FMplex: A Novel Method for Solving Linear Real Arithmetic Problems

Jasper Nalbach*
RWTH Aachen University, Germany
nalbach@cs.rwth-aachen.de

Valentin Promies
RWTH Aachen University, Germany
promies@cs.rwth-aachen.de

Erika Ábrahám
RWTH Aachen University, Germany
abraham@cs.rwth-aachen.de

Paul Kobialka
University of Oslo, Norway
paulkob@ifi.uio.no

In this paper we introduce a novel 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.

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*. The problem is known to be solvable in *polynomial* worst-case complexity but, surprisingly, the *ellipsoid* method [13] proposed in 1980 by Khachiyan is still the only available algorithm that implements this bound. However, this method is seldomly used in practice due to its high average-case effort. Instead, most approaches employ the *simplex* algorithm introduced by Dantzig in 1947, which has a *singly exponential* worst case complexity, but which is quite efficient in practice. A third available solution is the *Fourier-Motzkin variable elimination (FM)* method, proposed in 1827 by Fourier [9] and re-discovered in 1936 by Motzkin [23]. 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 recognizing and avoiding redundant computations (e.g. [11, 12]).

In this paper, we propose 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. More precisely, our contributions include:

- 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 it 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.

*Jasper Nalbach was supported by the DFG RTG 2236 UnRAVeL.

A. Achilleos and D. Della Monica (Eds.): Fourteenth International Symposium on Games, Automata, Logics, and Formal Verification (GandALF 2023).
EPTCS 390, 2023, pp. 16–32, doi:10.4204/EPTCS.390.2

- A theorem formalizing connections between FMplex and the simplex algorithm.
- An implementation of the SMT adaptation and its experimental evaluation.

After recalling necessary preliminaries in Section 2, we introduce our novel FMplex method first for quantifier 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.

An extended version of this paper including more detailed proofs can be found on arXiv [25].

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 For $m \in \mathbb{N}$ let $E^{(m)} \in \mathbb{Q}^{m\times m}$ be the identity matrix, and $\boldsymbol{0}^{(m)} = (0 \ \cdots \ 0)^T \in \mathbb{Q}^{m\times 1}$. The ith 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 ith row respectively column vector of A. Furthermore, $A[I]$ denotes the sub-matrix of A containing only the rows with indices from some $I \subseteq [m]$. 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 size of a maximal $I \subseteq [m]$ with $A[I]$ linearly independent.

Linear Constraints Let $\boldsymbol{a} = (a_1,\dots,a_n) \in \mathbb{Q}^{1\times n}$, $b \in \mathbb{Q}$ and $\sim \in \{=,\leq,<,\neq\}$ a *relation symbol*. We call $\boldsymbol{ax}$ a *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. For most 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 redundant rows in contrast to the set notation. As the latter will play a role later on, we will stick to the matrix representation.

Variable Assignment An *assignment* is a function $\alpha : Y \to \mathbb{R}$ with domain $dom(\alpha) = Y \subseteq X$. 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 $\alpha(a_1x_1 + \dots a_nx_n) \sim b$ 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 also make use of the following two well-known results.

Theorem 1 (Farkas' Lemma [8])**.** *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 $\boldsymbol{f}A = (0,\dots,0) \in \mathbb{Q}^{1\times n}$ it holds $\boldsymbol{f}\boldsymbol{b} \geq 0$.*

Theorem 2 (Fundamental Theorem of Linear Programming, as in [21])**.** *Let $A \in \mathbb{Q}^{m\times n}$ 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 $\alpha \models (A[I]\boldsymbol{x} = \boldsymbol{b}[I])$ and $\alpha \models (A\boldsymbol{x} \leq \boldsymbol{b})$.*

2.1 Fourier-Motzkin Variable Elimination

The *Fourier-Motzkin variable elimination* (FM) [9, 23] method allows to eliminate any $x_j \in X$ from a system $A\boldsymbol{x} \leq \boldsymbol{b}$ by computing $A'\boldsymbol{x} \leq \boldsymbol{b'}$ 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 $X \setminus \{x_j\}$.

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, denoted in both cases as $bnd_j(\boldsymbol{a}_{i,\text{-}}\,\boldsymbol{x} \leq b_i)$:

$$\Big(\sum_{k\in[n]\setminus\{j\}} -\frac{a_{i,k}}{a_{i,j}}\cdot x_k\Big) + \frac{b_i}{a_{i,j}} \leq x_j, \;\text{ if } a_{i,j} < 0, \qquad \text{resp.} \qquad x_j \leq \Big(\sum_{k\in[n]\setminus\{j\}} -\frac{a_{i,k}}{a_{i,j}}\cdot x_k\Big) + \frac{b_i}{a_{i,j}}, \;\text{ if } a_{i,j} > 0.$$

Definition 1. *For $A \in \mathbb{Q}^{m\times n}$, we define the index sets*

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

$I_j^-(A)$, $I_j^+(A)$ 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, since in general the involved bounds are symbolic and thus their values depend on the values of other variables, we cannot directly check this condition. To express this, we let $A'\boldsymbol{x} \leq \boldsymbol{b'}$ be defined by the constraint set

