



HAL
open science

Context-Aided Variable Elimination for Requirement Engineering

Inigo Incer, Albert Benveniste, Richard M Murray, Alberto Sangiovanni-Vincentelli, Sanjit A Seshia

► **To cite this version:**

Inigo Incer, Albert Benveniste, Richard M Murray, Alberto Sangiovanni-Vincentelli, Sanjit A Seshia. Context-Aided Variable Elimination for Requirement Engineering. 2023. hal-04807267

HAL Id: hal-04807267

<https://hal.science/hal-04807267v1>

Preprint submitted on 27 Nov 2024

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

Context-Aided Variable Elimination for Requirement Engineering

Inigo Incer Albert Benveniste Richard M. Murray Alberto Sangiovanni-Vincentelli Sanjit A. Seshia
CMS, Caltech *INRIA/IRISA* *CDS, Caltech* *EECS, UC Berkeley* *EECS, UC Berkeley*
 Pasadena, CA, USA Rennes, France Pasadena, CA, USA Berkeley, CA, USA Berkeley, CA, USA

Abstract—Deriving system-level specifications from component specifications usually involves the elimination of variables that are not part of the interface of the top-level system. This paper presents algorithms for eliminating variables from formulas by computing refinements or relaxations of these formulas in a context. We discuss a connection between this problem and optimization and give efficient algorithms to compute refinements and relaxations of linear inequality constraints.

Index Terms—automated reasoning, deduction, specifications, variable elimination

I. INTRODUCTION

In the setting of requirement engineering using assume-guarantee specifications [1], [2], [3], we come across the need to eliminate variables from a formula by computing refinements or relaxations in a context. Let ϕ be a formula containing some variables that must be eliminated. These will be called *irrelevant variables*, and the set of such variables will be denoted Y . In order to carry out the elimination, suppose we can use information from a set of formulas Γ called the *context*.

We will consider the problems of synthesizing missing formulas in the expressions

$$\Gamma \wedge ? \models \phi \quad \text{and} \quad \Gamma \wedge \phi \models ?$$

such that the result lacks irrelevant variables.

We will call the first problem *antecedent synthesis*, and the second *consequent synthesis*. If ψ is a solution to the antecedent synthesis problem, we will say that ψ is a Y -antecedent (or a Y -refinement) of ϕ in the context Γ . If ψ is a solution to the consequent synthesis problem, we will say that ψ is a Y -consequent (or a Y -relaxation) of ϕ in the context Γ . This problem appears in requirement engineering in the following situations.

Figure 1 shows two components connected in series, M_1 and M_2 . The first has input i and output o , and the second has input o and output o' . Each component comes with its assumptions and guarantees. The natures of M_1 and M_2 are left abstract; they could be routines executing in order, or they could be physical systems that interact through their input and

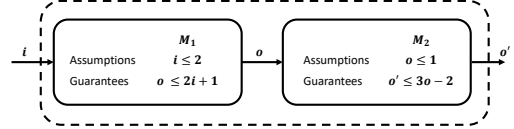


Fig. 1: Two components connected in series. We wish to compute the specification of the top-level system composed of these two elements.

output ports. Our problem is to obtain a specification for the entire system using the specifications of the subsystems in such a way that only the top-level input and output variables i and o' appear in the final answer. In other words, the top-level specification should not mention the internal variable o .

We would like to operate the system in such a way that the assumptions of the two components hold. This would mean that we can rely on the two subsystems to deliver their guarantees. Thus, the top-level system should assume $(i \leq 2) \wedge (o \leq 1)$. This cannot be the top-level specification because the second formula involves the *irrelevant variable* o . We would like to find a term only depending on i that somehow ensures that the assumptions $o \leq 1$ of M_2 are satisfied. For this, we make use of the knowledge that M_1 guarantees $o \leq 2i + 1$ when $i \leq 2$. We want to transform the constraint $o \leq 1$ into a constraint ψ on the input i with the property that, given the guarantees of M_1 , ψ implies $o \leq 1$. That is, this new constraint should satisfy $\psi \wedge (o \leq 2i + 1) \rightarrow (o \leq 1)$, which means that ψ should be a refinement (an antecedent) of $o \leq 1$ in the context of the guarantees of M_1 . We observe that $\psi: i \leq 0$ satisfies this requirement. Thus, we transform the term $o \leq 1$ into the term $i \leq 0$. The top-level assumptions become $i \leq 0$. We can verify that these top-level assumptions ensure that subsystems M_1 and M_2 have their assumptions met.

Similarly, the guarantees for the system are $(o \leq 2i + 1) \wedge (o' \leq 3o - 2)$. Again, the variable o is not welcome in the final answer, giving us two options: we could eliminate both terms and have no guarantees—which is right, but not useful—or we could relax (compute the consequent of) one of the terms in the context of the other term. We find out, for example, that $(o \leq 2i + 1) \wedge (o' \leq 3o - 2) \rightarrow (o' \leq 6i + 1)$. The constraint $o' \leq 6i + 1$ is an acceptable promise for the system

This work was supported by the DARPA LOGiCS project under contract FA8750-20-C-0156 and by NSF and ASEE through an eFellows postdoctoral fellowship.

