \in \widehat{K}\}$ are together unsatisfiable. We show that the algorithm will eventually construct a conflict from such a minimal unsatisfiable subset. Then, we prove that this will be a global conflict, which will prompt the algorithm to return *UNSAT*.

Let $(x_j, V) \in \mathit{branch\_choices}(\widehat{A}\boldsymbol{x} \leq \widehat{\boldsymbol{b}})$ and $i^* \in V$. If $\widehat{K} \subseteq I_j^0(\widehat{A})$, then each of the rows in $\widehat{K}$ are contained also in $P_{j,i^*}(\widehat{A}\boldsymbol{x} \leq \widehat{\boldsymbol{b}})$. Note that since $|\widehat{K}| > 0$, this case cannot always apply until termination (except for the degenerate case where a row contains only zeros).

Otherwise, both $I_j^-(\widehat{A}) \cap \widehat{K} \neq \emptyset$ and $I_j^+(\widehat{A}) \cap \widehat{K} \neq \emptyset$ must hold due to the minimality of $\widehat{K}$ (if only one intersection was non-empty, we could remove it from $\widehat{K}$ to still obtain an infeasible subset). Thus, at some point the algorithm will choose $i^* = i$ for some $i \in \widehat{K}$.

Let $(A\boldsymbol{x} \leq \boldsymbol{b}) = P_{j,i}(\widehat{A}\boldsymbol{x} \leq \widehat{\boldsymbol{b}})$ be the resulting partial projection. Since $\widehat{A}\boldsymbol{x} \leq \widehat{\boldsymbol{b}}$ is unsatisfiable, also $A\boldsymbol{x} \leq \boldsymbol{b}$ is unsatisfiable by Lemma 3.3. With the same reasoning, $A'\boldsymbol{x} \leq \boldsymbol{b'} := P_{j,i}(\widehat{A}[\widehat{K}]\boldsymbol{x} \leq \widehat{\boldsymbol{b}}[\widehat{K}])$ is unsatisfiable. We observe that all rows of $A'\boldsymbol{x} \leq \boldsymbol{b'}$ are contained in $A\boldsymbol{x} \leq \boldsymbol{b}$, thus there exists a minimal set K of rows of $A\boldsymbol{x} \leq \boldsymbol{b}$ that are also contained in $A'\boldsymbol{x} \leq \boldsymbol{b'}$ such that the corresponding rows are a minimal unsatisfiable subset. In particular, we have $0 < |K| < |\widehat{K}|$, because $A'\boldsymbol{x} \leq \boldsymbol{b'}$ has less than $|\widehat{K}|$ rows.

We can repeat this reasoning for the successive elimination of all variables to obtain a system $A\boldsymbol{x} \leq \boldsymbol{b}$ where the corresponding minimal unsatisfiable set has size 1. By construction, there exists a matrix F such that $A = F\widehat{A}$ and $\boldsymbol{b} = F\widehat{\boldsymbol{b}}$. This means that there is a conflicting row i in $A\boldsymbol{x} \leq \boldsymbol{b}$ which is a linear combination of rows from $\widehat{A}[\widehat{K}]\boldsymbol{x} \leq \widehat{\boldsymbol{b}}[\widehat{K}]$.

It remains to show that $\boldsymbol{f}_{i,\text{-}} \geq 0$. By construction, $f_{i,i} > 0$ holds, and for all $i' \in [m] \setminus \widehat{K}$ holds $f_{i,i'} = 0$. Towards a contradiction, assume that there was some $i' \in \widehat{K}$ with $f_{i,i'} < 0$. Since $\widehat{K}$ induces a minimal unsatisfiable subset, by Farkas' Lemma there is some other vector $\boldsymbol{f'} \in \mathbb{Q}^{m \times 1}$ with $\boldsymbol{f'} \geq 0$ such that $(f'_i \neq 0 \Leftrightarrow i \in \widehat{K})$, $\boldsymbol{f'}\widehat{A} = 0$ and $\boldsymbol{f'}\widehat{\boldsymbol{b}} < 0$. Now let

$$\lambda := \max\{\frac{-f_{i,j}}{f'_j} \mid f_{i,j} < 0\} \qquad \text{and} \qquad \boldsymbol{g} := \boldsymbol{f}_{i,\text{-}} + \lambda \boldsymbol{f'}.$$

We get $\boldsymbol{g}\widehat{A} = 0$ and $\boldsymbol{g}\widehat{\boldsymbol{b}} < 0$ because $\lambda > 0$. But now, $\boldsymbol{g} \geq 0$, $g_{i'} = 0$ for all $i' \in [\widehat{m}] \setminus \widehat{K}$ and $g_{i'} = 0$ for some $i' \in \widehat{K}$, which implies that $\widehat{K}$ does not induce a minimal unsatisfiable subset. As this is a contradiction, $\boldsymbol{f}_{i,\text{-}} \geq 0$ must already hold. □

In the next two subsections, we look at ways to prune the recursion tree, eventually leading to Algorithm 2. Additionally, the first improvement is based on insights that reveal a connection to the simplex method.

4.1. **Avoiding Redundant Checks.** We observe that each row in a sub-problem $A\boldsymbol{x} \leq \boldsymbol{b}$ in the recursion tree of $\texttt{FMplex}(\widehat{A}\boldsymbol{x} \leq \widehat{\boldsymbol{b}})$ corresponds to a unique row $\hat{\imath}$ in $\widehat{A}\boldsymbol{x} \leq \widehat{\boldsymbol{b}}$ in the sense that it is a linear combination of the rows $\{\hat{\imath}\} \cup \mathcal{N}$ of $\widehat{A}\boldsymbol{x} \leq \widehat{\boldsymbol{b}}$, where $\mathcal{N} \subseteq [\widehat{m}]$ corresponds to the lower/upper bounds designated as largest/smallest one to compute $A\boldsymbol{x} \leq \boldsymbol{b}$.

The intuition is simple for a single elimination: for a restricted projection $P_{j,i}(\widehat{A}\boldsymbol{x} \leq \widehat{\boldsymbol{b}})$, we have $\mathcal{N} = \{i\}$ containing only the designated bound. Each row $P_{j,i}(\widehat{A}\boldsymbol{x} \leq \widehat{\boldsymbol{b}})$ is a linear combination of the row i and a unique row different from i. This gives a one-to-one correspondence, which we can use to identify the next row that is added to $\mathcal{N}$ in the subsequent elimination.

Theorem 4.3. *Let $\widehat{A} \in \mathbb{Q}^{\widehat{m} \times n}$ and $\widehat{\boldsymbol{b}} \in \mathbb{Q}^{\widehat{m} \times 1}$. Let $\texttt{FMplex}(\widehat{A}\boldsymbol{x} \leq \widehat{\boldsymbol{b}}; A\boldsymbol{x} \leq \boldsymbol{b}, F)$ be a call in the recursion tree of the call $\texttt{FMplex}(\widehat{A}\boldsymbol{x} \leq \widehat{\boldsymbol{b}})$ to Algorithm 1, where $A \in \mathbb{Q}^{m \times n}$ and $\boldsymbol{b} \in \mathbb{Q}^{m \times 1}$ (by construction $m \leq \widehat{m}$).*

Then there exist a set $\mathcal{N} \subseteq [\widehat{m}]$ and an injective mapping $\mathcal{B} : [m] \to [\widehat{m}]$ such that

1. *$A\boldsymbol{x} \leq \boldsymbol{b}$ is satisfiable if and only if $(\widehat{A}\boldsymbol{x} \leq \widehat{\boldsymbol{b}}) \wedge (\widehat{A}[\mathcal{N}]\boldsymbol{x} = \widehat{\boldsymbol{b}}[\mathcal{N}])$ is satisfiable,*
2. *For each $i \in [m]$ holds $\{\mathcal{B}(i)\} = \{i' \in [\hat{m}] \mid f_{i,i'} \neq 0\} \setminus \mathcal{N}$.*

Proof. By assumption, there exists a sequence of calls of the form $\texttt{FMplex}(\widehat{A}\boldsymbol{x} \leq \widehat{\boldsymbol{b}}; \_, \_)$ starting from $\texttt{FMplex}(\widehat{A}\boldsymbol{x} \leq \widehat{\boldsymbol{b}}; \widehat{A}\boldsymbol{x} \leq \widehat{\boldsymbol{b}}, E)$ and ending in $\texttt{FMplex}(\widehat{A}\boldsymbol{x} \leq \widehat{\boldsymbol{b}}; A\boldsymbol{x} \leq \boldsymbol{b}, F)$ such that each of these calls is called by its predecessor in the sequence. We prove the property by induction over this sequence.

