Extending the Fundamental Theorem of Linear Programming for Strict Inequalities

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

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

Gereon Kremer
Stanford University
Stanford, USA
gkremer@stanford.edu

ABSTRACT

Usual formulations of the *fundamental theorem of linear programming* only consider weak inequalities as side conditions.

While this suffices for solving linear programs with the *Simplex algorithm*, when we want to check the satisfiability of general quantifier-free linear real arithmetic formulas via *Satisfiability Modulo Theories* (*SMT*) solving, we need to extend the Simplex method to be able to handle *strict* inequalities, too.

In this paper we formalize such an extension, which has been successfully used in SMT solving even before a correctness proof has been given by King in his dissertation in the year 2014. Our contribution is an alternative correctness formalization which is analogous to the original theorem, and a corresponding proof that better highlights the inherent nature of the problem.

CCS CONCEPTS

• **Hardware** → **Theorem proving and SAT solving**; • **Computing methodologies** → *Exact arithmetic algorithms*; • **Theory of computation** → Linear programming.

KEYWORDS

SMT solving; linear arithmetic; Simplex; linear programming

ACM Reference Format:
Jasper Nalbach, Erika Ábrahám, and Gereon Kremer. 2021. Extending the Fundamental Theorem of Linear Programming for Strict Inequalities. In *Proceedings of the 2021 International Symposium on Symbolic and Algebraic Computation (ISSAC '21), July 18–23, 2021, Virtual Event, Russian Federation.* ACM, New York, NY, USA, 8 pages. https://doi.org/10.1145/3452143.3465538

1 PROBLEM STATEMENT

Satisfiability Modulo Theories (*SMT*) solving uses a special framework to check the satisfiability of quantifier-free first-order logic formulas, which are Boolean combinations of constraints from a theory. To check the satisfiability of such formulas with SMT solving, a SAT solver determines truth values for the involved constraints such that the Boolean structure is satisfied, and consults a *theory solver* to check the consistency of this truth assignment in the underlying theory, i.e. to check the satisfiability of a *set* (or *conjunction*) of constraints.

In quantifier-free *linear real arithmetic* (QF_LRA, the SMT community's synonym for *linear algebra*), formulas combine constraints of the form $p \sim b$, where p is a linear expression over real-valued variables and rational coefficients, b is a rational constant and $\sim$ is a *weak* ($\leq$, $\geq$, $=$) or *strict* ($<$, $>$, $\neq$) relation symbol. In a QF_LRA theory solver, for checking the consistency of *weak* constraints, we can employ the *Simplex* algorithm [5, 6, 14], which was preceded by Kantorovich's method for problems of special type [19, 26]. The Simplex method builds on the foundation of the *fundamental theorem of linear programming*. In its generality, Simplex can do even more: it can solve *linear programs*, i.e. optimize a linear objective function under the side condition that certain weak linear inequalities are satisfied. If the side conditions admit a solution then their solution set is a (non-empty) polyhedron. Intuitively, the fundamental theorem states that we can restrict the search for an optimal solution to the vertices of this polyhedron, as one of those vertices evaluates the objective function to its optimal value. In the context of SMT solving where we do not require optimality, an adaption known as the *general Simplex* algorithm [11] has been proposed. This algorithm uses the fundamental theorem in the sense that every satisfiable system has a *basic feasible solution* and iterates over intersections of constraint-defining hyperplanes until it finds a vertex or detects that there is none.

When speaking about the Simplex algorithm, for some applications we are happy with approximate solutions, which makes the distinction of strict and weak constraints uninteresting. In contrast to that, in SMT solving, which has applications e.g. in program verification, exact solutions are required, satisfying the weak as well as the strict constraints. Note that even if syntactically only weak constraints appear in an input formula, solutions to its Boolean structure might require some of these weak constraints to be violated, indirectly resulting in strict inequalities.

We note that there exists research for dealing with strict inequalities [17, 21]; thus this work might be also of interest for research areas apart from SMT solving.

The fundamental theorem is not immediately applicable to problems involving strict constraints: the boundaries of a solution set might be open, thus the vertices of its closure are not necessarily solutions. Still, if the problem is solvable, then there exists a solution *in an infinitesimal-environment of a vertex of the closure*. To find such a solution, Dutertre and de Moura proposed in 2006 [10, 11] an elegant reduction: Strict constraints are replaced by weak constraints containing an infinitesimal ε. Solutions are then searched in the transformation and translated back to the original system. This approach is implemented in SMT solvers like `Yices`

[10], OpenSMT [18], CVC4 [1], Z3 [7] and SMT-RAT [4] and is even extended for infinity ∞ to represent unbounded objectives in optimization [2]; further *transcendental* extensions of the real closed field in the context of SMT solving are examined in [8].

While this extension seems natural, its correctness proof is non-trivial. A first formal proof has been given in 2014 by King [20], tailored specifically to the general Simplex method. In this paper we formalize and prove a more general statement, applicable to any procedure that relies on the general theorem of linear programming. Beyond its general nature, our proof helps to understand the mechanisms and elegantly illustrates the nature of the problem. A more in-depth comparison of the proof of King and our proof is given in Section 4.

We first introduce some mathematical foundations in Section 2. Then we present the transformation and our extension of the theorem in Section 3. Finally, advantages of the construction and its alternatives as well as related work and applications are discussed in Section 4.

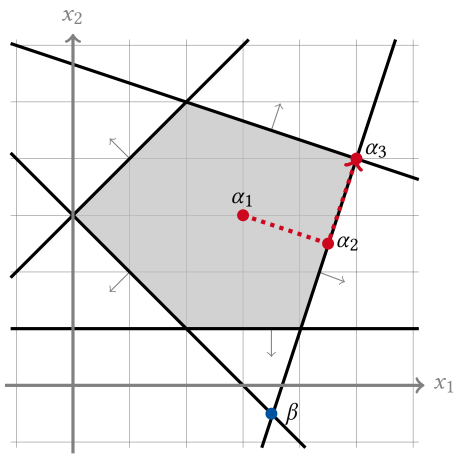


Figure 1: Illustration of Theorem 2.1.

2 PRELIMINARIES

Let $\mathcal{F}$ be a field, $(\mathcal{U}, <)$ an ordered vector space over $\mathcal{F}$ (or $\mathcal{F}$-vector space) and $X=\{x_1, \ldots, x_n\}$ be a set of $\mathcal{U}$-valued variables[1]. We will use $\mathcal{U}_{>0} = \{a \in \mathcal{U} \mid a > 0\}$ and similarly for other value restrictions.

We call $p = a_1 \cdot x_1 + \ldots + a_n \cdot x_n$ with $a_1, \ldots, a_n \in \mathcal{F}$ and $b \in \mathcal{U}$ a *linear* ($\mathcal{F}$-)*combination of* X, and $p \sim b$ with a *relation symbol* $\sim \in \{=, \leq, \geq, <, >, \neq\}$ a *linear constraint (over* $(\mathcal{U}, <)$ *in variables* X*)*. An *equation* is a linear constraint where $\sim$ is the equality relation $=$. A constraint $p \sim b$ is called *weak* if $\sim \in \{=, \leq, \geq\}$ and *strict* otherwise.