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

In matrix representation, the FM method applies the following transformation:

Definition 2 (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:*[1]

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

Then the Fourier-Motzkin variable elimination $\texttt{FM}_j(A\boldsymbol{x} \leq \boldsymbol{b})$ *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 consistency of $A\boldsymbol{x} \leq \boldsymbol{b}$ can be checked by successively eliminating variables $x_n, \ldots, x_1$, obtaining intermediate systems $A^{(n-1)}\boldsymbol{x} \leq \boldsymbol{b}^{(n-1)}, \ldots, A^{(0)}\boldsymbol{x} \leq \boldsymbol{b}^{(0)}$. 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 $0 \leq \boldsymbol{f} \in \mathbb{Q}^{m\times 1}$ s.t. $\boldsymbol{f}A = \boldsymbol{a}^{(k)}_{i',\text{-}}$ and $\boldsymbol{f}\boldsymbol{b} = b^{(k)}_{i'}$, or in short: $\sum_{i\in[m]} f_i \cdot (\boldsymbol{a}_{i,\text{-}}\,\boldsymbol{x} \leq b_i) = (\boldsymbol{a}^{(k)}_{i',\text{-}}\boldsymbol{x} \leq b^{(k)}_{i'})$. We call this kind of linear combinations *conical combinations*. By Farkas' Lemma (Theorem 1), if $A^{(0)}\boldsymbol{x} \leq \boldsymbol{b}^{(0)}$ is unsatisfiable, then so is $A\boldsymbol{x} \leq \boldsymbol{b}$. If it 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 complexity in time and space w.r.t. the number of eliminated variables. Moreover, many of the generated rows are redundant because they are linear combinations of the other rows, i.e. they could be omitted without changing the solution set of the system. Redundancies might already be contained in the input system, or they arise during the projection operation. While removing all redundancies is expensive, there are efficient methods for removing some redundancies of the latter type, for example Imbert's acceleration theorems [10, 11, 12].

[1] Remember that we use lower case letters for rows of matrices with the respective upper case letter as name. Thus, $\boldsymbol{e}^{(m)}_{i,\text{-}}$ denotes the ith column vector of the identity matrix $E^{(m)}$.

Lemma 1 (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 ith row of $A'\boldsymbol{x} \leq \boldsymbol{b'}$ is a conical combination $\boldsymbol{r}FA\boldsymbol{x} \leq \boldsymbol{r}F\boldsymbol{b}$ of the other rows), then that row is redundant in $A'\boldsymbol{x} \leq \boldsymbol{b'}$, i.e. the solution set does not change when omitting it: $sol(A'\boldsymbol{x} \leq \boldsymbol{b'}) = sol(A'[[m']\setminus\{i\}]\boldsymbol{x} \leq \boldsymbol{b'}[[m']\setminus\{i\}])$.*

3 FMplex as Variable Elimination Procedure

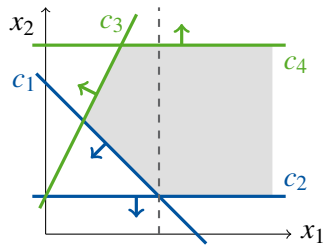
The FM method encodes that none of the lower bounds on some variable x_j in a system $A\boldsymbol{x} \leq \boldsymbol{b}$ is larger than any of its upper bounds. In our *FMplex* method, instead of considering all lower-upper bound combinations at once, we *split the problem into a set of sub-problems* by case distinction either on *which of the lower bounds is the largest* or alternatively on *which of the upper bounds is the smallest*. For splitting on lower bounds, for each lower bound on x_j we consider solutions where this lower bound is maximal under all lower bounds, and at the same time not larger than any of the upper bounds. The upper bound case is analogous. Then $A\boldsymbol{x} \leq \boldsymbol{b}$ is satisfiable if and only if there exists a solution in one of these sub-problems. 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 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

$$\begin{aligned}&\{bnd_j(\boldsymbol{a}_{i',\text{-}}\,\boldsymbol{x} \leq b_{i'}) \leq bnd_j(\boldsymbol{a}_{i,\text{-}}\,\boldsymbol{x} \leq b_i) \mid i' \in I_j^-(A),\ i' \neq i\}\\ \cup&\{bnd_j(\boldsymbol{a}_{i,\text{-}}\,\boldsymbol{x} \leq b_i) \leq bnd_j(\boldsymbol{a}_{i',\text{-}}\,\boldsymbol{x} \leq b_{i'}) \mid i' \in I_j^+(A)\} \cup \{\boldsymbol{a}_{i',\text{-}}\,\boldsymbol{x} \leq b_{i'} \mid i' \in I_j^0(A)\}.\end{aligned}$$

Example 1. *We eliminate x_2 from the system $A\boldsymbol{x} \leq \boldsymbol{b}$ 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. 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.*

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



The encoding of the c_1-case is given by $(bnd_2(c_2) \leq bnd_2(c_1)) \wedge (bnd_2(c_1) \leq bnd_2(c_3)) \wedge (bnd_2(c_1) \leq bnd_2(c_4))$, which evaluates to $(x_1 \leq 3) \wedge (-3x_1 \leq -3) \wedge (-x_1 \leq 1)$ satisfied by any $x_1 \in [1,3]$, on the left of the dashed line. The case for c_2 evaluates to $(-x_1 \leq -3) \wedge (-2x_1 \leq 0) \wedge (0 \leq 4)$ 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.