In the base case $\texttt{FMplex}(\widehat{A}\boldsymbol{x} \leq \widehat{\boldsymbol{b}}; \widehat{A}\boldsymbol{x} \leq \widehat{\boldsymbol{b}}, E)$, we choose $\mathcal{N} = \emptyset$ and $\mathcal{B}$ as the identity. Then both properties are trivially fulfilled.

Now assume $\texttt{FMplex}(\widehat{A}\boldsymbol{x} \leq \widehat{\boldsymbol{b}}; A'\boldsymbol{x} \leq \boldsymbol{b'}, F')$ for which an $\mathcal{N}'$ and $\mathcal{B}'$ fulfilling the desired properties exist, and let $\texttt{FMplex}(\widehat{A}\boldsymbol{x} \leq \widehat{\boldsymbol{b}}; A\boldsymbol{x} \leq \boldsymbol{b}, F)$ be the succeeding call in the sequence. If $A\boldsymbol{x} \leq \boldsymbol{b} = P_{j,\perp}(A'\boldsymbol{x} \leq \boldsymbol{b'})$, then we simply choose $\mathcal{N} = \mathcal{N}'$ and $\mathcal{B} = \mathcal{B}'$. The first property holds by Lemma 3.3, the second property is trivial.

Otherwise, $(A\boldsymbol{x} \leq \boldsymbol{b}) = P_{j,i}(A'\boldsymbol{x} \leq \boldsymbol{b'})$ for some row i. We choose $\mathcal{N} := \mathcal{N}' \cup \{\mathcal{B}'(i)\}$ and for every row i' of $A\boldsymbol{x} \leq \boldsymbol{b}$, we choose $\mathcal{B}(i') := \mathcal{B}'(k)$ for the unique k such that the row

i' of $A\boldsymbol{x} \leq \boldsymbol{b}$ is the linear combination of the rows i and k of $A'\boldsymbol{x} \leq \boldsymbol{b'}$ (according to the restricted projection). The second property obviously holds. We prove the first property:

$$\begin{aligned} A\boldsymbol{x} \leq \boldsymbol{b} \text{ satisfiable} &\iff (A'\boldsymbol{x} \leq \boldsymbol{b'}) \wedge (\boldsymbol{a'}_{i,\text{-}}\, \boldsymbol{x} = b'_i) \text{ satisfiable} \\ &\iff (\widehat{A}\boldsymbol{x} \leq \widehat{\boldsymbol{b}}) \wedge (\widehat{A}[\mathcal{N}']\boldsymbol{x} = \widehat{\boldsymbol{b}}[\mathcal{N}']) \wedge (\boldsymbol{a'}_{i,\text{-}}\, \boldsymbol{x} = b'_i) \text{ satisfiable} \\ &\iff (\widehat{A}\boldsymbol{x} \leq \widehat{\boldsymbol{b}}) \wedge (\widehat{A}[\mathcal{N}']\boldsymbol{x} = \widehat{\boldsymbol{b}}[\mathcal{N}']) \wedge (\widehat{\boldsymbol{a}}_{\mathcal{B}'(i),\text{-}}\boldsymbol{x} = \widehat{b}_{\mathcal{B}'(i)}) \text{ satisfiable} \\ &\iff (\widehat{A}\boldsymbol{x} \leq \widehat{\boldsymbol{b}}) \wedge (\widehat{A}[\mathcal{N}]\boldsymbol{x} = \widehat{\boldsymbol{b}}[\mathcal{N}]) \text{ satisfiable} \end{aligned}$$

The first equivalence holds due to Lemma 3.3, the second due to the induction hypothesis. □

We call the above defined set $\mathcal{N}$ the *non-basis*, inspired from the analogies to the simplex algorithm (discussed in Section 5.1). By the above theorem, the order in which a non-basis is constructed has no influence on the satisfiability of the induced sub-problem. In particular:

Theorem 4.4. *Let $A \in \mathbb{Q}^{m\times n}$, $\boldsymbol{b} \in \mathbb{Q}^{m\times 1}$, $j \in [n]$, and let $i, i' \in [m]$ with $a_{i,j} \neq 0$ and $a_{i',j} \neq 0$. If $P_{j,i}(A\boldsymbol{x} \leq \boldsymbol{b})$ is unsatisfiable, then $P_{j,i'}(A\boldsymbol{x} \leq \boldsymbol{b}) \wedge (\boldsymbol{a}_{i,\text{-}}\, \boldsymbol{x} = b_i)$ is unsatisfiable.*

Proof. By Theorem 4.3, if $P_{j,i}(A\boldsymbol{x} \leq \boldsymbol{b})$ is unsatisfiable, then $(A\boldsymbol{x} \leq \boldsymbol{b}) \wedge (\boldsymbol{a}_{i,\text{-}}\boldsymbol{x} = b_i)$ is unsatisfiable, and trivially $(A\boldsymbol{x} \leq \boldsymbol{b}) \wedge (\boldsymbol{a}_{i,\text{-}}\boldsymbol{x} = b_i) \wedge (\boldsymbol{a}_{i',\text{-}}\boldsymbol{x} = b_{i'})$ is unsatisfiable as well. Using Theorem 4.3 in the other direction yields that $(A\boldsymbol{x} \leq \boldsymbol{b}) \wedge (\boldsymbol{a}_{i',\text{-}}\boldsymbol{x} = b_{i'})$ is equivalent to $P_{j,i'}(A\boldsymbol{x} \leq \boldsymbol{b})$; thus $P_{j,i'}(A\boldsymbol{x} \leq \boldsymbol{b}) \wedge (\boldsymbol{a}_{i,\text{-}}\boldsymbol{x} = b_i)$ is unsatisfiable. □

Accordingly, if we know that a non-basis $\mathcal{N}$ induces an unsatisfiable sub-problem, then we also know that all supersets of $\mathcal{N}$ will do so. We apply this insight to our algorithm: Assume a call $\texttt{FMplex}(\widehat{A}\boldsymbol{x} \leq \widehat{\boldsymbol{b}}; A\boldsymbol{x} \leq \boldsymbol{b}, F)$, with corresponding $\mathcal{N}$ and $\mathcal{B}$, has a child call for row i which returns *UNSAT* or *PARTIAL-UNSAT*. Then every call in the recursion tree of $\texttt{FMplex}(\widehat{A}\boldsymbol{x} \leq \widehat{\boldsymbol{b}}; A\boldsymbol{x} \leq \boldsymbol{b}, F)$ whose corresponding non-basis contains $\mathcal{B}(i)$ will also return *UNSAT* or *PARTIAL-UNSAT*. Hence, after we checked the child call for row i, we can ignore $\mathcal{B}(i)$ as designated bound in the remaining recursion tree of $\texttt{FMplex}(\widehat{A}\boldsymbol{x} \leq \widehat{\boldsymbol{b}}; A\boldsymbol{x} \leq \boldsymbol{b}, F)$.

Example 4.5. Consider the system from Example 3.1, extended by $c_5 : (-x_2 \leq 0)$, depicted in the top left of the Figure 4 below. That figure also shows a (partial) trace of our algorithm. In front of each row is indicated how it was constructed from the original constraints; $\mathcal{B}$ would give the row corresponding to this combination's first term. Note that we omit the eliminated variables and corresponding zero-columns.

If c_5 is tried first as greatest lower bound on x_2, then the resulting partial projection contains a local conflict. Thus, this branch and, due to Theorem 4.4, any non-basis containing row 5 yields an unsatisfiable system.