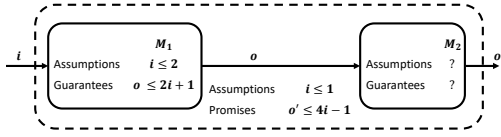


Fig. 2: Two components connected in series. We are given the specification of the top-level system and the specification of M_1 . The problem is to find the pre/post conditions of M_2 in order to obtain the given system-level specifications.

specification.

By computing antecedents and consequents, we concluded that the top-level system guarantees $o' \leq 6i + 1$ as long as the input satisfies $i \leq 0$.

This example shows that the computation of antecedents and consequents plays a key role in the identification of assume-guarantee pairs or pre/post conditions. One may be tempted to link antecedents to assumptions and consequents to guarantees. This is not always so. Figure 2 show a situation in which we again have two components connected in series, M_1 and M_2 , with inputs and outputs as before. Now we are given the top level assumptions and guarantees, and we also know the assumptions and guarantees of M_1 . The problem is to find the pre/post conditions of M_2 using this data.

We know that the top level assumes that $i \leq 1$. Under these assumptions, M_1 guarantees $o \leq 2i + 1$. The assumptions of M_2 should be met when the top-level system is operating within its assumptions. Thus, the assumptions of M_2 should be implied by the data $(i \leq 1) \wedge (o \leq 2i + 1)$. Since the assumptions of M_2 should only depend on o , we obtain the expression $o \leq 3$.

Now we look for the guarantees of M_2 , which we call ψ . The guarantees of M_1 and M_2 together must imply the top-level guarantees. Thus, we have the expression $\psi \wedge (o \leq 2i + 1) \rightarrow (o' \leq 4i - 1)$. In other words, ψ is an antecedent of $o \leq 2i + 1$ in the context $o' \leq 4i - 1$. We require ψ to only refer to variables o and o' and observe that $o' \leq 2o - 3$ is an acceptable promise.

We conclude that M_2 should assume $o \leq 3$ and promise $o' \leq 2o - 3$.

The examples just described motivate us to study automated mechanisms for the computation of antecedents and consequents of formulas in a given context with the objective of removing dependencies on irrelevant variables. We first consider this problem for general first-order formulas and then specialize to the situation when formulas are expressed as linear constraints in a context of linear inequalities. We provide efficient algorithms to address this problem. Our previous discussion shows that this problem is of relevance to requirement engineering.

II. COMPUTING ANTECEDENTS AND CONSEQUENTS IN FIRST-ORDER LOGIC

Suppose ϕ is a formula in first-order logic with free variables $x = (x_1, \dots, x_m)$ and $y = (y_1, \dots, y_n)$, and Γ a formula with free variables x, y , and $z = (z_1, \dots, z_o)$. As a matter of notation, when the free variables of a formula are understood, we will simply write the name of the formula, e.g., $\phi(x, y)$ is synonymous with ϕ . We let Y be the set of *irrelevant variables* that we want to eliminate from ϕ . Throughout this paper, the set of irrelevant variables is always $Y = \{y_i\}_i$, i.e., we are always interested in eliminating the y variables from ϕ .

Definition II.1. We say that a formula $\psi(x, z)$ with free variables $x = (x_1, \dots, x_m)$ and $z = (z_1, \dots, z_o)$ is a Y -antecedent of ϕ in the context Γ if

$$\Gamma(x, y, z) \wedge \psi(x, z) \models \phi(x, y).$$

We say that $\psi(x, z)$ is a Y -consequent of ϕ in the context Γ if

$$\Gamma(x, y, z) \wedge \phi(x, y) \models \psi(x, z).$$

Our objective is to eliminate the irrelevant variables y from ϕ by synthesizing a Y -antecedent or Y -consequent of ϕ in the context Γ .

This section contains the following results:

- 1) a characterization of the optimal solutions for Y -antecedent/consequent synthesis in general (Proposition II.1);
- 2) a characterization of the optimal solution to this problem when the context can be expressed as a conjunction of a formula that depends on irrelevant variables and a formula that does not (Proposition II.2);
- 3) a characterization of optimal solutions to this problem when ϕ monotonically depends on a function of y and does not depend on y in any other way (Proposition II.3); and
- 4) methods to synthesize Y -antecedents/consequents compositionally from the syntax of ϕ and Γ (Propositions II.4 and II.5).

We define two formulas obtained from ϕ , Γ , and Y . We then discuss an important property of these formulas.

Definition II.2. Given ϕ , Γ , and Y as above, let

$$\phi_\Gamma := \forall y. (\Gamma \rightarrow \phi)$$

and

$$\phi^\Gamma := \exists y. (\Gamma \wedge \phi).$$

Proposition II.1. A formula $\psi(x, z)$ is a Y -antecedent for ϕ in the context Γ if and only if $\psi \models \phi_\Gamma$.