The construction for the case $i \in I_j^+(A)$ designating i as smallest upper bound is analogous. In matrix representation, these projections are defined by the following transformation:

Definition 3 (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:*

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

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

- *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 $\boldsymbol{e}^{(m)}_{i',\text{-}}$ for each $i' \in I_j^0(A)$, and define $P_{j,\perp}(A\boldsymbol{x} \leq \boldsymbol{b})$ as $FA\boldsymbol{x} \leq F\boldsymbol{b}$.*

The following lemma states a crucial result for our method: 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.

Lemma 2. *Let $A \in \mathbb{Q}^{m\times n}$, $\boldsymbol{b} \in \mathbb{Q}^{m\times 1}$, $j \in [n]$ and $I \in \{I_j^-(A), I_j^+(A)\}$. If $I_j^-(A) \neq \emptyset$ and $I_j^+(A) \neq \emptyset$, then*

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

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. Assume $I = I_j^-(A)$, the case $I = I_j^+(A)$ 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 $\alpha \models A\boldsymbol{x} \leq \boldsymbol{b}$ and $i = \arg\max_{\ell\in I_j^-(A)}(\alpha(bnd_j(\boldsymbol{a}_{\ell,\text{-}}\, \boldsymbol{x} \leq b_\ell)))$, then for all $u \in I_j^+(A)$ it holds $\alpha(bnd_j(\boldsymbol{a}_{i,\text{-}}\, \boldsymbol{x} \leq b_i)) \leq \alpha(bnd_j(\boldsymbol{a}_{u,\text{-}}\, \boldsymbol{x} \leq b_u))$ and thus $\alpha \models P_{j,i}(A\boldsymbol{x} \leq \boldsymbol{b})$. □

Definition 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 2. Lifting this from sets to logic naturally results in the following theorem which demonstrates the usage of our method.

Theorem 3. *Let $A \in \mathbb{Q}^{m\times n}$, $\boldsymbol{b} \in \mathbb{Q}^{m\times 1}$, and $j \in [n]$. Then*

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

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 variant independently in every branch.

Example 2. *We continue Example 1, from which we eliminated x_2 and now want to eliminate x_1:*

$$\begin{aligned}\exists x_1.\,\exists x_2.\,A\boldsymbol{x}\le\boldsymbol{b} \;&\equiv\; \exists x_1.\bigvee\nolimits_{S\in\mathtt{FMP}_2^-(A\boldsymbol{x}\le\boldsymbol{b})} S\\ &\equiv\; \exists x_1.\;(x_1\le 3\wedge -3x_1\le -3\wedge -x_1\le 1)\;\vee\exists x_1.\;(-x_1\le -3\wedge -2x_1\le 0\wedge 0\le 4)\end{aligned}$$

We eliminate the two quantifiers for x_1 separately, using

$$\begin{aligned}\mathtt{FMP}_1^-(x_1\le 3\wedge -3x_1\le -3\wedge -x_1\le 1) &= \{(0\le 2\wedge 0\le 2),(0\le -2\wedge 0\le 4)\}\ \textit{and}\\ \mathtt{FMP}_1^-(-x_1\le -3\wedge -2x_1\le 0\wedge 0\le 4) &= \{(0\le 4)\}\end{aligned}$$

giving us the final result $\exists x_1.\,\exists x_2.\,A\boldsymbol{x}\le\boldsymbol{b}\;\equiv\;((0\le 2\wedge 0\le 2)\vee(0\le 4\wedge 0\le -2))\vee(0\le 4)$.

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

Theorem 4 (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 $A\boldsymbol{x}\le\boldsymbol{b}$, the $\mathtt{FMP}^-$ method constructs $\mathcal{O}(n\cdot m^{n+1})$ new rows.*

Proof. The number $N(m,n)$ of constructed rows is maximal if the system consists only of lower bounds and one upper bound. Then, $\mathtt{FMP}^-$ yields $m-1$ new systems of size $m-1$, from which $n-1$ variables need to be eliminated; thus $N(m,n)\le(m-1)\cdot((m-1)+N(m-1,n-1))$. With $k=\min(n,m)$, we obtain $N(m,n)\le\sum_{i=1}^{k}(m-i)\cdot\prod_{j=1}^{i}(m-j)\le n\cdot m^{n+1}$. □

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: In the worst case from the above proof, FM obtains a single system of the same size as each of the sub-problems computed by $\mathtt{FMP}^-$. Although in this case, we could simply employ $\mathtt{FMP}^+$ instead, it is unclear whether there exists a rule for employing $\mathtt{FMP}^-$ or $\mathtt{FMP}^+$ that never produces more constraints than FM.