Next, we try c_1 as greatest lower bound on x_2, giving the branch on the right. If we now choose $(x_1 \leq 4)$, which corresponds to c_5 via $\mathcal{N}$ and $\mathcal{B}$, as smallest upper bound on x_1, another local conflict occurs (bottom right, row 1). As 5 is contained in the non-basis of that system, we could have predicted this and thus avoid computing this branch.

We update the `FMplex` algorithm as shown in Algorithm 2b using the following definition:

Definition 4.6. The set of *branch choices* for $A\boldsymbol{x} \leq \boldsymbol{b}$ with m rows w.r.t. $I \subseteq [m]$ is

$$\mathit{branch\_choices}(A\boldsymbol{x} \leq \boldsymbol{b}, I) := \{(x_j, V \setminus I) \mid (x_j, V) \in \mathit{branch\_choices}(A\boldsymbol{x} \leq \boldsymbol{b})\}.$$

This modification clearly prevents visiting the same non-basis twice in the following sense:

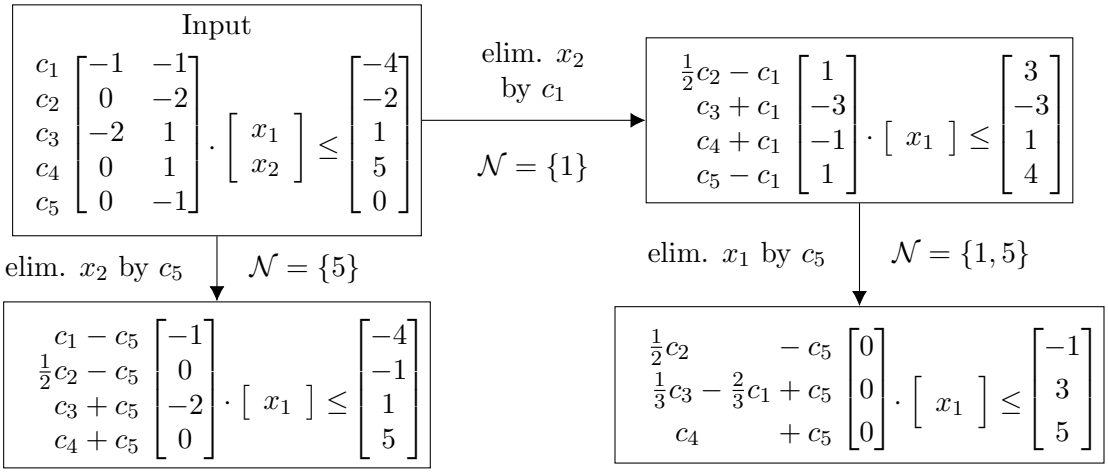


FIGURE 4. Algorithm trace described in Example 4.5.

Theorem 4.7. *Let* $\texttt{FMplex}(\widehat{A}\boldsymbol{x} \leq \widehat{\boldsymbol{b}}; A\boldsymbol{x} \leq \boldsymbol{b}, \_, \mathcal{N}, \_)$ *and* $\texttt{FMplex}(\widehat{A}\boldsymbol{x} \leq \widehat{\boldsymbol{b}}; A'\boldsymbol{x} \leq \boldsymbol{b'}, \_, \mathcal{N}', \_)$ *be two calls in the recursion tree of a call to Algorithm 2b. Then either* $\mathcal{N} \neq \mathcal{N}'$ *or one of the systems occurs in the subtree below the other and only unbounded variables are eliminated between them (i.e. one results from the other by deleting some rows).* □

Theorem 4.8 states that, still, every initial call to Algorithm 2b terminates with *SAT* or a global conflict. This follows by a slight modification of the proof of Theorem 4.2.

Theorem 4.8. *Let* $\widehat{A} \in \mathbb{Q}^{\widehat{m} \times n}$*, and* $\widehat{\boldsymbol{b}} \in \mathbb{Q}^{\widehat{m} \times 1}$*. Then* $\widehat{A}\boldsymbol{x} \leq \widehat{\boldsymbol{b}}$ *is unsatisfiable if and only if the call* $\texttt{FMplex}(\widehat{A}\boldsymbol{x} \leq \widehat{\boldsymbol{b}})$ *to Algorithm 2b terminates with a global conflict.*

Proof. Consider the proof of Theorem 4.2 again: We showed that there exists a path in the recursion tree leading to a global conflict. We now show that we can still construct such a path. We first fix the earliest path in the recursion tree that leads Algorithm 1 to a global conflict, which does exist according to Theorem 4.2; let $\widehat{K} \subseteq [\widehat{m}]$ be the corresponding subset inducing the minimal unsatisfiable subset. Along this path, all but one element of $\widehat{K}$ are chosen into the non-basis, as can be deduced from the proof of Theorem 4.2. Without loss of generality, we can assume that this path does not insert anything else into the non-basis (otherwise, it would not be the earliest global conflict, or there would be variables not occurring in the conflict, whose elimination is irrelevant).

We now execute Algorithm 2b on the same input. Towards a contradiction, we assume that there is a state on the given path where the choice to the next element is forbidden in Algorithm 2b, that is, an element $k \in \widehat{K}$ is ignored ($k \in I$). This is only possible if there has been an earlier call in the execution trace of Algorithm 2b where k was added to the non-basis, this call returned *PARTIAL-UNSAT*, and its parent call is on the given path (otherwise, k would already have been removed from the set of ignored rows). But then, still only elements from $\widehat{K}$ were in the non-basis of that call. But this means, there would be a earlier path that leads to the same global conflict $\widehat{K}$, which is a contradiction to the above assumption. □

Algorithm 2: `FMplex`$(\widehat{A}\boldsymbol{x} \le \widehat{\boldsymbol{b}}; A\boldsymbol{x} \le \boldsymbol{b}, F, \boxed{\mathcal{N}, \mathcal{B}, I}, \boxed{\texttt{lvl}, \texttt{bt\_lvl}})$

Algorithm 2a: The base method consists of the plain parts (as in Algorithm 1).
Algorithm 2b: Consists of the base method and the framed parts.
Algorithm 2c: Consists of the base method, the framed and the filled parts.

Data : $\widehat{A} \in \mathbb{Q}^{\widehat{m}\times n}$, $\widehat{\boldsymbol{b}} \in \mathbb{Q}^{\widehat{m}}$
Input : $A \in \mathbb{Q}^{m\times n}$, $\boldsymbol{b} \in \mathbb{Q}^m$, $F \in \mathbb{Q}^{m\times\widehat{m}}$ s.t. $F\widehat{A} = A$ and $F\widehat{\boldsymbol{b}} = \boldsymbol{b}$,
$\mathcal{N} \subseteq [\widehat{m}]$, $\mathcal{B}\colon [m] \to [\widehat{m}]$, $I \subseteq [\widehat{m}]$,
$\texttt{lvl} \in [n] \cup \{0\}$, $\texttt{bt\_lvl}\colon [m] \to [n] \cup \{0\}$
Output : (SAT, α) with $\alpha \models A\boldsymbol{x} \le \boldsymbol{b}$, or (UNSAT, K) where $K \subseteq [\widehat{m}]$, or
$(\mathit{PARTIAL\text{-}UNSAT}, l, K)$ where $l \in [n]$ and $K \subseteq [\widehat{m}]$