Linear combinations p and constraints c can be evaluated under an *assignment* $\alpha : X \longrightarrow \mathcal{U}$ the standard way. We denote by $\alpha(p) \in \mathcal{U}$ the value of p under α. For a constraint c of the form $p \sim b$, we write $\alpha \models c$ to state that $\alpha(p) \sim b$; we call α a *solution* of c. The solutions of c build its *solution set* $\mathrm{sol}(c) = \{\alpha : X \longrightarrow \mathcal{U} \mid \alpha \models c\}$. A constraint is *satisfiable* iff its solution set is not empty.

A *system* (*of linear constraints over* $(\mathcal{U}, <)$) is a finite set $C = \{p_1 \sim_1 b_1, \ldots, p_m \sim_m b_m\}$ of linear constraints for some $m \in \mathbb{N}$. We define $\mathrm{sol}(C) = \cap_{c \in C} \mathrm{sol}(c)$ and $P_C = \{p \mid (p \sim b) \in C\}$. A system C is *satisfiable* iff $\mathrm{sol}(C)$ not empty.

A set $\{p_1, \ldots, p_m\}$ of linear $\mathcal{F}$-combinations of X is *linearly independent* iff $f_1 \cdot p_1 + \ldots + f_m \cdot p_m = 0 \Leftrightarrow f_1 = \ldots = f_m = 0$ for all $f_1, \ldots, f_m \in \mathcal{F}$. A system C of linear constraints is linearly independent iff P_C is linearly independent. The *rank* of C is defined as $\mathrm{rank}(C) = \max\{\, |C'| \mid C' \subseteq C, C' \text{ linearly independent}\}$. The set $\mathfrak{B}_C$ contains all *maximal linearly independent subsets of* C, that are all sets $V \subseteq C$ such that $|V| = \mathrm{rank}(V) = \mathrm{rank}(C)$.

A formula φ in (quantifier-free) *linear real arithmetic* is a Boolean combination of linear constraints $p \sim b$ over $(\mathbb{R}, <)$ where $\mathbb{R}$ is viewed as a $\mathbb{Q}$-vector space and $b \in \mathbb{Q}$. Note that such a formula φ has a solution over $(\mathbb{R}, <)$ iff it has a solution over $(\mathbb{Q}, <)$.

2.1 The fundamental theorem of linear programming

As for SMT solving only the existence of a solution for the side conditions is of interest, we present a variant of the corresponding part of the *fundamental theorem of linear programming*, which builds the basis for the Simplex algorithm [5, 6, 14] as presented in [22].

Let c be a weak linear constraint $p \sim b$, then $\tilde{c}$ denotes the equation $p = b$. Given a set V of weak linear constraints, we define $\tilde{V} := \{\tilde{c} \mid c \in V\}$. Geometrically, $\mathrm{sol}(c)$ is a halfspace, $\mathrm{sol}(\tilde{c})$ is its bounding hyperplane, and $\mathrm{sol}(\tilde{V})$ consists of the intersection points of these bounding hyperplanes for the constraints in V.

Theorem 2.1 (Adaption of the fundamental theorem of linear programming). *Let C be a system of weak linear constraints over $(\mathcal{U}, <)$ in X. Then C is satisfiable iff there exists a maximal linearly independent subset $V \in \mathfrak{B}_C$ of C such that*

$$\exists \alpha : X \longrightarrow \mathcal{U}.\ \alpha \models \tilde{V} \cup C.$$

A formal proof is given in [24]; here, we give an intuition. While the backward direction is trivial, an intuitive illustration for the forward direction is shown in Figure 1. A constraint $p \sim b$ is called *tight* under an assignment α iff $\alpha(p) = b$. Observe that a solution α is a vertex of the solution set of a system C iff all constraints of a set $V \in \mathfrak{B}_C$ are tight under α (e.g. α_3 in Fig. 1), but not all assignments which make $\mathrm{rank}(C)$ many constraints tight are solutions (e.g. β in Fig. 1).

Starting from any solution, we construct the set V from the theorem as follows: We move the solution towards a direction until a constraint that defines a face of the solution polyhedron becomes tight (but is still satisfied); this step is iterated maintaining that all tight constraints remain tight, thus moving the solution on their boundaries. This is done until the solution cannot be moved along the boundaries of the tight constraints - which means that the set of tight constraints are a maximal linearly independent subset of C.

3 EXTENSION FOR STRICT INEQUALITIES

We will first formally introduce the transformation replacing strict constraints by weak ones and show some basic relationships between the solution sets of the original and the transformed system in Section 3.1. Then, we present an intermediate theorem in Section 3.2, before we introduce an infinitesimal to obtain the main statement in Section 3.3.

[1]This is partly unconventional, but required for formal reasons as we will see in Section 3.3.

For simplicity, we formalize our statements for the relations $\leq$ and $<$ only; their generalization is straightforward by replacing each equality $p{=}b$ by a pair of constraints $p{\geq}b$ and $p{\leq}b$, and transforming each $p{>}b$ resp. $p{\geq}b$ to $-p{<}-b$ resp. $-p{\leq}-b$. Handling disequalities using $\neq$ will be dealt with at the end of this section.

3.1 Transformation

Definition 3.1. Let C be a system of linear constraints over $(\mathcal{U}, <)$ in X with $\sim \in \{\leq, <\}$ for all $(p \sim b) \in C$. Using a fresh variable $\varepsilon \notin X$, we define the system C_w of linear constraints over $(\mathcal{U}, <)$ in $X \cup \{\varepsilon\}$ as

$$C_w \; = \; \{p + \varepsilon \leq b \mid (p < b) \in C\} \; \cup \; \{p \leq b \mid (p \leq b) \in C\}$$

Under the assumption $\varepsilon > 0$, the systems C and C_w are satisfiability equivalent: in C_w we replace each strict constraint $p < b$ by a weak counterpart $p + \varepsilon \leq b$, where the new variable ε puts a lower bound on the distance of p to its upper bound b. Thus we can reduce the satisfiability problem for C to that of C_w under the additional condition $\varepsilon > 0$. We formalize this relationship in the following two lemmas.

For a function $f : A \to B$ and a set $A' \subseteq A$, let $f\downharpoonright_{A'} : A' \to B$ with $f\downharpoonright_{A'}(a) = f(a)$ for all $a \in A'$, denote the *restriction* of f to A'. For $a \notin A$ and $b \in B$, $f[a \mapsto b] : (A \cup \{a\}) \to B$ denotes the *extension* of f such that $f[a \mapsto b](a) = b$ and $f[a \mapsto b](a') = f(a')$ for all $a' \in A$.

It is easy to see that a solution for C_w with positive ε is already a solution for C:

Lemma 3.2. *Let C be a system of linear constraints over $(\mathcal{U}, <)$ in X. Then for any $\alpha_w : (X \cup \{\varepsilon\}) \to \mathcal{U}$ such that $\alpha_w(\varepsilon) > 0$ and $\alpha_w \models C_w$,*

$$\alpha_w\downharpoonright_X \models C.$$

For the converse direction, we show that a solution for C can be extended with a strictly positive value for ε satisfying C_w. In fact, we prove something stronger, that is, there exists a value g for ε such that any value between 0 and g satisfies C_w.