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 1.

Theorem 5. *Let $A\in\mathbb{Q}^{m\times n}$, $\boldsymbol{b}\in\mathbb{Q}^{m\times 1}$ and $k\in[m]$. Assume $(A^{(0)}\boldsymbol{x}\le\boldsymbol{b}^{(0)})=(A\boldsymbol{x}\le\boldsymbol{b})$ and for all $j\in[k]$, let $(A^{(j)}\boldsymbol{x}\le\boldsymbol{b}^{(j)})\in\mathtt{FMP}_j^-(A^{(j-1)}\boldsymbol{x}\le\boldsymbol{b}^{(j-1)})\cup\mathtt{FMP}_j^+(A^{(j-1)}\boldsymbol{x}\le\boldsymbol{b}^{(j-1)})$. Let $F^{(1)},\dots,F^{(k)}$ be the respective projection matrices, and $F=F^{(k)}\cdot\ldots\cdot F^{(1)}$. Then F is linearly independent.*

Proof. By definition, the projection matrices are linearly independent, and thus so is their product F. □

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 computes smaller sub-problems whose satisfiability implies the satisfiability of the original problem. Therefore, we do not compute the whole projection at once, but explore the decomposition using a depth-first search. 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 a leaf without any trivially false constraints exists.

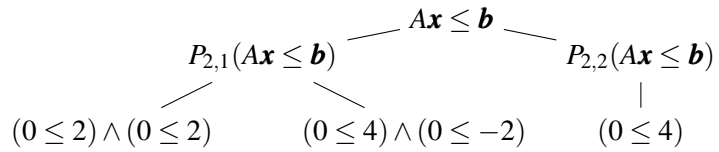


Figure 2: The search tree corresponding to Example 2. The very first leaf (bottom left) is already satisfiable, meaning that the rest would not need to be computed.

An example is depicted in Figure 2. We start with a basic version of the algorithm and then examine how the search tree can be pruned, resulting in two variants; all versions are given in Algorithm 1.

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.

Definition 5 (Branch Choices)**.** *The set of* branch choices *for a system $A\boldsymbol{x} \leq \boldsymbol{b}$ is*

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

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 1a where $A\boldsymbol{x} \leq \boldsymbol{b}$ is the currently processed sub-problem in the recursion tree. 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 $\texttt{FMplex}(\widehat{A}\boldsymbol{x} \leq \widehat{\boldsymbol{b}}) = \texttt{FMplex}(\widehat{A}\boldsymbol{x} \leq \widehat{\boldsymbol{b}}; \widehat{A}\boldsymbol{x} \leq \widehat{\boldsymbol{b}}, E^{(\widehat{m})})$. 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 extends it in reverse order in which the variables were eliminated. For every variable x_j, a value is picked above all lower and below all upper bounds on x_j evaluated at the underlying assignment. By the semantics of the projection, the value of the designated (largest lower or smallest upper) bound on x_j is suitable.

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 $\boldsymbol{a}_{i,\text{-}} = \boldsymbol{0}^{(n)}$ 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, we simply continue to explore the search tree. 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 1a, 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 and Theorem 6.

Theorem 6. *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 1a terminates with a global conflict.*

Algorithm 1: $\texttt{FMplex}(\widehat{A}\boldsymbol{x} \leq \widehat{\boldsymbol{b}}; A\boldsymbol{x} \leq \boldsymbol{b}, F, \boxed{\mathcal{N}, I}, \boxed{\texttt{lvl}, \texttt{bt\_lvl}})$

Algorithm 1a The base method consists of the plain (unframed and unfilled) parts.

Algorithm 1b Consists of the base method and the framed parts.

Algorithm 1c Consists of the base method, the framed parts and the filled boxes.

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 $F\widehat{\boldsymbol{b}} = \boldsymbol{b}$, $\boxed{\mathcal{N} \subseteq [\widehat{m}], I \subseteq [\widehat{m}]}$,
$\texttt{lvl} \in [n] \cup \{0\}$, and $\texttt{bt\_lvl} : [m] \to [n] \cup \{0\}$

Output: (SAT, α) with $\alpha \models A\boldsymbol{x} \leq \boldsymbol{b}$, or ($UNSAT$, K) where $K \subseteq [\widehat{m}]$, or
($PARTIAL\text{-}UNSAT$, l, K) where $l \in [n]$ and $K \subseteq [\widehat{m}]$