A formula $\rho(x, z)$ is a Y -consequent for ϕ in the context Γ if and only if $\phi^\Gamma \models \rho$.

Proof. We have $\Gamma \wedge \psi \models \phi \Leftrightarrow \psi \models (\Gamma \rightarrow \phi) \Leftrightarrow \psi \models \phi_\Gamma$. Similarly, $\Gamma \wedge \phi \models \rho \Leftrightarrow \exists y. (\Gamma \wedge \phi) \models \rho$. \square

This result means that ϕ_Γ is the weakest Y -antecedent of ϕ in the context Γ . Conversely, ϕ^Γ is the strongest Y -consequent.

Lemma II.1. *The denotations of ϕ_Γ and ϕ^Γ are*

$$\llbracket \phi_\Gamma \rrbracket = \bigcap_{\substack{b \in \text{dom}(y) \\ \Gamma(x,b,z)}} \llbracket \phi(x,b,z) \rrbracket \text{ and } \llbracket \phi^\Gamma \rrbracket = \bigcup_{\substack{b \in \text{dom}(y) \\ \Gamma(x,b,z)}} \llbracket \phi(x,b,z) \rrbracket.$$

Proof. We compute

$$\begin{aligned} \llbracket \phi_\Gamma \rrbracket &= \llbracket \forall y (\Gamma \rightarrow \phi) \rrbracket = \bigcap_{b \in \text{dom}(y)} \llbracket \Gamma(x,b,z) \rightarrow \phi(x,b) \rrbracket \\ &= \bigcap_{\substack{b \in \text{dom}(y) \\ \Gamma(x,b,z)}} \llbracket \phi(x,b) \rrbracket. \end{aligned}$$

A similar reasoning applies to ϕ^Γ . \square

Proposition II.1 provides a universal characterization of Y -antecedents and consequents. We now state a result that allows us to synthesize these objects using partial information from the context.

Proposition II.2. *Suppose that the context Γ can be written as $\Gamma(x,y,z) = \Gamma_1(x,y,z) \wedge \Gamma_2(x,y,z)$. Then $\phi_\Gamma = \Gamma_1 \rightarrow \phi_{\Gamma_2}$ and $\phi^\Gamma = \Gamma_1 \wedge \phi_{\Gamma_2}$.*

Proof. From Definition II.1, we have

$$\begin{aligned} \phi_\Gamma &= \forall y. ((\Gamma_1 \wedge \Gamma_2) \rightarrow \phi) \\ &= \forall y. (\Gamma_1(x,z) \rightarrow (\Gamma_2(x,y,z) \rightarrow \phi(x,y))) \\ &= \Gamma_1 \rightarrow (\forall y. (\Gamma_2(x,y,z) \rightarrow \phi(x,y))) = \Gamma_1 \rightarrow \phi_{\Gamma_2} \end{aligned}$$

and

$$\begin{aligned} \phi^\Gamma &= \exists y. ((\Gamma_1(x,z) \wedge \Gamma_2(x,y,z)) \wedge \phi(x,y)) \\ &= \Gamma_1 \wedge (\exists y. (\Gamma_2(x,y,z)) \wedge \phi(x,y)) = \Gamma_1 \wedge \phi_{\Gamma_2}. \quad \square \end{aligned}$$

An immediately corollary of Proposition II.2 is the fact that $\phi_\Gamma \wedge \Gamma = \phi_{\Gamma_2} \wedge \Gamma$ and $\phi^\Gamma \wedge \Gamma = \phi_{\Gamma_2} \wedge \Gamma$. This means that ϕ_{Γ_2} and $\phi_{\Gamma_2}^\Gamma$ are optimal in the context Γ and thus can be used as the optimal Y -antecedents and consequents, respectively. The following example shows that quantifying over smaller contexts can yield preferable results.

Example II.1. *Suppose that ϕ and Γ are formulas in linear arithmetic such that $\phi(x,y) = (x \leq y)$ and $\Gamma(x,y,z) = \Gamma_1(x,z) \wedge \Gamma_2(x,y,z)$, where $\Gamma_1 = (z \leq 2)$ and $\Gamma_2 = (z \leq y)$. From Proposition II.1, the optimal Y -antecedent of ϕ in Γ is*

$$\begin{aligned} \phi_\Gamma &= \forall y. ((z \leq 2) \wedge (z \leq y) \rightarrow (x \leq y)) \\ &= ((z \leq 2) \rightarrow (x \leq z)). \end{aligned}$$

As we just discussed, we may as well use ϕ_{Γ_2} as the Y -antecedent of ϕ . This formula is

$$\phi_{\Gamma_2} = \forall y. ((z \leq y) \rightarrow (x \leq y)) = (x \leq z).$$

For requirement engineering, it is preferable to output ϕ_{Γ_2} instead of ϕ_Γ because it is syntactically simpler. \blacksquare

Now we discuss an optimization formulation of antecedent/consequent synthesis.