Lemma 3.3. *Let C be a system of linear constraints over $(\mathcal{U}, <)$ in X. Then for any $\alpha : X \to \mathcal{U}$ such that $\alpha \models C$,*

$$\exists g \in \mathcal{U}_{>0}.\ \forall e \in \mathcal{U}_{(0,g]}.\ \alpha[\varepsilon \mapsto e] \models C_w.$$

Proof. Note that for $(p_i \leq b_i) \in C \cap C_w$, $\alpha[\varepsilon \mapsto e] \models p_i \leq b_i$ for all $e \in \mathcal{U}$; in particular, if $C_w = C$, then $\alpha[\varepsilon \mapsto e] \models C_w$ for all $e \in \mathcal{U}$, thus any g can be taken as witness. Otherwise, let $(p_i + \varepsilon \leq b_i) \in C_w$, thus $(p_i < b_i) \in C$. Then by assumption,

$$\alpha(p_i) < b_i \Leftrightarrow 0 < b_i - \alpha(p_i) =: g_i.$$

For all $e \in \mathcal{U}$ with $0 < e \leq g_i$ we have $\alpha(p_i) = b_i - g_i < b_i - e$ and therefore

$$\alpha[\varepsilon \mapsto e] \models p_i + \varepsilon \leq b_i\ .$$

Thus, for all $e \in \mathcal{U}$ such that $0 < e \leq g := \min(\{g_i \mid (p_i + \varepsilon \leq b_i) \in C_w\} \cup \{1\})$, it holds $\alpha[\varepsilon \mapsto e] \models C_w$. □

We illustrate this relationship in the following example.

Example 3.4. Consider $C = \{-x_1 < 0,\ -x_2 < 0,\ x_2 < 2\}$ and its transformation $C_w = \{-x_1 + \varepsilon \leq 0,\ -x_2 + \varepsilon \leq 0,\ x_2 + \varepsilon \leq 2\}$. Figure 2a depicts $\mathrm{sol}(C_w)$ restricted to $\varepsilon > 0$, which has a convex tent-shaped form: the open grey area depicts the two-dimensional solutions of C, where the tent's roof is the upper bound on ε w.r.t. the values for x_1 and x_2, in order to satisfy C_w.

Lemma 3.2 states that any solution of C_w (red point) is a solution of C after projecting out ε. Lemma 3.3 states that for any solution of C (purple point on the horizontal plane), any point strictly above it (to satisfy $\varepsilon > 0$) up to and including the tent's roof (on the dashed red line) is a satisfying solutions of C_w.

Now consider the system $C' = \{-x_1 < 0,\ -x_2 < 0,\ 2 \cdot x_2 < 4\}$ and its transformation $C'_w = \{-x_1 + \varepsilon \leq 0,\ -x_2 + \varepsilon \leq 0,\ 2x_2 + \varepsilon \leq 4\}$, which is the same as above except that the last constraint is scaled by a factor of 2. Although the solution set of C and C' are identical, the solution set of C_w and C'_w are not, as depicted in Figure 2b.

3.2 Intermediate theorem

In this section we characterize solutions of C that are arbitrarily close to a vertex of the closure of $\mathrm{sol}(C)$. This set is obtained by the application of Theorem 2.1 to the transformed system C_w.

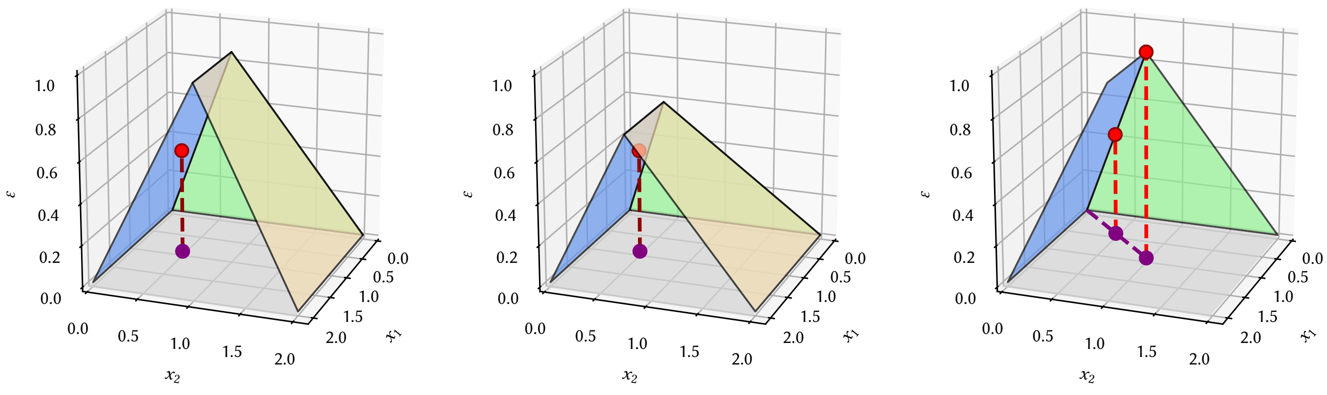
Example 3.5. Figure 2c depicts again the system from Example 3.4 but without plotting the right tent wall for a better view. Assume a solution for C_w with a value $g > 0$ for ε. Fixing the value g for ε, the solution space in the variables x_1 and x_2 is the horizontal cut surface of the tent at $\varepsilon = g$. According to the fundamental theorem, there exists a solution which is a vertex of this cut, on a ridge at $\varepsilon = g$ (red point). As already seen, the projection of this solution (purple point, dashed red line) is a solution of C. The same holds for any value $0 < e \leq g$ for ε, resulting in solutions for C arbitrarily close to a vertex of the closure of $\mathrm{sol}(C)$ (dashed purple line).

Theorem 3.6. *Let C be a system of linear constraints over $(\mathcal{U}, <)$ in X. Then C is satisfiable if and only if there exists $V \subseteq C_w$ with $|V| = \mathrm{rank}(C)$ such that $P_V \cup \{\varepsilon\}$ is linearly independent and*

$$\exists g \in \mathcal{U}_{>0}.\ \forall e \in \mathcal{U}_{(0,g]}.\ \exists \alpha : X \to \mathcal{U}.\ \alpha[\varepsilon \mapsto e] \models \tilde{V} \cup C_w.$$