---

if $A = 0 \wedge \boldsymbol{b} \ge 0$ **then return** $(\mathit{SAT}, ())$ `// trivial sat`
if $\exists i \in [m].\ \boldsymbol{a}_{i,\text{-}} = 0 \wedge b_i < 0 \wedge \boldsymbol{f}_{i,\text{-}} \ge 0$ **then** `// global conflict`
  return $(\mathit{UNSAT}, \{i' \mid f_{i,i'} \ne 0\})$
if $\exists i \in [m].\ \boldsymbol{a}_{i,\text{-}} = 0 \wedge b_i < 0 \wedge \boldsymbol{f}_{i,\text{-}} \not\ge 0$ **then** `// local conflict`
  $i := \arg\min_{i\in[m]}\{\texttt{bt\_lvl}(i) \mid \boldsymbol{a}_{i,\text{-}} = 0 \wedge b_i < 0\}$
  return $(\mathit{PARTIAL\text{-}UNSAT}, \texttt{bt\_lvl}(i) - 1, \{i' \mid f_{i,i'} \ne 0\})$
$K := \emptyset$
choose $(x_j, V) \in \mathit{branch\_choices}(A\boldsymbol{x} \le \boldsymbol{b}, \boxed{\{\mathcal{B}^{-1}(i) \mid i \in I\}})$
foreach $i \in V$ **do**
  compute $A'\boldsymbol{x} \le \boldsymbol{b'} := P_{j,i}(A\boldsymbol{x} \le \boldsymbol{b})$ with projection matrix F'
  and backtrack levels $\texttt{bt\_lvl}'$
  compute $\mathcal{N}' := (\mathcal{N} \cup \{\mathcal{B}(i)\}$ **if** $i \ne \perp$ **else** $\mathcal{N})$ and $\mathcal{B}'$ from $\mathcal{N}'$ and F'
  switch `FMplex` $(\widehat{A}\boldsymbol{x} \le \widehat{\boldsymbol{b}}; A'\boldsymbol{x} \le \boldsymbol{b'}, F'F, \boxed{\mathcal{N}', \mathcal{B}', I}, \boxed{\texttt{lvl}+1, \texttt{bt\_lvl}'})$ **do**
    case (UNSAT, K') **do return** (UNSAT, K')
    case (SAT, α) **do return** $(\mathit{SAT}, \alpha[x_j \mapsto r])$ for a suitable $r \in \mathbb{Q}$
    case $(\mathit{PARTIAL\text{-}UNSAT}, l, K')$ **do**
      if $l < \texttt{lvl}$ **then return** $(\mathit{PARTIAL\text{-}UNSAT}, l, K')$ `// backtrack`
      else $K := K \cup K'$
  $I := I \cup \{\mathcal{B}(i)\}$
  if $\texttt{lvl} = 0$ **then return** (UNSAT, K)
return $(\mathit{PARTIAL\text{-}UNSAT}, \texttt{lvl-1}, K)$

4.2. **Backtracking of Local Conflicts.** So far, we ignored local conflicts that witness the unsatisfiability of a given sub-problem. In this section, we will cut off parts of the search tree based on local conflicts and examine the theoretical implications.

We applied Farkas' Lemma on conflicting rows in some sub-problem that are positive linear combinations of rows from the input system. We can also apply Farkas' Lemma to conflicting rows which are positive linear combinations of some *intermediate* system to conclude the unsatisfiability of the latter. Whenever such a conflict occurs, we can backtrack to the parent system of that unsatisfiable system. Instead of tracking the linear combinations of every row in terms of the rows of each preceding intermediate system, we can do an incomplete check: If a conflicting row was computed only by addition operations, then it is a positive linear combination of the involved rows. Thus, we assign to every intermediate system a level, representing its depth in the search tree and store for every row the level where the last subtraction was applied to the row (i.e. a lower (upper) bound was subtracted from another lower (upper) bound). If a row is conflicting, we can conclude that the intermediate system at this level is unsatisfiable, thus we can jump back to its parent.

Assume the current system is $A\boldsymbol{x} \leq \boldsymbol{b}$ at level $\texttt{lvl}$ with m rows whose backtracking levels are stored in $\texttt{bt\_lvl} : [m] \to ([n] \cup \{0\})$. If $\texttt{lvl} = 0$, then $\texttt{bt\_lvl}$ maps all values to 0. When computing $P_{j,i}(A\boldsymbol{x} \leq \boldsymbol{b})$ for some x_j and i with projection matrix F, the backtracking levels of the rows in the resulting system $FA\boldsymbol{x} \leq F\boldsymbol{b}$ are stored in $\texttt{bt\_lvl}'$ where for each row i''

$$\texttt{bt\_lvl}'(i'') := \begin{cases} \max\{\texttt{bt\_lvl}(i), \texttt{bt\_lvl}(i')\} & \text{if } f_{i'',i}, f_{i'',i'} > 0 \text{ and } f_{i'',k} = 0,\ k \notin \{i, i'\} \\ \texttt{lvl} + 1 & \text{otherwise.} \end{cases}$$

The backtracking scheme is given in Algorithm 2c , which returns additional information in the *PARTIAL-UNSAT* case, that is the backtrack level l of the given conflict, and a (possibly non-minimal) unsatisfiable subset K.

Theorem 4.9. *Let* $\texttt{FMplex}(\_; A\boldsymbol{x} \leq \boldsymbol{b}, \_, \_, \_, \texttt{lvl}, \_)$ *be a call to Algorithm 2c , and consider a second call* $\texttt{FMplex}(\_; A'\boldsymbol{x} \leq \boldsymbol{b'}, \_, \_, \_, \_, \texttt{bt\_lvl}')$ *in the recursion tree of the first call. If* $A'\boldsymbol{x} \leq \boldsymbol{b'}$ *has a local conflict in a row* i *with* $\texttt{bt\_lvl}'(i) = \texttt{lvl}$*, then* $A\boldsymbol{x} \leq \boldsymbol{b}$ *is unsatisfiable.*

Proof. By construction of $\texttt{bt\_lvl}'$, $\boldsymbol{a'_{i,\text{-}}}\ \boldsymbol{x} \leq b'_i$ is a positive sum of rows from $A\boldsymbol{x} \leq \boldsymbol{b}$, i.e. there exists an $\boldsymbol{f} \in \mathbb{Q}^{1\times m}$ such that $(\boldsymbol{f}A\boldsymbol{x} \leq \boldsymbol{f}\boldsymbol{b}) = (\boldsymbol{a'_{i,\text{-}}}\ \boldsymbol{x} \leq b'_i)$. Then by Farkas' Lemma, $A\boldsymbol{x} \leq \boldsymbol{b}$ is unsatisfiable. □

While it is complete and correct, Algorithm 2c does not always terminate with a *global* conflict (i.e. Theorem 4.2 does not hold anymore), explaining the need for Line 19:

Example 4.10. We use Algorithm 2c to check the satisfiability of the system depicted in the top left of Figure 5 below. That figure also shows a trace of our algorithm, and to the left of each row is indicated how it was constructed from the original constraints.

We start by eliminating x_3, using the lower bounds given by rows 1 and 2. First, c_1 is tried as greatest lower bound, leading to the branch on the left. Following that path leads to the system in the bottom right, which contains two local conflicts (local, because the linear combinations contain negative coefficients). The first row has backtrack-level 3, as it resulted from combining two lower bounds on level 2. The second row has backtrack-level 1, as it resulted from the system on level 1 by only combining lower with upper bounds. This means that the ancestor call at level 1 is unsatisfiable, and thus we jump back to level 0.