Proposition II.3. *Suppose that ϕ can be expressed as $\phi(x,g(y))$, where ϕ is monotonic in the second argument. Define*

$$g^-(a,c) = \begin{cases} \text{minimize} & g(b) \\ \text{subject to} & [x := a, y := b, z := c] \models \Gamma \end{cases}$$

and

$$g^+(a,c) = \begin{cases} \text{maximize} & g(b) \\ \text{subject to} & [x := a, y := b, z := c] \models \Gamma. \end{cases}$$

Then we have $\phi(x,g^-(x,z)) \models \phi_\Gamma$ and $\Gamma \wedge \phi_\Gamma \models \phi(x,g^-(x,z))$. Similarly, $\phi^\Gamma \models \phi(x,g^+(x,z))$ and $\phi(x,g^+(x,z)) \wedge \Gamma \models \phi^\Gamma$.

Proof. If $[x := a, y := b, z := c] \not\models \Gamma$ for all $b \in \text{dom}(y)$, then $[x := a, z := c] \models \phi_\Gamma$. Otherwise, $\bigcap_{\substack{b \in \text{dom}(y) \\ \Gamma(a,b,c)}} \llbracket \phi(a,g(b)) \rrbracket = \llbracket \phi(a,g^-(a,c)) \rrbracket$. The second part is proved similarly. \square

From Propositions II.1 and II.3, we know that $\phi(x,g^-(x,z))$ and $\phi(x,g^+(x,z))$ are, respectively, a Y -antecedent and a Y -consequent of ϕ in the context Γ . Moreover, $\phi(x,g^-(x,z))$ and $\phi(x,g^+(x,z))$ have the same denotations as the optimal solutions ϕ_Γ and ϕ^Γ , respectively, in the context Γ . Thus, we regard $\phi(x,g^-(x,z))$ and $\phi(x,g^+(x,z))$ as optimal.

Example II.2. *Suppose we want to compute an antecedent of the formula*

$$\phi: (p \wedge q) \vee r$$

in the context $\Gamma: s \rightarrow q$, where q is an irrelevant variable. Let $g(q) = q$ and $\phi(x,g(q)) = (p \wedge q) \vee r$. Then ϕ is monotonic in its last argument. To apply Proposition II.3, we compute g^- :

$$\begin{aligned} g^-(a,c) &= \begin{cases} \text{minimize} & q \\ \text{subject to} & [p := a, q := b, s := c] \models s \rightarrow q \end{cases} \\ &= c. \end{aligned}$$

By Proposition II.3, we conclude that $\phi(p,g^-(p,s))$ is a Y -antecedent of ϕ in the given context, i.e., we get the antecedent

$$(p \wedge s) \vee r.$$

In contrast, we compute $\phi_\Gamma: \forall y. ((s \rightarrow q) \rightarrow ((p \wedge q) \vee r)) = (p \vee r) \wedge (s \vee r)$. \blacksquare

In Example II.2, ϕ_Γ and $\phi(x,g^-(x,z))$ match. Yet, observe that the latter immediately yields a result in a syntactic form which is closer to the original ϕ . This happens because this expression is obtained by simply replacing q with $g^-(x,z)$ in ϕ . Having results which are syntactically similar to the original expressions is important in requirement engineering, as the syntax of requirements entered by users has a close connection to the semantics they want to express.

A. Compositional results

We now express ϕ and Γ as $\phi = \bigvee_j \bigwedge_i \phi_i^j(x, y)$, where the $\phi_i^j(x, y)$ are clauses (i.e., terms formed from atomic formulas and Boolean connectives), and $\Gamma = \bigvee_k \Gamma^k(x, y, z)$, where each $\Gamma^k(x, y, z)$ is a conjunction of clauses. Instead of synthesizing an antecedent or consequent for ϕ directly, we will seek methods to do this compositionally.

Proposition II.4. *Let $\tilde{\phi}_\Gamma := \bigvee_j \bigwedge_{i,k} \forall y. (\Gamma^k \rightarrow \phi_i^j)$ and $\tilde{\phi}^\Gamma := \bigvee_{k,j} \bigwedge_i \exists y. (\Gamma^k \wedge \phi_i^j)$. $\tilde{\phi}_\Gamma$ is an antecedent and $\tilde{\phi}^\Gamma$ a consequent of ϕ in the context Γ .*

Proof. $\tilde{\phi}_\Gamma = \bigvee_j \bigwedge_{i,k} \forall y. (\Gamma^k \rightarrow \phi_i^j) = \bigvee_j \forall y. (\Gamma \rightarrow \bigwedge_i \phi_i^j) \models \forall y. \bigvee_j (\Gamma \rightarrow \bigwedge_i \phi_i^j) = \phi_\Gamma$. By applying Proposition II.1, we proved the first part. We also have $\phi^\Gamma = \exists y. (\Gamma \wedge \phi) = \exists y. \bigvee_{k,j} \bigwedge_i (\Gamma^k \wedge \phi_i^j) = \bigvee_{k,j} \exists y. \bigwedge_i (\Gamma^k \wedge \phi_i^j) \models \bigvee_{k,j} \bigwedge_i \exists y. (\Gamma^k \wedge \phi_i^j) = \tilde{\phi}^\Gamma$, which shows the second part after applying Proposition II.1. \square

The compositional result of Proposition II.4 allows us to focus on the case in which ϕ is a clause and Γ is a conjunction of clauses. We shall assume this from now on.