Proof. For the backward direction, we choose witnesses for $g > 0$, $e \in (0, g]$ and α to obtain an assignment $\alpha \models C_w$ with $\alpha(\varepsilon) = e > 0$. By Lemma 3.2, it follows that C is satisfiable. The forward direction is proven as follows:

$$\begin{aligned}
& \exists \alpha : X \to \mathcal{U}.\ \alpha \models C \\
\overset{(1)}{\Longrightarrow}\ & \exists \alpha : X \to \mathcal{U}.\ \exists g \in \mathcal{U}_{>0}.\ \forall e \in \mathcal{U}_{(0,g]}.\ \alpha[\varepsilon \mapsto e] \models C_w \\
\Longrightarrow\ & \exists \alpha : X \to \mathcal{U}.\ \exists g \in \mathcal{U}_{>0}.\ \forall e \in \mathcal{U}_{(0,g]}. \\
& \quad \alpha[\varepsilon \mapsto e] \models C_w \cup \{\varepsilon = e\} \\
\Longrightarrow\ & \exists g \in \mathcal{U}_{>0}.\ \forall e \in \mathcal{U}_{(0,g]}. \\
& \quad \exists \alpha : X \cup \{\varepsilon\} \to \mathcal{U}.\ \alpha \models C_w \cup \{\varepsilon = e\} \\
\overset{(2)}{\Longrightarrow}\ & \exists g \in \mathcal{U}_{>0}.\ \forall e \in \mathcal{U}_{(0,g]}.\ \exists V \in \mathfrak{B}_{C_w \cup \{\varepsilon = e\}}. \\
& \quad \exists \alpha : X \cup \{\varepsilon\} \to \mathcal{U}.\ \alpha \models \tilde{V} \cup C_w \cup \{\varepsilon = e\} \\
\Longrightarrow\ & \exists g \in \mathcal{U}_{>0}.\ \forall e \in \mathcal{U}_{(0,g]}.\ \exists V \in \mathfrak{B}_{C_w \cup \{\varepsilon = e\}}. \\
& \quad \exists \alpha : X \to \mathcal{U}.\ \alpha[\varepsilon \mapsto e] \models \tilde{V} \cup C_w \\
\overset{(3)}{\Longrightarrow}\ & \exists g \in \mathcal{U}_{>0}.\ \forall e \in \mathcal{U}_{(0,g]}. \\
& \quad \exists V' \subseteq C_w.\ (|V'| {=} \mathrm{rank}(C) \wedge (P_{V'} \cup \{\varepsilon\} \text{ lin. indep.}) \wedge \\
& \quad\quad \exists \alpha : X \to \mathcal{U}.\ \alpha[\varepsilon \mapsto e] \models \tilde{V'} \cup C_w) \\
\overset{(4)}{\Longrightarrow}\ & \exists V' \subseteq C_w.\ (|V'| {=} \mathrm{rank}(C) \wedge (P_{V'} \cup \{\varepsilon\} \text{ lin. indep.}) \wedge \\
& \quad \exists g \in \mathcal{U}_{>0}.\ \forall e \in \mathcal{U}_{(0,g]}. \\
& \quad\quad \exists \alpha : X \to \mathcal{U}.\ \alpha[\varepsilon \mapsto e] \models \tilde{V'} \cup C_w)
\end{aligned}$$



(a) System C (b) System C' (c) Solutions arbitrarily close to a vertex

Figure 2: Illustration of Definition 3.1

(1) Lemma 3.3
(2) The consequence results from the application of Theorem 2.1 to the term $\exists\alpha : X \cup \{\varepsilon\} \longrightarrow \mathcal{U}.\ \alpha \models C_w \cup \{\varepsilon = e\}$. That means, for every e witnessing the left statement, we can find a satisfying vertex of the system $C_w \cup \{\varepsilon = e\}$; thus, we apply the theorem simultaneously to all those systems.
(3) Let $g \in \mathcal{U}_{>0}$ such that the left statement holds. Let $e \in \mathcal{U}_{(0,g]}$ and $V \in \mathfrak{B}_{C_w \cup \{\varepsilon = e\}}$. We now construct a witness for V'. First note that $\mathrm{rank}(C_w \cup \{\varepsilon = e\}) = \mathrm{rank}(C)+1$ as $C_w \cup \{\varepsilon = e\}$ contains one variable ε more than C and at least one constraint $\varepsilon = e$ in this variable. Thus, we set $V' = V \setminus \{c\}$ for a constraint c such that $P_{V'} \cup \{\varepsilon\}$ is linearly independent, which is chosen as follows:
Let $V = \{p_1 \sim_1 b_1, \ldots, p_k \sim_k b_k\}$. By the choice of V, $\varepsilon = e$ linearly depends on V, that is there exists a unique $f_1, \ldots, f_k$ such that $\varepsilon = f_1 \cdot p_1 + \ldots + f_k \cdot p_k$ and $f_i \neq 0$ for at least one $i \in [k]$. Choosing $c = (p_i \sim_i b_i)$ guarantees the linear independence of $P_{V'} \cup \{\varepsilon\}$ (as otherwise, V would be linearly dependent).
(4) Let $g \in \mathcal{U}_{>0}$ such that the left statement holds. Let $e \in \mathcal{U}_{(0,g]}$. Let $W_e \neq \emptyset$ denote the set of possible choices of V'. Since C_w is finite, also W_e is finite. Lemma 3.7 below implies that if there is a $V' \in W_e$ such that $V' \notin W_{e'}$ for some $e' < e$, then $V' \notin W_{e''}$ for all $e'' < e'$. It follows that there exists $g' \in (0, g]$ such that for $e, e' \in (0, g']$, it holds $W_e = W_{e'}$. Thus, for any set $V' \in W_e = W_{e'}$, g' satisfies the statement. It follows that the implication holds. □

Lemma 3.7. *Let $V \subseteq C_w$ with $|V| = \mathrm{rank}(C)$ such that $P_V \cup \{\varepsilon\}$ is linearly independent. Let $e, e' \in \mathcal{U}$ such that $0 < e' < e$. Let $\alpha : X \cup \{\varepsilon\} \longrightarrow \mathcal{U}$ such that $\alpha \models \tilde{V} \cup C_w \cup \{\varepsilon = e\}$. Furthermore, assume there exists no α' such that $\alpha' \models \tilde{V} \cup C_w \cup \{\varepsilon = e'\}$.*

Then for every $e'' \in \mathcal{U}$, $e'' < e'$ and every assignment α'' it holds $\alpha'' \not\models \tilde{V} \cup C_w \cup \{\varepsilon = e''\}$.

Proof. Let $V = \{p_i \leq b_i \mid i = 1, \ldots, k\}$, e, e' and α as specified in the Lemma 3.7. Furthermore, let $\alpha', \alpha'' : X \cup \{\varepsilon\} \longrightarrow \mathcal{U}$ such that $\alpha' \models \tilde{V} \cup \{\varepsilon = e'\}$ but $\alpha' \not\models C_w$, and $\alpha'' \models \tilde{V} \cup \{\varepsilon = e''\}$. Note that such α' and α'' exist, since $P_V \cup \{\varepsilon\}$ is linearly independent.