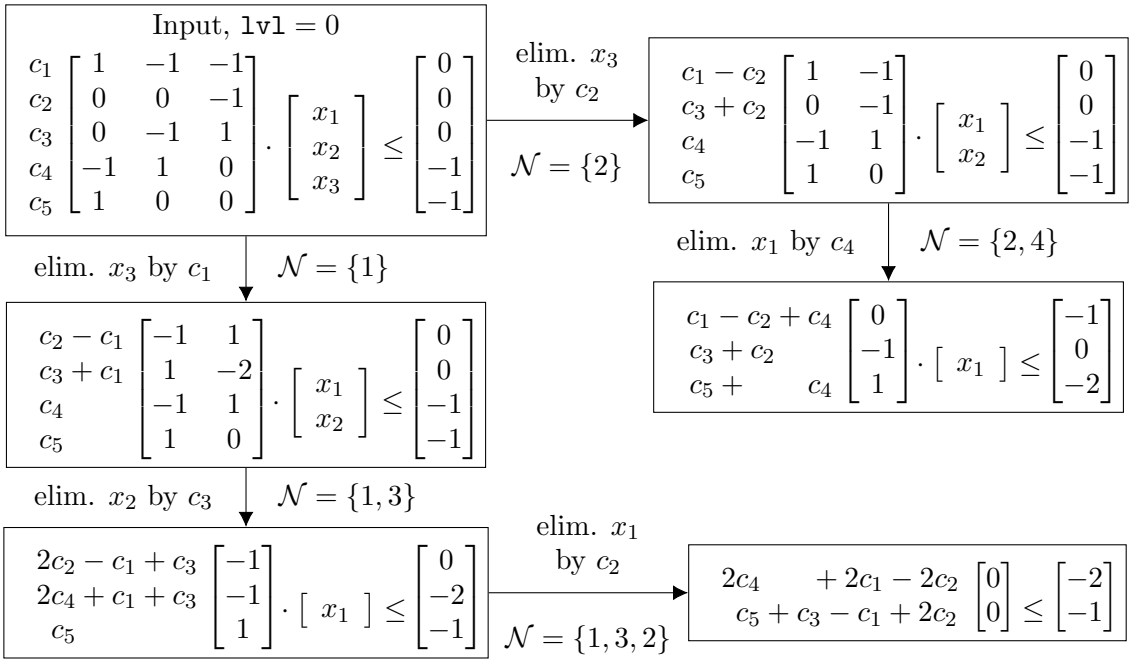


FIGURE 5. Algorithm trace described in Example 4.10.

The second branch for x_3 is visited (on the right), and we now first eliminate x_1 instead of x_2. There is only one lower bound, the row corresponding to c_4, and the resulting child system contains a local conflict, so this branch returns *PARTIAL-UNSAT*. Now, both branches of the initial call returned *PARTIAL-UNSAT*, but no global conflict was found.

Note that the optimization from Section 4.1 is irrelevant in this example; backjumping alone suffices to break Theorem 4.2.

4.3. **Strict Constraints.** So far, we only considered conjunctions of weak linear constraints as input. In practice, however, we often want to admit strict constraints. In the context of SMT solving, strict constraints cannot be avoided, as they appear as the negation of weak constraints which need to be satisfied due to the Boolean structure. For example, an input $A\boldsymbol{x} \leq \boldsymbol{b} \wedge \neg(\boldsymbol{c}\boldsymbol{x} \leq d)$ is equivalent to the system $A\boldsymbol{x} \leq \boldsymbol{b} \wedge (-\boldsymbol{c}\boldsymbol{x} < -d)$.

For the Fourier-Motzkin method, handling strict constraints is straightforward: Upper and lower bounds are combined as before, with the addition that the combination of two constraints c_1 and c_2 is a strict constraint if and only if at least one of c_1, c_2 is strict. The intuition here is that from $(l < x) \wedge (x \leq u)$ follows $(l < u)$, but not necessarily $l \leq u$.

Transferring this idea to `FMplex` is only straightforward for combinations of lower and upper bounds, but we cannot simply apply the same reasoning for combinations of two lower or two upper bounds. The following example shows how this would lead to incorrect results.

Example 4.11. Consider the algorithm trace given below in Figure 6. The first constraint (c_1) is strict. If we therefore assume that the combination of c_1 and c_2 must yield a strict constraint, then we end up with a trivially false constraint $(0 < 0)$ after eliminating both variables x_1 and x_2. This constraint is a positive linear combination of the input and thus the algorithm would report *UNSAT*. This is incorrect, as $x_1 = x_2 = 1$ is indeed a model.

Note that the alleged global conflict $(0 < 0)$ is actually a linear combination of only the weak constraints c_2 and c_3, and the coefficients for the strict constraint c_1 cancel out in the elimination of x_2. But $(0 < 0)$ is not a consequence of c_2 and c_3 and the linear combination actually equates to $(0 \leq 0)$.

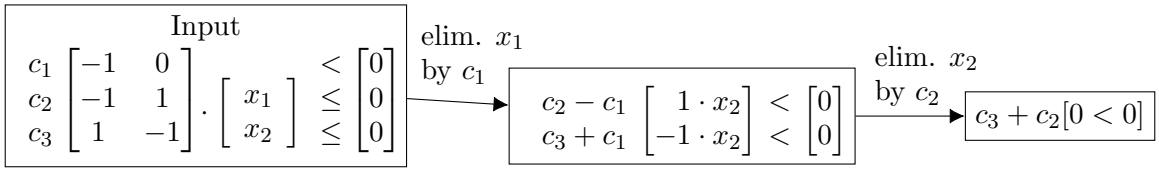


FIGURE 6. Algorithm trace described in Example 4.11.

Instead, we use a well-known way of accomodating strict constraints described e.g. in described in [NÁK21] and [Kin14], which is to transform them into weak constraints by adding a new variable (which we call δ here) representing an infinitesimal positive value.

Lemma 4.12 (Elimination of Strict Constraints)**.** *Let $A \in \mathbb{Q}^{m\times n}$, $\boldsymbol{b} \in \mathbb{Q}^m$, $A' \in \mathbb{Q}^{k\times n}$ and $\boldsymbol{b'} \in \mathbb{Q}^k$ for some $m, k, n \in \mathbb{N}$. Then*

$$\left(A\boldsymbol{x} \leq \boldsymbol{b} \wedge \bigwedge_{i=1}^{k} (\boldsymbol{a'}_{i,\text{-}}\boldsymbol{x} < b'_i)\right) \quad \equiv \quad \exists\delta.\left(A\boldsymbol{x} \leq \boldsymbol{b} \wedge \bigwedge_{i=1}^{k} (\boldsymbol{a'}_{i,\text{-}}\boldsymbol{x} + \delta \leq b'_i) \wedge (\delta > 0)\right). \qquad \square$$

The symbol δ is treated as a variable, but we do not eliminate it in the `FMplex` algorithm. Hence, the single remaining strict constraint $\delta > 0$ does not pose any problems. When `FMplex` reaches a leaf, all other variables have been eliminated, and only constant bounds on δ and trivial constraints are left, making it easy to decide whether that leaf is satisfiable. If it is satisfiable, we can choose any value for δ within the bounds and construct the rest of the model as before.

We can deduce *UNSAT* from global conflicts as before, but we can now also count constraints equivalent to $(\delta \leq 0)$ as global conflicts if they are constructed as a non-negative linear combination of the input system, because they contradict the assertion $(\delta > 0)$.

5. Relation to Other Methods

5.1. **Simplex Algorithm.** The simplex method [Dan98, Lem54] is an algorithm for linear optimization over the reals and is able to solve *linear programs*. The *general simplex* [DDM06] is an adaption for checking the satisfiability of systems of linear constraints. We illustrate its idea for the weak case.

Remind that given a system $A\boldsymbol{x} \leq \boldsymbol{b}$ with m rows, by the fundamental theorem of linear programming (Theorem 2.2), $A\boldsymbol{x} \leq \boldsymbol{b}$ is satisfiable if and only if there exists some maximal subset $\mathcal{N} \subseteq [m]$ such that $A[\mathcal{N}]$ is linearly independent and $A\boldsymbol{x} \leq \boldsymbol{b} \cup A[\mathcal{N}]\boldsymbol{x} = \boldsymbol{b}[\mathcal{N}]$ is satisfiable - the latter can be checked algorithmically using Gaussian elimination, resulting in a system where each variable is replaced by bounds induced by the rows $\mathcal{N}$. This system along with the information which element in $\mathcal{N}$ was used to eliminate which variable is called a *tableau*. The idea of the simplex method is to do a local search on the set $\mathcal{N}$ (called *non-basis*). That is, we replace some $i \in \mathcal{N}$ (*leaving variable*) by some $i' \in [m] \setminus \mathcal{N}$ (*entering*

variable) obtaining $\mathcal{N}' := \mathcal{N} \cup \{i'\} \setminus \{i\}$ such that $A[\mathcal{N}']$ is still linearly independent. The clou is that the tableau representing $(A\boldsymbol{x} \leq \boldsymbol{b}) \wedge (A[\mathcal{N}]\boldsymbol{x} = \boldsymbol{b}[\mathcal{N}])$ can be efficiently transformed into $(A\boldsymbol{x} \leq \boldsymbol{b}) \wedge (A[\mathcal{N}']\boldsymbol{x} = \boldsymbol{b}[\mathcal{N}'])$ (called *pivot operation*), and progress of the local search can be achieved by the choice of i and i'. These local search steps are performed until a satisfying solution is found, or a conflict is found. These conflicts are detected using Farkas' Lemma (Theorem 2.1), i.e. a row in the tableau induces a trivially false constraint and is a positive linear combination of some input rows.