To further exploit compositionality, express ϕ as

$$\phi(x, y) = \phi(x, g_1(y), \dots, g_p(y)), \quad (1)$$

where ϕ is required to be monotonic in all arguments, except the first. We assume that our language allows any clause to be expressed in this way. This is true for linear arithmetic and for real arithmetic with a ReLu (rectified linear unit) function.

Example II.3. *Suppose ϕ is the real-arithmetic formula $x^2 - xy + y^2 \leq 0$. This is equivalent to $x^2 - r(x)y + r(-x)y + y^2 \leq 0$, where r is the ReLu function. We can thus write ϕ as $\phi(x, g_1(y), g_2(y), g_3(y))$ with $g_1(y) = y$, $g_2(y) = -y$, and $g_3(y) = -y^2$. \blacksquare*

We now study how to exploit the structure (1) to compute Y -antecedents and consequents for ϕ in Γ .

Proposition II.5. *Let*

$$g_i^-(a, c) = \begin{cases} \text{minimize} & g_i(b) \\ \text{subject to} & [x := a, y := b, z := c] \models \Gamma \end{cases}$$

and

$$g_i^+(a, c) = \begin{cases} \text{maximize} & g_i(b) \\ \text{subject to} & [x := a, y := b, z := c] \models \Gamma. \end{cases}$$

The formulas $\phi(x, g_1^-(x, z), \dots, g_p^-(x, z))$ and $\phi(x, g_1^+(x, z), \dots, g_p^+(x, z))$ are, respectively, y -antecedents and consequents of ϕ in Γ .

Proof. From Proposition II.1, we have

$$\llbracket \phi_\Gamma \rrbracket = \bigcap_{\substack{b \in \text{dom}(y) \\ \Gamma(x, b, z)}} \llbracket \phi \rrbracket \supseteq \llbracket \phi(x, g_1^-(x, z), \dots, g_p^-(x, z)) \rrbracket.$$

We conclude that $\phi(x, g_1^-(x, z), \dots, g_p^-(x, z)) \models \phi_\Gamma$. Proposition II.1 yields the first part. The second part is proved similarly. \square

Proposition II.5 tells us that we can independently optimize over the monotonic functions composing ϕ to compute Y -antecedents and consequents. This result does not yield the optimality guarantees of Proposition II.3, but it allows us to compute antecedents/consequents compositionally.

Example II.4. *Continuing Example II.3, suppose we want to compute a consequent of ϕ in the context $\Gamma: (y \leq 2) \wedge (1 \leq y)$. We apply Proposition II.5 by optimizing over the g_i separately. We obtain $g_1^+(x) = 2$, $g_2^+(x) = -1$, and $g_3^+(x) = -1$. The formula $\phi(x, g_1^+(x), g_2^+(x), g_3^+(x)) = x^2 - 2r(x) - r(-x) - 1 \leq 0$ is a Y -consequent of ϕ in Γ .*

III. LINEAR INEQUALITY CONSTRAINTS

We apply the results of Section II to the situation when atoms are linear inequalities, or polyhedral constraints. Polyhedral constraints are an intuitive formalism for writing specifications for complex systems, as they allow us to place piecewise linear bounds on quantities of interest.

We consider formulas of the form $\bigvee_j \bigwedge_i \phi_i^j(x, y)$, where $\phi_i^j(x, y)$ are atoms, and contexts of the form $\bigvee_k \Gamma^k(x, y, z)$, where $\Gamma^k(x, y, z)$ are conjunctions of atoms.

Due to Proposition II.4, we will focus our attention on algorithms for the efficient computation of antecedents and consequents when ϕ is an atom and Γ a conjunction of atoms. ϕ will have the form

$$\phi: \sum_{i=1}^m p_i x_i + \sum_{i=1}^n q_i y_i + r \leq 0,$$

where r and the p_i and q_i are constants. The set of irrelevant variables to be eliminated is $Y = \{y_i\}_i$. The context Γ is a set of linear inequalities of the form

$$\Gamma = \left\{ \sum_{j=1}^m \alpha_{ij} x_i + \sum_{j=1}^n \beta_{ij} y_i + \sum_{j=1}^o \gamma_{ij} z_i + K_i \leq 0 \right\}_{i=1}^N,$$

where the K_j , α_{ij}^j , β_{ij}^j , and γ_{ij}^j are constants.