Let $(p \leq b) \in C_w$ such that $\alpha'(p) > b$. Note that from $\alpha' \models \tilde{V}$ it follows that $(p \leq b) \notin V$. As $V \cup \{\varepsilon = e\}$ is a maximal linearly independent subset of C_w, there exist $f_1, \ldots, f_k, f_\varepsilon \in \mathcal{F}$ not all 0 such that

$$p = f_1 \cdot p_1 + \ldots + f_k \cdot p_k + f_\varepsilon \cdot \varepsilon$$

and as $\alpha \models \tilde{V} \cup \{\varepsilon = e\}$,

$$\alpha(p) = f_1 \cdot \underbrace{\alpha(p_1)}_{=b_1} + \ldots + f_k \cdot \underbrace{\alpha(p_k)}_{=b_k} + f_\varepsilon \cdot \underbrace{\alpha(\varepsilon)}_{=e}.$$

Analogous statements hold for α' and α''. From these statements together with the observation that $\alpha(p_i) = \alpha'(p_i) = \alpha''(p_i) = b_i$ for $i = 1, \ldots, k$ (due to the satisfaction of $\tilde{V}$) it follows that

$$\alpha'(p) - \alpha(p) = f_\varepsilon \cdot (e' - e) \text{ and } \alpha''(p) - \alpha'(p) = f_\varepsilon \cdot (e'' - e').$$

As by assumption, $\alpha(p) \leq b$ and $\alpha'(p) > b$, it holds $\alpha'(p) - \alpha(p) > 0$, i.e. $f_\varepsilon \cdot (e' - e) > 0$ and thus using $e' < e$ we get $f_\varepsilon < 0$. Taking into account that $e'' < e'$, it holds $\alpha''(p) - \alpha'(p) = f_\varepsilon \cdot (e'' - e') > 0$ as well. Finally, we can conclude

$$\alpha''(p) = \underbrace{\alpha'(p)}_{>b} + \underbrace{\alpha''(p) - \alpha'(p)}_{>0} > b$$

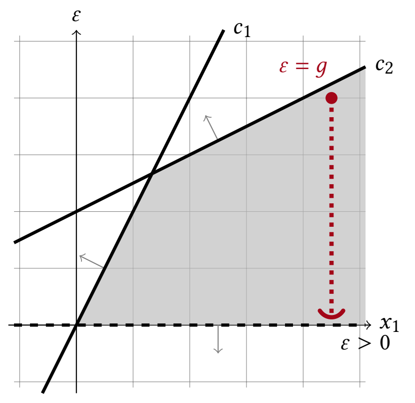
and thus, $\alpha'' \not\models p \leq b$. □

Example 3.8. We illustrate the steps in the proof of Theorem 3.6 on the example $C = \{-2 \cdot x_1 < 0, -\frac{1}{2} \cdot x_1 < 2\}$ respectively

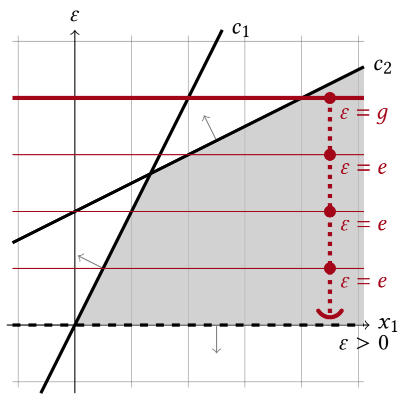
$$C_w = \{\underbrace{-2 \cdot x_1 + \varepsilon \leq 0}_{c_1}, \underbrace{-\frac{1}{2} \cdot x_1 + \varepsilon \leq 2}_{c_2}\}.$$

By Implication (1), if C is satisfiable, then $C_w \wedge \varepsilon > 0$ (depicted in Figure 3a) is as well, and for any solution of the latter, its ε-component can be made arbitrarily small while still remaining in the solution set. The ε-component of the depicted point corresponds to the quantified variable $g \in \mathcal{U}_{>0}$ from the statement and for the points below to the variable $e \in \mathcal{U}_{(0,g]}$ respectively.

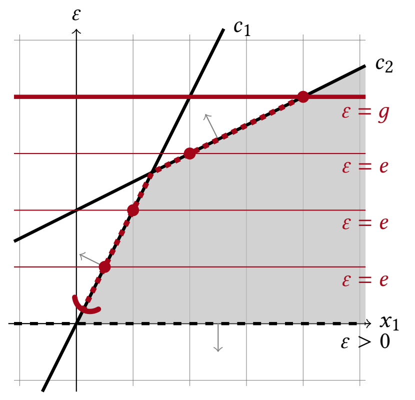
Now, we apply the fundamental theorem of linear programming. To do so, we introduce an artificial constraint $\varepsilon = e$, thus for every such point, we obtain a restricted system. Figure 3b depicts these systems for some possible values e.



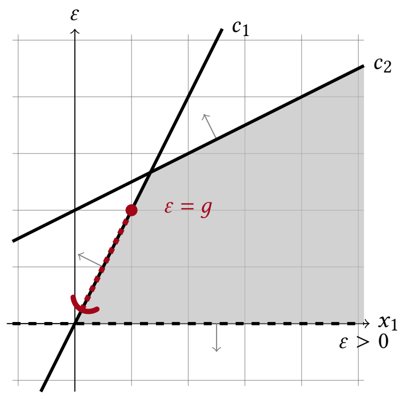
(a) Any point strictly between a point (red) in the solution set (grey) of $C_w \wedge \varepsilon > 0$ and its projection onto the x_1 axis is a solution (dashed line).



(b) For any value for ε less or equal to g and greater than 0, a satisfying solution can be found.



(c) For any value for ε less or equal to g and greater than 0, a satisfying solution on a ridge of the solution set can be found.



(d) If g is chosen small enough, for any value for ε less or equal to g and greater than 0, a satisfying solution on the same ridge can be found.

Figure 3: Illustration of the proof of Theorem 3.6

Implication (2) is the application of the fundamental theorem of linear programming to each of those systems. By doing so, we know that for every value e below g, we can find a satisfying solution with ε-component e on the ridge of the original polyhedron, as depicted in Figure 3c.

The aim of the last steps in Implications (3) and (4) is to choose the set V defining the ridge of the polyhedron independently from the choice of g. To do so, we prove that there is a sufficiently small g such that we can reside on the same ridge for decreasing ε-values - especially for arbitrarily small values of ε. Intuitively this is clear: Given a ridge that we moved along (making ε smaller) but is cut off by another ridge at some point, we will never move along this ridge again; this is exactly, what is proven in Lemma 3.7. As there are only finitely many constraints, the set of ridges that we can move along is also finite. Thus, we obtain the desired statement, graphically depicted in Figure 3d.

3.3 Infinitesimal arithmetic

The characterization given by Theorem 3.6 allows to find solutions arbitrarily close to a vertex. Finally, by employing *infinitesimal arithmetic*, we obtain a formalism allowing to find those solutions algorithmically.

We introduce a common transcendental extension of the real closed field for an infinitesimal value. For exhaustive theoretical foundations we refer for example to [8]; this extension has also been examined in the context of linear programming [6, 12].

Definition 3.9 (Infinitesimal). Let $(\mathcal{U}, <)$ be an ordered vector space over $\mathcal{F}$. We define ε as positive infinitesimal, that is

$$\forall c \in \mathcal{U}_{>0}.\ 0 < \varepsilon < c.$$

The *extension of $\mathcal{U}$ for ε* is the vector space $\mathcal{U}[\varepsilon] = \{a + b \cdot \varepsilon \mid a, b \in \mathcal{U}\}$ over $\mathcal{F}$ with operations

- $+ : \mathcal{U}[\varepsilon] \times \mathcal{U}[\varepsilon] \rightarrow \mathcal{U}[\varepsilon]$ with $(a_1 + b_1 \cdot \varepsilon) + (a_2 + b_2 \cdot \varepsilon) = (a_1 + a_2) + (b_1 + b_2) \cdot \varepsilon$
- $\cdot : \mathcal{F} \times \mathcal{U}[\varepsilon] \rightarrow \mathcal{U}[\varepsilon]$ with $c \cdot (a + b \cdot \varepsilon) = c \cdot a + c \cdot b \cdot \varepsilon$

We define the *extension of $(\mathcal{U}, <)$ for ε* as the ordered vector space $(\mathcal{U}[\varepsilon], <)$ over $\mathcal{F}$ with $< \subseteq \mathcal{U}[\varepsilon] \times \mathcal{U}[\varepsilon]$ such that

$$(a_1 + b_1 \cdot \varepsilon) < (a_2 + b_2 \cdot \varepsilon) \text{ iff } a_1 < a_2 \vee (a_1 = a_2 \wedge b_1 < b_2).$$

For a system C of linear constraints over $(\mathcal{U}, <)$ in $X \cup \{\varepsilon\}$, C^* denotes C interpreted as system over $(\mathcal{U}[\varepsilon], <)$ in X.