1 **if** $A = 0 \wedge \boldsymbol{b} \geq 0$ **then return** (SAT, ())
2 **if** $\exists i \in [m].\ \boldsymbol{a}_{i,\text{-}} = 0 \wedge b_i < 0 \wedge \boldsymbol{f}_{i,\text{-}} \geq 0$ **then return** $(UNSAT, \{i' \mid f_{i,i'} \neq 0\})$
3 **if** $\exists i \in [m].\ \boldsymbol{a}_{i,\text{-}} = 0 \wedge b_i < 0 \wedge \boldsymbol{f}_{i,\text{-}} \not\geq 0$ **then**
4 $\quad i := \arg\min_{i \in [m]} \{\texttt{bt\_lvl}(i) \mid \boldsymbol{a}_{i,\text{-}} = 0 \wedge b_i < 0\}$
5 $\quad$ **return** $(PARTIAL\text{-}UNSAT, \texttt{bt\_lvl}(i) - 1, \{i' \mid f_{i,i'} \neq 0\})$
6 $K = \emptyset$
7 **choose** $V \in branch\_choices(A\boldsymbol{x} \leq \boldsymbol{b}, \boxed{\{\mathcal{B}^{-1}_{\mathcal{N},F}(i) \mid i \in I\}})$
8 **foreach** $(x_j, i) \in V$ **do**
9 $\quad$ **compute** $A'\boldsymbol{x} \leq \boldsymbol{b}' := P_{j,i}(A\boldsymbol{x} \leq \boldsymbol{b})$ with projection matrix F' and backtrack levels $\texttt{bt\_lvl}'$
10 $\quad \boxed{\mathcal{N}' := \mathcal{N} \cup \{\mathcal{B}_{\mathcal{N},F}(i)\}}$ **if** $i \neq \perp$ **else** $\mathcal{N}$
11 $\quad$ **switch** $\texttt{FMplex}(\widehat{A}\boldsymbol{x} \leq \widehat{\boldsymbol{b}}; A'\boldsymbol{x} \leq \boldsymbol{b}', F'F, \boxed{\mathcal{N}', I}, \texttt{lvl} + 1, \texttt{bt\_lvl}')$ **do**
12 $\quad\quad$ **case** $(UNSAT, K')$ **do return** $(UNSAT, K')$
13 $\quad\quad$ **case** (SAT, α) **do return** $(SAT, \alpha[x_j \mapsto r])$ for a suitable $r \in \mathbb{Q}$
14 $\quad\quad$ **case** $(PARTIAL\text{-}UNSAT, l, K')$ **do**
15 $\quad\quad\quad$ **if** $l < \texttt{lvl}$ **then return** $(PARTIAL\text{-}UNSAT, l, K')$
16 $\quad\quad\quad$ **else** $K = K \cup K'$
17 $\quad \boxed{I := I \cup \{\mathcal{B}_{\mathcal{N},F}(i)\}}$
18 **if** $\texttt{lvl} = 0$ **then return** $(UNSAT, K)$
19 **return** $(PARTIAL\text{-}UNSAT, \texttt{lvl-1}, K)$

Proof Idea for Theorem 6. If $\widehat{A}\boldsymbol{x} \leq \widehat{\boldsymbol{b}}$ is unsatisfiable, then there exists a minimal unsatisfiable subset $\widehat{K}$ of the corresponding constraints. We construct a path in the search tree induced by Algorithm 1a yielding a conflict that is a linear combination of $\widehat{K}$. As $\widehat{K}$ is minimal, the linear combination is positive, i.e. the conflict is global. The other direction of the equivalence follows immediately with Farkas' Lemma. Consult the extended version for a detailed proof. □

4.1 Avoiding Redundant Checks

We observe that each row i 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 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 $A\boldsymbol{x} \leq \boldsymbol{b}$:

Theorem 7. *Let $\widehat{A} \in \mathbb{Q}^{\widehat{m}\times n}$ and $\widehat{\boldsymbol{b}} \in \mathbb{Q}^{\widehat{m}\times 1}$. Let $\texttt{FMplex}(\widehat{A}\boldsymbol{x} \leq \widehat{\boldsymbol{b}}; A\boldsymbol{x} \leq b, F)$ be a call in the recursion tree of the call $\texttt{FMplex}(\widehat{A}\boldsymbol{x} \leq \widehat{\boldsymbol{b}})$ to Algorithm 1a , 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 exists a set $\mathcal{N} \subseteq [\widehat{m}]$ 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. *there exists an injective mapping $\mathcal{B}_{\mathcal{N},F} : [m] \to [\hat{m}], i \mapsto \hat{\imath}$ with $\{\hat{\imath}\} = \{i' \in [\hat{m}] \mid f_{i,i'} \neq 0\} \setminus \mathcal{N}$.*

Proof Idea. The statement follows with a straight forward induction over the elimination steps, where the original row corresponding to the chosen bound is added to $\mathcal{N}$, and $\mathcal{B}_{\mathcal{N},F}$ keeps track of which constraint corresponds to which original row. Consult the extended version for a detailed proof. □

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 8. *Let $A \in \mathbb{Q}^{m\times n}$, $\boldsymbol{b} \in \mathbb{Q}^{m\times 1}$, $j \in [n]$, and let $i, i' \in [m]$ be row indices 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 7, if $P_{j,i}(A\boldsymbol{x} \leq \boldsymbol{b})$ is unsatisfiable, then $(A\boldsymbol{x} \leq \boldsymbol{b}) \wedge (\boldsymbol{a}_{i,\text{-}}\, \boldsymbol{x} = \boldsymbol{b_i})$ is unsatisfiable, and trivially $(A\, \boldsymbol{x} \leq \boldsymbol{b}) \wedge (\boldsymbol{a}_{i,\text{-}}\, \boldsymbol{x} = \boldsymbol{b_i}) \wedge (\boldsymbol{a}_{i',\text{-}}\, \boldsymbol{x} = \boldsymbol{b}_{i'})$ is unsatisfiable as well. Using Theorem 7 in the other direction yields that $P_{j,i'}(A\boldsymbol{x} \leq \boldsymbol{b}) \wedge (\boldsymbol{a}_{i,\text{-}}\, \boldsymbol{x} = \boldsymbol{b_i})$ is unsatisfiable. □