As suggested by Theorem 4.3, there is a strong correspondence between a tableau of the simplex algorithm and the intermediate systems constructed in FMplex. More precisely, if a non-basis of a simplex tableau is equal to the non-basis of a leaf system of Algorithm 1, then the tableau is satisfiable if and only if the FMplex system is satisfiable. In fact, we could use the same data structure to represent the algorithmic states. Comparing the two algorithms structurally, FMplex explores the search space in a tree-like structure using backtracking, while simplex can jump between neighbouring leaves directly.

The idea for Algorithm 2b that excludes visiting the same non-basis in fact arose from the analogies between the two methods. Further, we observe a potential advantage of FMplex: Simplex has more non-bases reachable from a given initial state than the leaves of the search tree of FMplex, as FMplex needs only to explore all lower or all upper bounds of a variable while simplex does local improvements blindly. Heuristically, simplex cuts off large parts of its search space and we expect it often visits fewer non-bases than FMplex - however, as the pruning done by FMplex is by construction of the algorithm, we believe that there might be combinatorially hard instances on which it is more efficient than simplex.

5.2. **Virtual Substitution Method.** *Virtual substitution* [LW93, Wei97] admits quantifier elimination for real arithmetic formulas. Here, we consider its application on existentially quantified conjunctions of linear constraints.

The underlying observation is that the satisfaction of a formula changes at the zeros of its constraints and is invariant between the zeros. Thus, the idea is to collect all *symbolic zeros* $\operatorname{zeros}(\varphi)$ of all constraints in some input formula φ. If all these constraints are weak, then a variable x_j is eliminated by plugging every zero and an arbitrarily small value $-\infty$ into the formula, i.e. $\exists x_j.\ \varphi$ is equivalent to $\varphi[-\infty /\!\!/ x_j] \vee \bigvee_{\xi \in \operatorname{zeros}(\varphi)} \varphi[\xi /\!\!/ x_j]$. The formula $\varphi[t /\!\!/ x_j]$ encodes the semantics of substituting the term t for x_j into the formula φ (which is a disjunction of conjunctions). As we can pull existential quantifiers into disjunctions, we can iteratively eliminate multiple variables by handling each case separately.

The resulting algorithm for quantifier elimination is singly exponential; further optimizations ([Nip08] even proposes to consider only lower or upper bounds for the test candidates) lead to a procedure very similar to the FMplex quantifier elimination: Substituting a test candidate into the formula is equivalent to computing the restricted projection w.r.t. a variable bound. However, our presentation allows to exploit the correspondence with the FM method. Moreover, we handle the special case that there are only lower (resp. upper) bounds separately, while the virtual substitution handles it by always computing $\varphi[-\infty /\!\!/ x_j]$.

Virtual substitution can also be adapted for SMT solving [CÁ11] to a depth-first search similar to FMplex. A conflict-driven search for virtual substitution on conjunctions of weak linear constraints has been introduced in [KKS14], which tracks intermediate constraints as linear combinations of the input constraints similarly to FMplex. Their conflict analysis is a direct generalization of the global conflicts in FMplex and is thus slightly stronger than our

notion of local conflicts. However, their method requires storing and maintaining a lemma database, while FMplex stores all the information for pruning the search tree locally. The approaches have strong similarities, although they originate from quite different methods. Further, our presentation shows the similarities to simplex, is easily adaptable for strict constraints, and naturally extensible to work incrementally.

5.3. **Sample-Based Methods.** There exist several depth-first search approaches, including McMillan et al. [MKS09], Cotton [Cot10] and Korovin et al. [KTV09, KV11], which maintain and adapt a concrete (partial) variable assignment. They share the advantage that combinations of constraints are only computed to guide the assignment away from an encountered conflict, thus avoiding many unnecessary computations, compared to FM.

Similar to FMplex, these methods perform a search with branching, backtracking and learning from conflicting choices. However, they branch on variable assignments, with infinitely many possible choices in each step. Interestingly, the bounds learned from encountered conflicts implicitly partition the search space into a finite number of regions to be tried, similar to what FMplex does explicitly. In fact, we deem it possible that [KTV09] or [KV11] try and exclude assignments from exactly the same regions that FMplex would visit (even in the same order). However, the sample-based perspective offers different possibilities for heuristic improvements than FMplex: choosing the next assigned value vs. choosing the next lower bound; deriving constant variable bounds vs. structural exploits using Farkas' Lemma; potential for rapid solutions vs. control and knowledge about the possible choices.

Moreover, these methods offer no straight-forward adaption for quantifier elimination, while FMplex does. However, [MKS09] and [Cot10] can handle not only conjunctions, but any quantifier-free LRA formula in conjunctive normal form.

6. Experimental Evaluation

We implemented several heuristic variants of the FMplex algorithm, as well as the generalized *simplex* and the *FM* methods as non-incremental DPLL(T) theory backends in our SMT-RAT solver [CKJ+15] and compared their performance in the context of satisfiability checking. Using the transformation given in [NÁK21] and case splitting as in [BNOT06], we extended the method to also handle strict and not-equal-constraints.

The base version of FMplex, Algorithm 1, was tested with two different heuristics for the choice of the eliminated variable and for the order in which the branches are checked. These choices may strongly influence the size of the explored search tree; in the best case, the very first path leads to a satisfiable leaf or to a global conflict.

Min-Fanout. We greedily minimize the number of children: for any $A\boldsymbol{x} \leq \boldsymbol{b}$ and I, we choose $(x_j, V) \in \mathit{branch\_choices}(A\boldsymbol{x} \leq \boldsymbol{b}, I)$ such that $|V|$ is minimal; in case that this minimum is 1, we prefer choices with $V = \{\bot\}$ over the other choices.

We prefer rows with a lower (earlier) backtrack level, motivated by finding a global conflict through trying positive linear combinations first. Moreover, if backtracking is used then we expect this heuristic to allow for backtracking further back on average.

Min-Column-Length. A state-of-the-art heuristic for simplex in the context of SMT solving is the *minimum column length* [KBD13]: we choose the variables for leaving and entering the non-basis such that the number of necessary row operations is minimized. We resemble this heuristic in FMplex as follows: we prefer choices $(x_j, \{\bot\})$ and if there is no such j, we take the $j \in [n]$ with minimal $|I_j^-(A)| + |I_j^+(A)|$ and take the smallest choice between $I_j^-(A)$ and $I_j^+(A)$.

We first choose the rows which have the least non-zero coefficients (i.e. contain the least variables) to prefer sparse sub-problems. This can be understood as *Min-Row-Length*.

We consider the following solver variants: `FMplex-a-MFO` and `FMplex-a-MCL` implement Algorithm 2a (i.e. Algorithm 1) with the Min-Fanout and the Min-Column-Length heuristic, respectively. `FMplex-a-Rand-1/2` denotes two variants of Algorithm 2a where all choices are taken pseudo-randomly with different seeds. `FMplex-b-MFO` implements Algorithm 2b and `FMplex-c-MFO` implements Algorithm 2c , both using the Min-Fanout heuristic. Our approach is also compared to non-incremental implementations `FM` and `Simplex`. The FMplex variants and `FM` always first employ Gaussian elimination to handle equalities.