Theorem 3.10 (Main theorem). *Let C be a system of linear constraints over $(\mathcal{U}, <)$ in X. Then C is satisfiable if and only if there exists a maximal linearly independent subset $V \in \mathfrak{B}_{C_w^*}$ of C_w^* such that*

$$\exists \alpha^* : X \to \mathcal{U}[\varepsilon].\ \alpha^* \models \tilde{V} \cup C_w^*.$$

Proof.

$$\begin{aligned}
&\exists \alpha : X \to \mathcal{U}.\ \alpha \models C\\
\overset{(1)}{\Leftrightarrow}\ &\exists V \subseteq C_w.\ (\ |V| = \operatorname{rank}(C) \wedge (P_V \cup \{\varepsilon\} \text{ lin. indep.}) \wedge\\
&\qquad \exists g \in \mathcal{U}_{>0}.\ \forall e \in \mathcal{U}_{(0,g]}.\\
&\qquad \exists \alpha : X \to \mathcal{U}.\ \alpha[\varepsilon \mapsto e] \models \tilde{V} \cup C_w)\\
\overset{(2)}{\Leftrightarrow}\ &\exists V \in \mathfrak{B}_{C_w^*}.\\
&\qquad \exists g \in \mathcal{U}_{>0}.\ \forall e \in \mathcal{U}_{(0,g]}.\\
&\qquad \exists \alpha^* : X \to \mathcal{U}[\varepsilon].\ (\alpha^* \models \tilde{V} \wedge (\varphi_e \circ \alpha^*)[\varepsilon \mapsto e] \models C_w)\\
\overset{(3)}{\Leftrightarrow}\ &\exists V \in \mathfrak{B}_{C_w^*}.\\
&\qquad \exists \alpha^* : X \to \mathcal{U}[\varepsilon].\ (\alpha^* \models \tilde{V} \wedge\\
&\qquad \exists g \in \mathcal{U}_{>0}.\ \forall e \in \mathcal{U}_{(0,g]}.\ (\varphi_e \circ \alpha^*)[\varepsilon \mapsto e] \models C_w)\\
\overset{(4)}{\Leftrightarrow}\ &\exists V \in \mathfrak{B}_{C_w^*}.\ \exists \alpha^* : X \to \mathcal{U}[\varepsilon].\ (\alpha^* \models \tilde{V} \wedge \alpha^* \models C_w^*)
\end{aligned}$$

where $\varphi_e : \mathcal{U}[\varepsilon] \to \mathcal{U}, a + b \cdot \varepsilon \mapsto a + b \cdot e$ is the *substitution homomorphism* for $e \in \mathcal{U}$.

(1) Theorem 3.6
(2) First note that $\operatorname{rank}(C_w^*) = \operatorname{rank}(C)$; moreover, the witnesses for V on both sides of the equivalence correspond to each other: On the left side, it is a set of constraint with variables in $X \cup \{\varepsilon\}$, on the right side the same constraints but seen in variables X assigned to values in $\mathcal{U}[\varepsilon]$.
Given this, the backward direction is already proven. For the forward direction, there is more involved: As $P_V \cup \{\varepsilon\}$ is linearly independent, $\tilde{V}$ admits a solution for any value of ε (i.e. by the *Rouché-Capelli theorem* [25]). It is easy to see that such a set of solutions in $X \cup \{\varepsilon\}$ can be described by an $\alpha^* : X \to \mathcal{U}[\varepsilon]$.
(3) The backward direction is trivial.
For the forward direction we note that given a fixed V, the set of assignments α^* for which $\alpha^* \models \tilde{V}$ holds is independent from the choice of g and e.
(4) The forward direction follows immediately from $\varepsilon < g$ for all $g \in \mathcal{U}_{>0}$.
For the backward direction, plugging in α^* into C_w^* results in bounds on ε, which are, by the semantics of $<$ on $\mathcal{U}[\varepsilon]$, only positive upper bounds or non-positive lower bounds. Thus, g can be chosen as any value smaller than the smallest upper bound (or as any positive value if no such bound exists).

□

While this formalism seems natural, there are some pitfalls in understanding which choices for $V \in \mathfrak{B}_{C_w^*}$ are accepted by Theorem 3.10. These are illustrated in the following example.

Example 3.11. Consider the system depicted in Figure 4a. Clearly, either $\{c_1, c_2\}$, $\{c_2, c_3\}$ or $\{c_1, c_3\}$ can be chosen for inducing the vertex, which is also called a *degenerate solution.*

Figure 4b depicts the system after replacing c_1 by a strict inequality respectively its weakened version $(c_1)_w$. Now, $\{c_1, c_2\}$ and $\{c_1, c_3\}$ represent different vertices while $\{c_2, c_3\}$ induces no solution.

Figure 4c shows the system after replacing all constraints by strict ones. Though the illustration suggests that any combination of constraints induces the (same) vertex, this is not true in general. As illustrated in Figure 4d, if we scale c_1 by the factor $\frac{1}{2}$, only $\{c_1, c_2\}$ and $\{c_1, c_3\}$ induce vertices while selecting $\{c_2, c_3\}$ violates c_1. While this seems counter-intuitive at first sight, Theorem 3.10 guarantees the existence of a satisfying vertex w.r.t. infinitesimal arithmetic if and only if the original system is satisfiable.

3.4 Equal and not-equal constraints

So far, we covered the relations $\leq$ and $<$. As already mentioned, $\geq$, $>$ and $=$ can be equivalently expressed by $\leq$ and $<$, even though for better efficiency, in practice these relations are rather handled separately.

Dealing with disequalities $p \neq b$ is a bit more involved. Graphically, each disequality plits the solution set by a hyperplane into two halves, sharing one open boundary but in different directions. Algorithmically, both possibilities $p < b$ and $p > b$ need to be considered in order to achieve a conclusive answer to the satisfiability question. This can be implemented by case splitting, checking both branches individually; optimizations could use e.g. explanations of unsatisfiability in certain branches in order to generalize the result to other branches, or postpone branching in the computations as far as possible by computing first without disequalities and considering $\neq$ constraints belatedly.

4 DISCUSSION AND RELATED WORK

Theorem 3.10 admits the extension of any algorithm relying on the fundamental theorem of linear programming (Theorem 2.1) for strict inequalities: The original system needs to be transformed into its weak version as in Definition 3.1. The arithmetic operations and evaluation functions need to be extended for an infinitesimal value ε with the semantics given in Definition 3.9. From there on, Theorem 3.10 guarantees that everything works analogously as for the case with weak inequalities and thus serves as a drop-in extension.