This suggests that if $\texttt{FMplex}(\widehat{A}\boldsymbol{x} \leq \widehat{\boldsymbol{b}}; A\boldsymbol{x} \leq \boldsymbol{b}, F)$ with non-basis $\mathcal{N}$ has a child call for row i which does not return *SAT*, then no other call in the recursion tree of $\texttt{FMplex}(\widehat{A}\boldsymbol{x} \leq \widehat{\boldsymbol{b}}; A\boldsymbol{x} \leq \boldsymbol{b}, F)$ where the corresponding non-basis contains $\mathcal{B}_{\mathcal{N},F}(i)$ will return *SAT* either. Hence, we can ignore $\mathcal{B}_{\mathcal{N},F}(i)$ 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 3. *Consider the system from Example 1, with an additional constraint $c_5 : (-x_2 \leq 0)$. If c_5 is tried first as greatest lower bound on x_2, then the combination with $c_2 : (-2x_2 \leq -2)$ yields the local conflict $\frac{1}{2}c_2 - c_5 = (0 \leq -1)$. Thus, this branch and, due to Theorem 8, any non-base containing row 5 yields an unsatisfiable system.*

Next, we try c_1 as greatest lower bound on x_2 resulting in the combinations $\frac{1}{2}c_2 - c_1 = (x_1 \leq 3)$, $c_5 - c_1 = (x_1 \leq 4)$, $c_1 + c_3 = (-3x_1 \leq -3)$ and $c_1 + c_4 = (-x_1 \leq 1)$ and corresponding non-base $\{1\}$.

If we now choose $(x_1 \leq 4)$ as smallest upper bound on x_1, leading to the non-base $\{1,5\}$, another local conflict occurs: $(x_1 \leq 3) - (x_1 \leq 4) = (0 \leq -1)$. As 5 is contained in the non-base, we could know beforehand that this would happen and thus avoid computing this branch.

We update the `FMplex` algorithm as shown in Algorithm 1b using the following definition:

Definition 6. *The set of* branch choices *for $A\boldsymbol{x} \leq \boldsymbol{b}$ with m rows w.r.t. $I \subseteq [m]$ is*

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

It is easy to see that this modification prevents visiting non-basis twice in the following sense:

Theorem 9. *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 1b. 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 10 states that, still, Algorithm 1b always terminates with *SAT* or a global conflict. This follows by a slight modification of the proof of Theorem 6, presented in the extended version of this paper.

Theorem 10. *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 1b terminates with a global conflict.* □

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 `lvl` with m rows whose backtracking levels are stored in $\texttt{bt\_lvl} : [m] \to ([n] \cup \{0\})$. If $\texttt{lvl} = 0$, then `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''

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

The backtracking scheme is given in Algorithm 1c , 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 11. *Let $\texttt{FMplex}(\_; A\boldsymbol{x} \leq \boldsymbol{b}, \_, \_, \_, \texttt{lvl}, \_)$ be a call to Algorithm 1c , 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 `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 1c does not always terminate with a *global* conflict (i.e. Theorem 6 does not hold any more), even if we do not ignore any rows (i.e. omit Line 17):

Example 4. *We use Algorithm 1c to eliminate variables with the static order x_3, x_2, x_1 from the system on the right, always branching on lower bounds. We first choose row* 1 *as greatest lower bound on x_3. Rows* 3 *and* 4 *are retained as they do not contain x_3 and the combination of row* 1 *with row* 5 *is positive, so these constraints have backtrack level* 0.

$$\begin{bmatrix} 0 & 0 & -1 \\ 1 & -1 & -1 \\ 1 & 0 & 0 \\ -1 & 1 & 0 \\ 0 & -1 & 1 \end{bmatrix} \cdot \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} \leq \begin{bmatrix} 0 \\ 0 \\ -1 \\ -1 \\ 0 \end{bmatrix}$$

The combination with row 2 *has backtrack level* 1 *because both rows are lower bounds. Using this constraint as greatest lower bound on x_2 and combining it with row* 4 *leads to a local conflict with backtrack level* 1. *This means that the call at level* 1 *is unsatisfiable and thus we backjump to level* 0.