Let $A = (\alpha_{ij}) \in \mathbb{R}^{N \times m}$, $B = (\beta_{ij}) \in \mathbb{R}^{N \times n}$, $C = (\gamma_{ij}) \in \mathbb{R}^{N \times o}$, $K \in \mathbb{R}^N$, $p \in \mathbb{R}^m$, and $q \in \mathbb{R}^n$. We also let $x = (x_i)$, $y = (y_i)$, $z = (z_i)$ be m -, n -, and o -dimensional vectors of variables, respectively.

Our problem is to eliminate the y variables from

$$\phi: p^\top x + q^\top y + r \leq 0 \quad (2)$$

using the context

$$\Gamma: Ax + By + Cz + K \leq 0 \quad (3)$$

by computing Y -antecedents/consequents.

Let $b(x, z) = -K - Ax - Cz$. We obtain the following corollary from Proposition II.3.

Corollary III.1. Let ϕ and Γ be as above. Let

$$g^-(x, z) = \begin{cases} \text{minimize} & -q^\top y \\ \text{subject to} & By \leq b(x, z) \end{cases} \quad (4)$$

and

$$g^+(x, z) = \begin{cases} \text{maximize} & -q^\top y \\ \text{subject to} & By \leq b(x, z). \end{cases} \quad (5)$$

Then the formula $p^\top x - g^-(x, z) \leq r$ is an optimal Y -antecedent of ϕ in the context Γ and $p^\top x - g^+(x, z) \leq r$ is an optimal Y -consequent of ϕ in the context Γ .

Example III.1. Suppose we wish to eliminate variables y_1 and y_2 from $2x + y_1 - 2y_2 \leq 5$ through antecedent computation, using the context $\{x - 2y_1 + y_2 + z \leq 1, 3y_1 - 4y_2 \leq 6\}$. We compute

$$\begin{aligned} g^-(x, z) &= \begin{cases} \text{minimize} & -(y_1 - 2y_2) \\ \text{subject to} & x - 2y_1 + y_2 + z \leq 1 \\ & 3y_1 - 4y_2 \leq 6 \end{cases} \\ &= -\left(4 - \frac{2}{5}(x + z)\right). \end{aligned}$$

The antecedent formula is $2x + 4 - \frac{2}{5}(x + z) \leq 5$, which becomes $8x - 2z \leq 5$. ■

Example III.2. Suppose we wish to eliminate variables y_1 and y_2 from $x + 5y_1 - 2y_2 \leq 5$ by the computation of a consequent, using the context $\{x - 2y_1 + y_2 + z \leq 1, 3y_1 - 4y_2 \leq 6\}$. We compute

$$\begin{aligned} g^+(x, z) &= \begin{cases} \text{maximize} & -(5y_1 - 2y_2) \\ \text{subject to} & x - 2y_1 + y_2 + z \leq 1 \\ & 3y_1 - 4y_2 \leq 6 \end{cases} \\ &= -\left(-4 + \frac{14}{5}(x + z)\right). \end{aligned}$$

The consequent is $x - 4 + \frac{14}{5}(x + z) \leq 5$, or $19x + 14z \leq 45$. ■

A. Solving the symbolic optimization problems

Corollary III.1 provides an explicit expression to compute optimal Y -antecedents and consequents. The next issue we face is the computation of (4) and (5). Both are linear programs, but their solutions are symbolic due to the presence of $b(x, z)$. We observe that if we have a context Γ' such that $\Gamma = \Gamma' \wedge \Gamma''$, and if ϕ' is a Y -antecedent/consequent of ϕ in the context Γ' then ϕ' is also a Y -antecedent/consequent in the context Γ . First, we will discuss conditions required for solving linear programs with symbolic constraints when the context has as many constraints as optimization variables (i.e., when $N = n$). Then we will discuss approaches for selecting from Γ a set of formulas that meets these requirements. We consider two selection criteria: a method based on positive solutions to linear equations and a method based on linear programming.

1) *Optimization in a subset of the context:* A linear program achieves its optimal value on the boundary of its constraints. If the context Γ contains N constraints and is a bounded polyhedron, then the optimal value of the linear program will occur at one of the $\binom{N}{n}$ possible vertices. We will look for ways to choose n constraints from Γ such that the optimization problems achieve optimal values at the vertex determined by those n constraints. First, we focus on solving symbolic LPs when the context contains n constraints. The following definition will be useful:

Definition III.1. Let $M \in \mathbb{R}^{n \times n}$ and $\nu \in \mathbb{R}^n$. We say that (M, ν) is a refining pair if M is invertible and $(M^\top)^{-1}\nu$ has nonnegative entries. We say that the pair (M, ν) is a relaxing pair if M is invertible and $-(M^\top)^{-1}\nu$ has nonnegative entries.