All solvers were tested on the SMT-LIB [BFT16] benchmark set for QF_LRA containing 1753 formulas. As all evaluated solvers are non-incremental, we also generated conjunctions of constraints by solving each of these QF_LRA problems using a DPLL(T) SMT solver with an `FMplex-c-MFO` theory solver backend, and extracting all conjunctions passed to it. If the solver terminated within the time and memory limits, we sampled 10 satisfiable and 10 unsatisfiable conjunctions (or gathered all produced conjunctions if there were fewer than 10). This amounted to 3084 (777 sat, 2307 unsat) additional benchmarks. The experiments were conducted on identical machines with two Intel Xeon Platinum 8160 CPUs (2.1 GHz, 24 cores). For each formula, the time and memory were limited to 10 minutes and 5 GB.

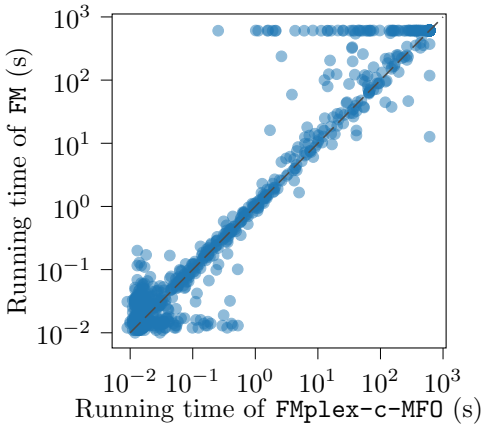
The results in Table 1 show that `Simplex` solved the most SMT-LIB instances, followed by our `FMplex-c-MFO` and then `FM`. Interestingly, `FM` solves fewer conjunctive instances than the base version of FMplex due to higher memory consumption (43 memory-outs for `FM`, while the others have none). We see that a reasonable variable heuristic makes a difference as `FMplex-a-Rand-*` perform much worse than `FMplex-a-MFO` and `FMplex-a-MCL`. However,

	SMT-LIB					Conjunctions				
	solved	sat	unsat	TO	MO	solved	sat	unsat	TO	MO
`Simplex`	**958**	**527**	**431**	714	81	**3084**	**777**	**2307**	0	0
`FM`	860	461	399	577	316	2934	747	2187	107	43
`FMplex-a-MFO`	814	432	382	840	99	2962	743	2219	122	0
`FMplex-a-MCL`	820	435	385	830	103	2965	742	2223	119	0
`FMplex-a-Rand-1`	742	383	359	906	105	2806	668	2138	278	0
`FMplex-a-Rand-2`	743	383	360	905	105	2823	671	2152	261	0
`FMplex-b-MFO`	822	434	388	830	101	2988	744	2244	96	0
`FMplex-c-MFO`	920	499	421	733	100	**3084**	**777**	**2307**	0	0
`Virtual-Best`	982	532	450	651	120	3084	777	2307	0	0

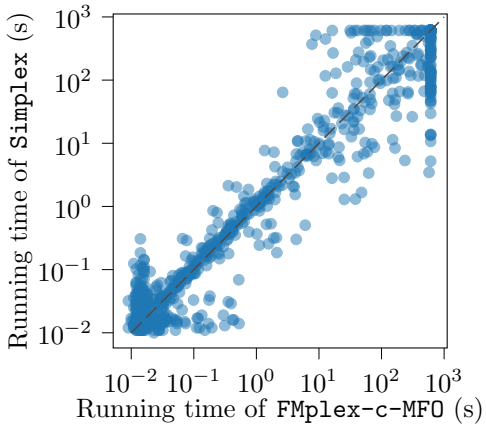
TABLE 1. Number of solved instances (total and differentiated into satisfiable and unsatisfiable), timeouts (TO) and memory-outs (MO).

between the latter two, there is no significant difference. While our first optimization used in `FMplex-b-MFO` has no big impact, the backtracking implemented in `FMplex-c-MFO` allows for solving more instances within the given resource limits.

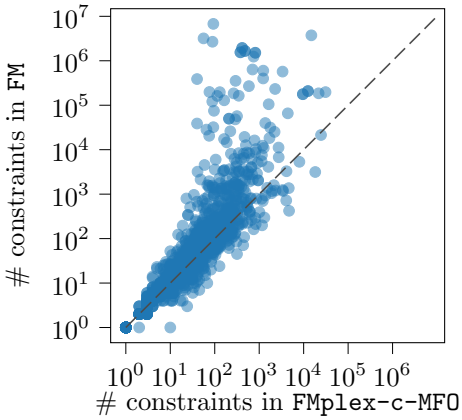
The running times for each individual SMT-LIB instance depicted in Figure 7a and Figure 7b reveal that `FM` and `FMplex-c-MFO` often behave similar, but `FM` fails on a number of larger instances. We suspect that the smaller intermediate systems of FMplex are a main factor here. While `Simplex` is often faster than `FMplex-c-MFO` and solves 61 SMT-LIB instances not solved by `FMplex-c-MFO`, it fails to solve 23 instances on which `FMplex-c-MFO` succeeds (Of these instances, `FM` solves 3 respectively 14 instances). Accordingly, the `Virtual-Best` of the tested solvers performs significantly better than just `Simplex`, indicating potential for a combination of `Simplex` and `FMplex-c-MFO`.



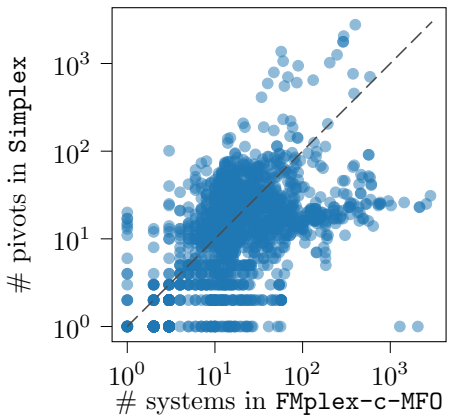
(A) Running times in seconds on the SMT-LIB benchmark set, `FMplex` vs `FM`.



(B) Running times in seconds on the SMT-LIB benchmark set, `FMplex` vs `Simplex`.



(C) Number of generated constraints on the conjunctive benchmark set.



(D) Number of visited non-bases (intermediate systems) on the conjunctive benchmark set.

Figure 7. Scatter plots: Each dot represents a single instance. In (A) and (B), instances at the very top or right exceeded the resource limit. Such instances are not considered in (C) and (D).

Figure 7c compares the number of constraints generated by `FM` and `FMplex-c-MFO` on the conjunctive inputs. Especially on larger instances, FMplex seems to be in the advantage. Motivated by Section 4.1, Figure 7d compares the number of `Simplex` pivots to the number of systems in `FMplex-c-MFO`. We see that neither is consistently lower than the other, though `Simplex` seems to be slightly superior. Due to the log-log scale, not shown are 1305 instances in which either measurement is 0 (920 instances for `Simplex`, 981 for `FMplex-c-MFO`).
The implementation and collected data are available at `https://doi.org/10.5281/zenodo.7755862`.

7. Conclusion

We introduced a novel method *FMplex* for quantifier elimination and satisfiability checking for conjunctions of linear real arithmetic constraints. Structural observations based on Farkas' Lemma and the Fundamental Theorem of Linear Programming allowed us to prune the elimination or the search tree. Although the new method is rooted in the FM method, it has strong similarities with both the virtual substitution method and the simplex method.

The experimental results in the context of SMT solving show that FMplex is faster than Fourier-Motzkin and, although simplex is able to solve more instances than FMplex, there is a good amount of instances which can be solved by FMplex but cannot be solved by simplex. While the current state of FMplex cannot compete with established solvers using the simplex method, these solvers benefit from implementations highly optimized through decades of research.

Thus, in future work we aim to combine the structural savings of FMplex with the efficient heuristic of simplex, i.e. we transfer ideas from FMplex to simplex and vice-versa. Furthermore, we will investigate in tweaks and heuristics. For instance, we plan to adapt the perfect elimination ordering from [LXZZ21] and work on an incremental adaption for SMT solving.