The second branch is visited, leading to the non-basis $\mathcal{N} = \{2,5,1\}$ after three steps, where a local *conflict lets us backjump to level* 0 *again. As there are no more lower bounds on x_3, the algorithm returns UNSAT without finding a global conflict.*

5 Relation to Other Methods

5.1 Simplex Algorithm

The simplex method [6, 18] is an algorithm for linear optimization over the reals and is able to solve *linear programs*. The *general simplex* [7] 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), $A\boldsymbol{x} \leq \boldsymbol{b}$ 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 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 7, 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 1a, 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 1b 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 [20, 27] 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* $\text{zeros}(\varphi)$ 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 $\varphi[-\infty // x_j] \vee \bigvee_{\xi \in \text{zeros}(\varphi)} \varphi[\xi // x_j]$. 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 ([26] even proposes to consider only lower or upper bounds for the test candidates) lead to a very similar procedure as 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.

Virtual substitution can also be adapted for SMT solving [3] 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 [15], 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. [22], Cotton [5] and Korovin et al. [16, 17], 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 combinations which FM would compute.

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 [16] or [17] 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; possibility of very quick solutions vs. more control and knowledge about the possible choices.

Moreover, these methods offer no straight-forward adaption for quantifier elimination, while FMplex does. However, [22] and [5] 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 [4] and compared their performance in the context of satisfiability checking. Using the transformation given in [24] and case splitting as in [2], we extended the method to also handle strict and not-equal-constraints.

The base version of FMplex (Algorithm 1a) 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 $V \in \mathit{branch\_choices}(A\boldsymbol{x} \leq \boldsymbol{b}, I)$ such that $|V|$ is minimal; in case that this minimum is 1, we prefer choices $V = \{(x_j, \bot)\}$ for a $j \in [n]$ 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* [14]: 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 $I_j^+(A)$.

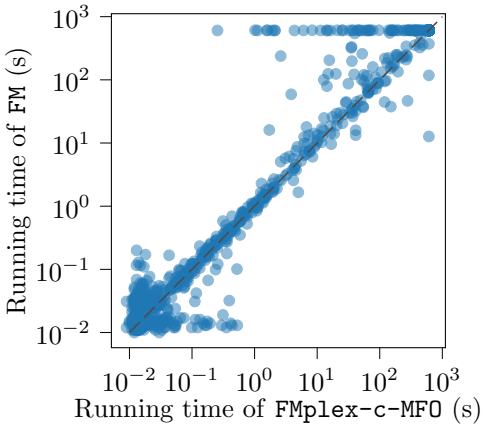
We first choose the rows which have the least non-zero coefficients (i.e. contain the least variables) to prefer sparse sub-problems. This can be understood as *Min-Row-Length*.

We consider the following solver variants: `FMplex-a-MFO` and `FMplex-a-MCL` implement Algorithm 1a with the Min-Fanout and the Min-Column-Length heuristic, respectively. `FMplex-a-Rand-1/2` denotes two variants of Algorithm 1a where all choices are taken pseudo-randomly with different seeds. `FMplex-b-MFO` implements Algorithm 1b and `FMplex-c-MFO` implements Algorithm 1c , 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.

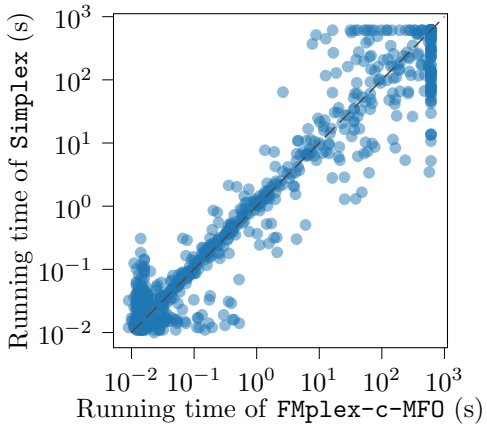
All solvers were tested on the SMT-LIB [1] benchmark set for QF_LRA containing 1753 formulas. As all evaluated solvers are non-incremental, we also generated conjunctions of constraints by solving each of these QF_LRA problems using a DPLL(T) SMT solver with an `FMplex-c-MFO` theory solver backend, and 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.

	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

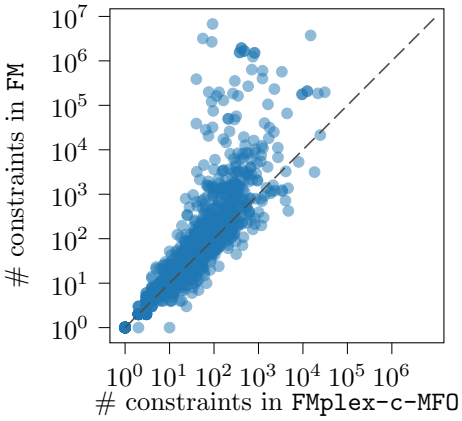
Table 1: Number of solved instances, timeouts (TO) and memory-outs (MO).



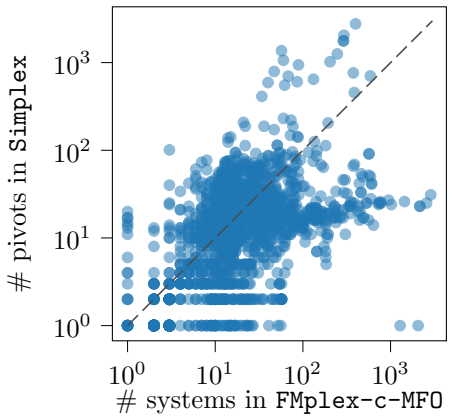
(a) Running times in seconds on the SMT-LIB benchmark set, `FMplex` vs `FM`.