As the next result shows, these conditions are sufficient to solve the problems (4) and (5) when there are as many context formulas as irrelevant variables (i.e., when $N = n$). Suppose $J \subseteq \{1, \dots, N\}$ has cardinality n . We let $B_J = (\beta_{J_i, j})_{i, j=1}^n$ and $b_J = (b_{J_i})_{i=1}^n$ be the J -indexed rows of B and b , respectively.

Lemma III.1. Suppose (B_J, q) is a refining pair. Then

$$\begin{cases} \text{maximize} & q^\top y \\ \text{subject to} & B_J y \leq b_J(x, z) \end{cases} = q^\top B_J^{-1} b_J(x, z).$$

Suppose (B_J, q) is a relaxing pair. Then

$$\begin{cases} \text{minimize} & q^\top y \\ \text{subject to} & B_J y \leq b_J(x, z) \end{cases} = q^\top B_J^{-1} b_J(x, z).$$

Proof. Let (B_J, q) be a refining pair. We consider the first problem and its Lagrange dual (see [4], Section 5.2.1):

$$\begin{aligned} \text{primal} & \begin{cases} \text{minimize} & -q^\top y \\ \text{subject to} & B_J y \leq b_J(x, z) \end{cases} \\ \text{dual} & \begin{cases} \text{maximize} & -b_J^\top \lambda \\ \text{subject to} & B_J^\top \lambda - q = 0 \\ & \lambda \geq 0 \end{cases} \end{aligned}$$

The dual problem only admits the solution $\lambda^* = (B_J^\top)^{-1}q$ if $\lambda^* \geq 0$, which is the case, as (B_J, q) is a refining pair. Thus, the optimal value of the dual problem is $v^* = -q^\top (B_J^{-1}b_J)$. As strong duality holds for any linear program (see [4], Section 5.2.4), v^* is also the optimal value of the primal problem. The statement of the theorem follows.

Now suppose (B_J, q) is a relaxing pair. We consider the second problem and its dual:

$$\begin{aligned} \text{primal} & \begin{cases} \text{minimize} & q^\top y \\ \text{subject to} & B_J y \leq b_J(x, z) \end{cases} \\ \text{dual} & \begin{cases} \text{maximize} & -b_J^\top \lambda \\ \text{subject to} & B_J^\top \lambda + q = 0 \\ & \lambda \geq 0 \end{cases} \end{aligned}$$

The dual only admits the solution $\lambda^* = -(B_J^\top)^{-1}q$ if $\lambda^* \geq 0$, which is the case because (B_J, q) is a relaxing pair. The optimal value of the dual problem is $v^* = q^\top(B_J^{-1}b)$. Due to strong duality, v^* is also the optimal value of the primal problem. \square

As a consequence of Corollary III.1 and Lemma III.1, we obtain the following result:

Corollary III.2. *With all definitions as above, if (B_J, q) is a refining pair, then $p^\top x + q^\top B_J^{-1}b_J(x, z) \leq r$ is a Y -antecedent of $p^\top x + q^\top y \leq r$ in the context $By \leq b(x, z)$. If (B_J, q) is a relaxing pair, then $p^\top x + q^\top B_J^{-1}b_J(x, z) \leq r$ is a Y -consequent of $p^\top x + q^\top y \leq r$ in the context $By \leq b(x, z)$.*

Corollary III.2 gives explicit formulas for computing Y -antecedents/consequents of a formula in a context. This result is missing methods for computing J , the set of the indices of formulas in Γ , in such a way that it yields refining or relaxing pairs (B_J, q) , as needed. We consider two methods to identify J .

2) *Computing J by seeking positive solutions to linear equations:* Our first method is based on identifying constraints yielding linear systems of equations whose solutions are guaranteed to be nonnegative. We will use the following result.

Theorem III.1 (Kaykobad [5]). *Let $M = (\mu_{ij}) \in \mathbb{R}^{n \times n}$ and $\nu \in \mathbb{R}^n$. Suppose the entries of M are nonnegative, its diagonal entries are positive, the entries of ν are positive, and $\nu_i > \sum_{j \neq i} \mu_{ij} \frac{\nu_j}{\mu_{jj}}$ for all $i \leq n$. Then M is invertible and $M^{-1}\nu$ has positive entries.*

Definition III.2. *A pair (M, ν) , where $M = (\mu_{ij}) \in \mathbb{R}^{n \times n}$ and $\nu \in \mathbb{R}^n$, satisfying the conditions of Theorem III.1 is called a Kaykobad pair.*

We have the following result.