In the work [PÁ24] following up our first paper [NPÁK23] on FMplex, we show, in the context of quantifier elimination, how the resulting disjunction computed by FMplex may be reduced to a single conjunction. This process can be done with practically no overhead, solely based on the coefficients of the linear combinations (the matrix F in the algorithm). This overcomes the main drawback in the comparison to the Fourier-Motzkin method. Thus, FMplex may be used as replacement for Fourier-Motzkin, and is still singly exponential and relatively straight-forward to implement. Moreover, [PÁ24] contains an experimental evaluation in the context of quantifier elimination, and there, FMplex outperforms several established tools. We plan to also pursue this direction further.

References

[BFT16] Clark Barrett, Pascal Fontaine, and Cesare Tinelli. The Satisfiability Modulo Theories Library (SMT-LIB). `www.SMT-LIB.org`, 2016.

[BNOT06] Clark Barrett, Robert Nieuwenhuis, Albert Oliveras, and Cesare Tinelli. Splitting on demand in SAT modulo theories. In *International Conference on Logic for Programming Artificial Intelligence and Reasoning (LPAR'06)*, pages 512–526. Springer, 2006. `doi:10.1007/11916277_35`.

[CÁ11] Florian Corzilius and Erika Ábrahám. Virtual substitution for SMT-solving. In *International Symposium on Fundamentals of Computation Theory (FCT'11)*, pages 360–371. Springer, 2011. `doi:10.1007/978-3-642-22953-4_31`.

[CKJ+15] Florian Corzilius, Gereon Kremer, Sebastian Junges, Stefan Schupp, and Erika Ábrahám. SMT-RAT: An open source C++ toolbox for strategic and parallel SMT solving. In *International Conference on Theory and Applications of Satisfiability Testing (SAT'15)*, pages 360–368. Springer, 2015. doi:10.1007/978-3-319-24318-4_26.

[Cot10] Scott Cotton. Natural domain SMT: A preliminary assessment. In *International Conference on Formal Modeling and Analysis of Timed Systems (FORMATS'10)*, pages 77–91. Springer, 2010. doi:10.1007/978-3-642-15297-9_8.

[Dan98] George Bernard Dantzig. *Linear Programming and Extensions*, volume 48. Princeton University Press, 1998. doi:10.1515/9781400884179.

[DDM06] Bruno Dutertre and Leonardo De Moura. Integrating Simplex with DPLL(T). *Computer Science Laboratory, SRI International, Tech. Rep. SRI-CSL-06-01*, 2006.

[Far02] Julius Farkas. Theorie der einfachen Ungleichungen. *Journal für die reine und angewandte Mathematik (Crelles Journal)*, 1902(124):1–27, 1902. doi:10.1515/crll.1902.124.1.

[Fou27] Jean Baptiste Joseph Fourier. Analyse des travaux de l'académie royale des sciences pendant l'année 1824. *Partie mathématique*, 1827.

[Imb90] Jean-Louis Imbert. About redundant inequalities generated by Fourier's algorithm. In *International Conference on Artificial Intelligence: Methodology, Systems, Applications (AIMSA'90)*, pages 117–127. Elsevier, 1990. doi:10.1016/B978-0-444-88771-9.50019-2.

[Imb93] Jean-Louis Imbert. Fourier's elimination: Which to choose? In *International Conference on Principles and Practice of Constraint Programming (PPCP'93)*, volume 1, pages 117–129. Citeseer, 1993.

[JMMT20] Rui-Juan Jing, Marc Moreno-Maza, and Delaram Talaashrafi. Complexity estimates for Fourier-Motzkin elimination. In *International Workshop on Computer Algebra in Scientific Computing (CASC'20)*, pages 282–306. Springer, 2020. doi:10.1007/978-3-030-60026-6_16.

[KBD13] Tim King, Clark Barrett, and Bruno Dutertre. Simplex with sum of infeasibilities for SMT. In *International Conference on Formal Methods in Computer-Aided Design (FMCAD'13)*, pages 189–196, 2013. doi:10.1109/FMCAD.2013.6679409.

[Kha80] Leonid Genrikhovich Khachiyan. Polynomial algorithms in linear programming. *USSR Computational Mathematics and Mathematical Physics*, 20(1):53–72, 1980. doi:10.1016/0041-5553(80)90061-0.

[Kin14] Tim King. *Effective algorithms for the satisfiability of quantifier-free formulas over linear real and integer arithmetic*. PhD Thesis, New York University, 2014.

[KKS14] Konstantin Korovin, Marek Kosta, and Thomas Sturm. Towards conflict-driven learning for virtual substitution. In *International Workshop on Computer Algebra in Scientific Computing (CASC'14)*, pages 256–270. Springer, 2014. doi:10.1007/978-3-319-10515-4_19.

[KTV09] Konstantin Korovin, Nestan Tsiskaridze, and Andrei Voronkov. Conflict resolution. In *International Conference on Principles and Practice of Constraint Programming (CP'09)*, pages 509–523. Springer, 2009. doi:10.1007/978-3-642-04244-7_41.

[KV11] Konstantin Korovin and Andrei Voronkov. Solving systems of linear inequalities by bound propagation. In *International Conference on Automated Deduction (CADE'23)*, pages 369–383. Springer, 2011. doi:10.1007/978-3-642-22438-6_28.

[Lem54] Carlton E. Lemke. The dual method of solving the linear programming problem. *Naval Research Logistics Quarterly*, 1(1):36–47, 1954. doi:10.1002/nav.3800010107.

[LW93] Rüdiger Loos and Volker Weispfenning. Applying linear quantifier elimination. *The Computer Journal*, 36(5):450–462, 1993. doi:10.1093/comjnl/36.5.450.

[LXZZ21] Haokun Li, Bican Xia, Huiying Zhang, and Tao Zheng. Choosing the variable ordering for cylindrical algebraic decomposition via exploiting chordal structure. In *International Symposium on Symbolic and Algebraic Computation (ISSAC'21)*, pages 281–288. ACM, 2021. doi:10.1145/3452143.3465520.

[LY+84] David G Luenberger, Yinyu Ye, et al. *Linear and Nonlinear Programming*, volume 2nd edition. Springer, 1984. doi:10.1007/978-3-030-85450-8.

[MKS09] Kenneth L. McMillan, Andreas Kuehlmann, and Mooly Sagiv. Generalizing DPLL to richer logics. In *International Conference on Computer Aided Verification (CAV'09)*, pages 462–476. Springer, 2009. doi:10.1007/978-3-642-02658-4_35.

[Mot36] Theodore Samuel Motzkin. *Beiträge zur Theorie der linearen Ungleichungen*. Azriel, 1936.

[NÁK21] Jasper Nalbach, Erika Ábrahám, and Gereon Kremer. Extending the fundamental theorem of linear programming for strict inequalities. In *International Symposium on Symbolic and Algebraic Computation (ISSAC'21)*, pages 313–320. ACM, 2021. doi:10.1145/3452143.3465538.

[Nip08] Tobias Nipkow. Linear quantifier elimination. In *Internation Joint Conference on Automated Reasoning (IJCAR'08)*, pages 18–33. Springer, 2008. doi:10.1007/978-3-540-71070-7_3.

[NPÁK23] Jasper Nalbach, Valentin Promies, Erika Ábrahám, and Paul Kobialka. FMplex: A novel method for solving linear real arithmetic problems. In *International Symposium on Games, Automata, Logics, and Formal Verification (GandALF'23)*, volume 390 of *EPTCS*, pages 16–32, 2023. doi:10.4204/EPTCS.390.2.

[PÁ24] Valentin Promies and Erika Ábrahám. A divide-and-conquer approach to variable elimination in linear real arithmetic. In André Platzer, Kristin Yvonne Rozier, Matteo Pradella, and Matteo Rossi, editors, *International Symposium on Formal Methods (FM'24)*, volume 14933 of *LNCS*, pages 131–148. Springer, 2024. doi:10.1007/978-3-031-71162-6_7.

[Wei97] Volker Weispfenning. Quantifier elimination for real algebra — The quadratic case and beyond. *Applicable Algebra in Engineering, Communication and Computing*, 8(2):85–101, 1997. doi:10.1007/s002000050055.