4.1 Combinatorial complexity through strict constraints

Example 3.11 shows that systems containing strict inequalities might be more complex than their counterparts containing only weak relations: While for the latter, any choice of the non-basis yields a satisfying vertex, this is not necessarily the case if some of these constraints are made strict. In fact, one could construct bad examples that are trivial after replacing all strict relation symbols by weak ones but harder to solve otherwise. However, this blow-up is not greater than adding an additional variable to the problem.

4.2 Applications

Our approach enables decision procedures for weak inequalities to handle also strict inequalities in the context of SMT solving, without further modifications.

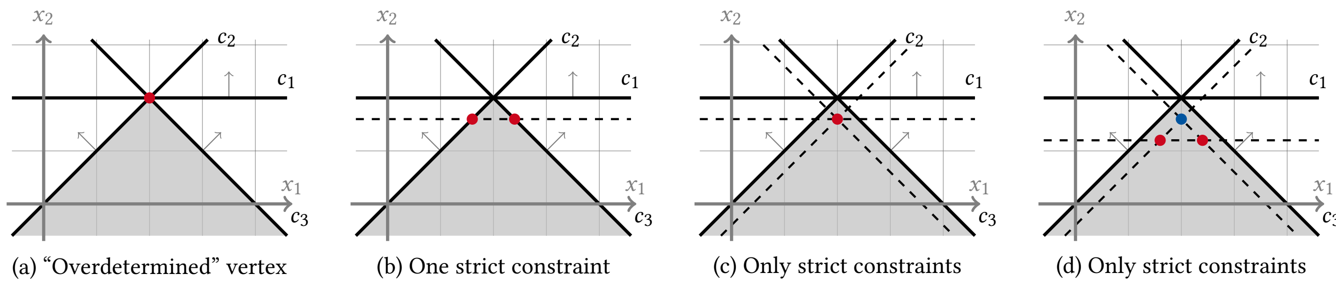

x_2
c_2
c_1
x_1
c_3
(a) "Overdetermined" vertex
(b) One strict constraint
(c) Only strict constraints
(d) Only strict constraints

Figure 4: Illustration of Theorem 3.10

4.2.1 Simplex in SMT solving. The classical approach for SMT solving is the *DPLL(T)* framework. A *SAT solver* searches for solutions of the *Boolean abstraction* of the problem where every constraint is replaced by a propositional variable, while a *theory solver* is employed regularly for checking the consistency of a (possibly partial) Boolean solution with the underlying theory. Thus, the input to the theory solver is a set of constraints containing all input constraints whose abstraction variable is assigned to true and the negation of all input constraints whose abstraction variable is set to false. The theory either finds the current solution consistent, that is, all passed constraints are satisfiable together, or an (as small as possible) set of constraints that are unsatisfiable together is returned to the SAT solver, which adds a formula excluding this selection to the input formula.

For checking the consistency of linear constraints, there are two common methods: The *Fourier-Motzkin elimination* [15, 23] is a quantifier elimination method; while its original formulation only assumes weak constraints, the extension for strict constraints is rather trivial. However, due to its doubly exponential complexity, it is not suitable for applications in the DPLL(T) context where the state-of-the-art method is based on the *general Simplex* algorithm [11]. Here, we give a simplified intuition using a notation that is consistent with the notation of this paper.

Given a system C of *weak* linear constraints, the algorithm can be seen as a heuristic that solves a sequence of equation systems $\tilde{V}_1, \ldots, \tilde{V}_k$ where $V_i \subseteq \mathfrak{B}_C$. If in a step i it holds $\mathrm{sol}(\tilde{V}_i) \models C$, then satisfiability of C is proven. Otherwise the algorithm improves the given solution by choosing $\tilde{V}_{i+1} = \tilde{V}_i \setminus \{\tilde{c}_j\} \cup \{\tilde{c}_{j'}\}$ for some $j \neq j'$ such that $V_{i+1} \subseteq \mathfrak{B}_C$ still holds. Additionally, $c_{j'}$ is chosen such that it is violated in step i and $c_j, c_{j'}$ are satisfied in step $i+1$. If no such constraint is found, unsatisfiability is returned.

The improvement of a solution is also called a *pivot operation*, the elements of $\tilde{V}_i$ are called *non-basis* and the elements in $C \setminus V_i$ are called *basis*. By representing the equation system in a *tableau*, these pivot operations can be implemented efficiently avoiding the need to solve the whole equation system $\tilde{V}_{i+1}$ from scratch. By a heuristic selecting the pivots, it is made sure that no selection of the non-basis is visited again, i.e. $\tilde{V}_i \neq \tilde{V}_j$ for all $i \neq j$; together with Theorem 2.1, this implies completeness of the algorithm.

The splits caused by not-equal constraints can be deferred as long as possible by only deciding for a case when the constraint $p \neq b$ is violated by the current solution and is picked for the next pivot step.

4.2.2 An adaption of the general Simplex method. The formulation presented in this paper was needed in Nalbach's master thesis on an adaption [24] of the general Simplex:

The idea in the thesis was to move the selection of the non-basis V into the SAT solver and to pass theory information as additional lemmas to the SAT solver allowing for Boolean learning and reasoning about the vertex selection. A theory call then not only consists of a set of constraints to be satisfied, but also an encoding of the vertex selection, thus the system $\tilde{V} \cup C$ is solved by Gaussian elimination.

The general Simplex is able to transfer some theory-specific knowledge between theory calls by starting from the tableau from the previous call, keeping progress on small changes to the input. This approach comes to its limits on instances with complex Boolean structure, while the novel approach in the master's thesis learned combinatorial properties of the theory across theory calls.

Although the new approach did not compete with the general Simplex in practise, the view of the Simplex algorithm as sequence of equation systems as described above inspired the formulation of the main theorem of this paper.

Our formulation of the fundamental theorem allows the extension of the general Simplex method as well as the adapted method for strict inequalities. Similarly, further novel methods for solving QF_LRA can be built on its foundation due to its generality.

4.2.3 Exact solutions for linear programming. Most linear programming solvers use floating point arithmetic for efficiency but at the cost of precision; this makes the study of strict inequalities uninteresting. However, there exist solvers that refine solutions [16] or even employ exact rational arithmetic [3, 9] to increase precision. For scenarios where high or exact precision is required, the presented construction could be of interest.

4.3 Alternative approaches

The presented transformation serves as an almost drop-in solution for extending existing algorithms for strict inequalities. Here, we present two alternative approaches that are also feasible, but require a more extensive adaption of the algorithms and thus leaving some questions open. Furthermore note that as both alternatives described below also introduce an additional variable ε, the increase in combinatorial complexity is the same as the method presented in the previous section.

4.3.1 Transformation to maximization problem. Lemma 3.2 and Lemma 3.3 already yield an obvious transformation of a system

C of linear constraints over $(\mathcal{U}, <)$ in variables X to the linear program max ε *subject to* C_w.

C is satisfiable if and only if there exists a solution for C_w with a strictly positive value for ε, that is, the outcome of the linear program is positive.

4.3.2 Allow arbitrary positive values for ε. Our approach interprets ε as an infinitesimal, which can be seen as a variable that can take arbitrarily small values. Alternatively, one could interpret it as a variable that can take any strictly positive value.