(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 3: 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).

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, 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 Figures 3a and 3b 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.

Figure 3c 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 3d 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.

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 [19] and work on an incremental adaption for SMT solving. Last but not least, we plan to increase the applicability of FMplex as a quantifier elimination procedure, including a different handling of strict inequalities, which is more similar to FM.

References

[1] Clark Barrett, Pascal Fontaine & Cesare Tinelli (2016): *The Satisfiability Modulo Theories Library (SMT-LIB)*. www.SMT-LIB.org.

[2] Clark Barrett, Robert Nieuwenhuis, Albert Oliveras & Cesare Tinelli (2006): *Splitting on Demand in SAT Modulo Theories*. In: *International Conference on Logic for Programming Artificial Intelligence and Reasoning (LPAR'06)*, Springer, pp. 512–526, doi:10.1007/11916277_35.

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

[4] Florian Corzilius, Gereon Kremer, Sebastian Junges, Stefan Schupp & Erika Ábrahám (2015): *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)*, Springer, pp. 360–368, doi:10.1007/978-3-319-24318-4_26.

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

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

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

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

[9] Jean Baptiste Joseph Fourier (1827): *Analyse des travaux de l'Académie Royale des Sciences pendant l'année 1824*. *Partie mathématique*.

[10] Jean-Louis Imbert (1990): *About Redundant Inequalities Generated by Fourier's Algorithm*. In: *International Conference on Artificial Intelligence: Methodology, Systems, Applications (AIMSA'90)*, Elsevier, pp. 117–127, doi:10.1016/B978-0-444-88771-9.50019-2.

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

[12] Rui-Juan Jing, Marc Moreno-Maza & Delaram Talaashrafi (2020): *Complexity Estimates for Fourier-Motzkin Elimination*. In: *Computer Algebra in Scientific Computing (CASC'20)*, Springer, pp. 282–306, doi:10.1007/978-3-030-60026-6_16.

[13] Leonid Genrikhovich Khachiyan (1980): *Polynomial Algorithms in Linear Programming*. *USSR Computational Mathematics and Mathematical Physics* 20(1), pp. 53–72, doi:10.1016/0041-5553(80)90061-0.

[14] Tim King, Clark Barrett & Bruno Dutertre (2013): *Simplex with Sum of Infeasibilities for SMT*. In: *Formal Methods in Computer-Aided Design (FMCAD'13)*, pp. 189–196, doi:10.1109/FMCAD.2013.6679409.

[15] Konstantin Korovin, Marek Kosta & Thomas Sturm (2014): *Towards Conflict-driven Learning for Virtual Substitution*. In: *International Workshop on Computer Algebra in Scientific Computing (CASC'14)*, Springer, pp. 256–270, doi:10.1007/978-3-319-10515-4_19.

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

[17] Konstantin Korovin & Andrei Voronkov (2011): *Solving Systems of Linear Inequalities by Bound Propagation*. In: *Conference on Automated Deduction (CADE'23)*, Springer, pp. 369–383, doi:10.1007/978-3-642-22438-6_28.

[18] Carlton E. Lemke (1954): *The Dual Method of Solving the Linear Programming Problem*. *Naval Research Logistics Quarterly* 1(1), pp. 36–47, doi:10.1002/nav.3800010107.

[19] Haokun Li, Bican Xia, Huiying Zhang & Tao Zheng (2021): *Choosing the Variable Ordering for Cylindrical Algebraic Decomposition via Exploiting Chordal Structure*. In: *International Symposium on Symbolic and Algebraic Computation (ISSAC'21)*, ACM, pp. 281–288, doi:10.1145/3452143.3465520.

[20] Rüdiger Loos & Volker Weispfenning (1993): *Applying Linear Quantifier Elimination*. The Computer Journal 36(5), pp. 450–462, doi:10.1093/comjnl/36.5.450.

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

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

[23] Theodore Samuel Motzkin (1936): *Beiträge zur Theorie der linearen Ungleichungen*. Azriel.

[24] Jasper Nalbach, Erika Ábrahám & Gereon Kremer (2021): *Extending the Fundamental Theorem of Linear Programming for Strict Inequalities*. In: International Symposium on Symbolic and Algebraic Computation (ISSAC'21), ACM, pp. 313–320, doi:10.1145/3452143.3465538.

[25] Jasper Nalbach, Valentin Promies, Erika Ábrahám & Paul Kobialka (2023): *FMplex: A Novel Method for Solving Linear Real Arithmetic Problems*. arXiv:2309.03138.

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

[27] Volker Weispfenning (1997): *Quantifier Elimination for Real Algebra—the Quadratic Case and Beyond*. Applicable Algebra in Engineering, Communication and Computing 8(2), pp. 85–101, doi:10.1007/s002000050055.