This can be achieved using a weaker version of Theorem 3.6: A system C is satisfiable iff there exists a subset $V \subseteq C_w$ such that $|V| = \text{rank}(C)$ and $P_V \cup \{\varepsilon\}$ is linearly independent such that

$$\exists g \in \mathcal{U}_{>0}.\ \exists \alpha : X \to \mathcal{U}.\ \alpha[\varepsilon \mapsto g] \models \tilde{V} \cup C_w.$$

Note that consistency of a system $\tilde{V} \cup C_w$ can be checked by application of Gaussian variable elimination. The resulting system $C_{\tilde{V}}$ then only contains ε as a single variable, its constraints thus are weak lower and upper bounds on ε. Thus $\tilde{V} \cup C_w$ is consistent if and only if there exists a positive value g for satisfying these bounds.

4.4 Differences to King's proof

The formulation of King's work focusses on the extension of the general Simplex method and thus is only applicable to this specific method. This paper gives a more general formulation as an extension of (parts of) the fundamental theorem of linear programming; while the two formulations are identical for the general Simplex, our generalization can be applied to other methods such as given in Section 4.2.2 without crucial modifications to the method.

Although the idea of King's proof can be applied also to the new formulation, our proof employs a different idea than King's proof [20]: Both proofs use the same transformation of a system C of linear constraints to a system C^*_w with only weak constraints and containing an infinitesimal ε with a special interpretation as given in Definitions 3.1 and 3.9. Also, both use that any solution for C^*_w can be transformed to a solution for C.

We gave a constructive proof stating that if a system C of linear constraints is satisfiable, then there also exists a solution for C^*_w which is in particular a vertex of the solution polyhedron (under application of the fundamental theorem of linear programming). The latter is the crucial step in the proof, as this enables restricting the search to vertex candidates analogously to the fundamental theorem of linear programming.

King [20] proves the contraposition: He applies *Farkas' lemma* [13] on the transformation C^*_w, implying that if the Simplex algorithm detects a conflict, then C^*_w does not have a solution; then by construction and the semantics of ε, the original system C is also unsatisfiable.

Our constructive proof gives insight to the relation between a system C and its transformation C^*_w which is the main contribution of this paper.

5 CONCLUSION

The presented transformation is an elegant way to extend algorithms relying on the fundamental theorem of linear programming for strict inequalities, which is commonly used for applications in SMT solving. While there was already a proof for this method, we gave an additional proof that shows the nature of the problem and hopefully helps for deeper understanding its implications.

ACKNOWLEDGMENTS

We thank the anonymous reviewers for their feedback on an earlier version of this paper. This work has been supported by the DFG RTG 2236 "UnRAVeL".

REFERENCES

[1] Clark Barrett, Christopher L Conway, Morgan Deters, Liana Hadarean, Dejan Jovanović, Tim King, Andrew Reynolds, and Cesare Tinelli. 2011. CVC4. In *International Conference on Computer Aided Verification.* 171–177.

[2] Nikolaj Bjørner and Anh-Dung Phan. 2014. *vZ*-Maximal Satisfaction with Z3. (2014). Symbolic Computation in Software Science.

[3] William Cook, Thorsten Koch, Daniel E Steffy, and Kati Wolter. 2011. An exact rational mixed-integer programming solver. In *International Conference on Integer Programming and Combinatorial Optimization.* 104–116.

[4] Florian Corzilius, Gereon Kremer, Sebastian Junges, Stefan Schupp, and 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 (LNCS, Vol. 9340).* 360–368.

[5] George B Dantzig. 1990. *Origins of the Simplex Method.* Association for Computing Machinery, 141–151.

[6] George B Dantzig. 1998. *Linear Programming and Extensions.* Princeton University Press.

[7] Leonardo de Moura and Nikolaj Bjørner. 2008. Z3: An efficient SMT solver. In *International conference on Tools and Algorithms for the Construction and Analysis of Systems.* 337–340.

[8] Leonardo de Moura and Grant O Passmore. 2013. Computation in real closed infinitesimal and transcendental extensions of the rationals. In *International Conference on Automated Deduction.* 178–192.

[9] Marcel Dhiflaoui, Stefan Funke, Carsten Kwappik, Kurt Mehlhorn, Michael Seel, Elmar Schomer, Ralph Schulte, and Dennis Weber. 2003. Certifying and Repairing Solutions to Large LPs, How good are LP-solvers?. In *SODA*, Vol. 3. 255–256.

[10] Bruno Dutertre and Leonardo de Moura. 2006. A fast linear-arithmetic solver for DPLL (T). In *International Conference on Computer Aided Verification (LNCS, Vol. 4144).* 81–94.

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

[12] Ioannis Z Emiris and John F Canny. 1995. A general approach to removing degeneracies. *SIAM J. Comput.* 24, 3 (1995), 650–664.

[13] Julius Farkas. 1902. Theorie der einfachen Ungleichungen. *Journal für die reine und angewandte Mathematik* 1902, 124 (1902), 1–27.

[14] Institute for Numerical Analysis (US). 1951. *Problems for the Numerical Analysis of the Future.* US Government Printing Office.

[15] Jean BJ Fourier. 1825. Analyse des travaux de l'Académie Royale des Sciences pendant l'année 1824. *Partie mathématique* (1825).

[16] Ambros M Gleixner, Daniel E Steffy, and Kati Wolter. 2016. Iterative refinement for linear programming. *INFORMS Journal on Computing* 28, 3 (2016), 449–464.

[17] Harvey J Greenberg. 1996. Consistency, redundancy, and implied equalities in linear systems. *Annals of Mathematics and Artificial Intelligence* 17, 1 (1996), 37–83.

[18] Antti EJ Hyvärinen, Matteo Marescotti, Leonardo Alt, and Natasha Sharygina. 2016. OpenSMT2: An SMT solver for multi-core and cloud computing. In *International Conference on Theory and Applications of Satisfiability Testing.* 547–553.

[19] Leonid V Kantorovich. 1960. Mathematical methods of organizing and planning production. *Management science* 6, 4 (1960), 366–422.

[20] Tim King. 2014. *Effective algorithms for the satisfiability of quantifier-free formulas over linear real and integer arithmetic.* Ph.D. Dissertation. Courant Institute of Mathematical Sciences New York.

[21] Jean-Louis Lassez and Ken McAloon. 1989. Independence of negative constraints. In *Colloquium on Trees in Algebra and Programming.* 19–27.

[22] David G Luenberger and Yinyu Ye. 1984. *Linear and Nonlinear Programming.* Springer.

[23] Theodore S Motzkin. 1936. *Beiträge zur Theorie der linearen Ungleichungen.* Azriel.

[24] Jasper Nalbach. 2020. *A novel adaption of the simplex algorithm for linear real arithmetic.* Master's thesis. RWTH Aachen University. https://doi.org/10.18154/RWTH-2021-04303

[25] Igor R Shafarevich and Alexey O Remizov. 2012. *Linear Algebra and Geometry.* Springer.

[26] Cornelis van de Panne and Farhood Rahnama. 1985. The first algorithm for linear programming: An analysis of Kantorovich's method. *Economics of Planning* 19, 2 (1985), 76–91.