Lemma III.2. *Let Q be an $n \times n$ diagonal matrix whose i -th diagonal entry is $\text{SIGN}(q_i)$. Let $\bar{B}_J = B_J Q$ and $\bar{q} = Qq$. If $(\bar{B}_J^\top, \bar{q})$ is a Kaykobad pair, then (B_J, q) is a refining pair. If $(-\bar{B}_J^\top, \bar{q})$ is a Kaykobad pair, then (B_J, q) is a relaxing pair.*

Proof. Suppose $(\bar{B}_J^\top, \bar{q})$ is a Kaykobad pair. Then \bar{B}_J^\top is invertible. We have $B_J(\bar{B}_J Q)^{-1} = B_J Q(\bar{B}_J)^{-1} = I$ and $(\bar{B}_J Q)^{-1} B_J = Q(\bar{B}_J)^{-1}(B_J Q)Q = I$, so B_J is invertible. Moreover, we have

$$0 < (\bar{B}_J^\top)^{-1} \bar{q} = (Q B_J^\top)^{-1} (Q q) = (B_J^\top)^{-1} q,$$

which means that (B_J, q) is a refining pair.

If $(-\bar{B}_J^\top, \bar{q})$ is a Kaykobad pair, then \bar{B}_J^\top is invertible, which means that so is B_J . Moreover,

$$0 < -(\bar{B}_J^\top)^{-1} \bar{q} = -(Q B_J^\top)^{-1} (Q q) = -(B_J^\top)^{-1} q.$$

Thus, (B_J, q) is a relaxing pair. \square

Corollary III.2 and Lemma III.2 yield a method for computing Y -antecedents and consequents of formulas in a context. To use it, we must construct J such that (B_J, q) meets the corollary's conditions. We construct J by choosing n formulas

TABLE I: Execution time in seconds of Algorithm 1 for a given total number of variables, number of constraints in the context Γ , and number of irrelevant variables (or variables that need to be eliminated)

	Constraints in context	Total number of variables					
		5	10	15	20	25	30
2 irrelevant variables	5	0.24	0.41	0.64	0.88	1.11	1.43
	10	0.23	0.42	0.63	0.89	1.13	1.45
	20	0.23	0.45	0.67	0.93	1.18	1.49
	100	0.39	0.7	1.06	1.44	1.88	2.47
4 irrelevant variables	5	0.43	0.86	NA	1.74	NA	NA
	10	0.45	0.86	1.31	1.8	2.21	2.65
	20	0.46	0.88	1.34	1.88	2.29	2.7
	100	0.67	1.16	1.73	2.44	2.95	3.76
	300	2.05	3.1	4.5	6.36	8.23	10.95

from the context Γ ; these formulas must meet the conditions of a Kaykobad pair. One advantage of the Kaykobad condition is that it allows us to incrementally identify suitable constraints to add to the context Γ' , i.e., we don't have to select n constraints before we run the verification. That is, when we have identified $k < n$ constraints, we can easily verify whether a candidate $(k+1)$ -th formula would be acceptable for constructing a Kaykobad pair. Algorithm 1 computes Y -antecedents and consequents for linear inequality constraints based on Corollary III.2. Lines 6–30 search the context Γ for n constraints meeting the Kaykobad conditions. The rest of the algorithm computes the Y -antecedents/consequents. If there are n variables to be eliminated, and $N = |\Gamma|$ constraints in the context Γ , the algorithm has complexity $O(n^2 N + N^3)$. The function $\text{COEFF}(\gamma, y_j)$ extracts the coefficient of the variable y_j from the term γ . The call $\text{MATRIXFROMTERMS}(\text{MatrixRowTerms}, y)$ extracts all coefficients of the y variables contained in MatrixRowTerms and makes these coefficients the rows of the resulting matrix. The call $\text{VECTORFROMTERMS}(\text{MatrixRowTerms}, y)$ returns a vector of all expressions contained in MatrixRowTerms with their y variables removed. These are the elements of $b(x, z)$. Finally, $\text{DIAG}(v)$ returns a diagonal matrix whose entries are the vector v .

We implemented Algorithm 1 in Python and generated benchmarks for it. We considered formulas ϕ of the form (2) and contexts of the form (3). We varied the number N of constraints in the context, the number m of variables to be eliminated, and the total number of variables present in the problem $n + m + o$ (in our experiments ϕ and Γ had the same variables, so we had $o = 0$). We randomly generated coefficients for all variables in ϕ and Γ , which means that Γ was always a dense matrix, a scenario unlikely to occur in applications. We executed the algorithm for the situations in which we wanted to eliminate 2 irrelevant variables and 4 irrelevant variables. The results of the experiments are shown in Table I.

3) *Computing J via linear programming:* Now we will build J by numerically solving (4) and (5) for fixed values

Algorithm 1 Antecedents and consequents for linear inequality constraints by identifying systems of equations with positive solutions

Input: Term to transform $p^\top x + q^\top y \leq r$, context Γ ,
transform instruction s (**true** for antecedents and **false** for consequents)

Output: Transformed term t' lacking any y variables

```

1: MatrixRowTerms  $\leftarrow \emptyset$  ▷ Rows of the context matrix  $A$ 
2: PartialSums  $\leftarrow \text{ZEROS}(\text{LENGTH}(y))$ 
3: TCoeff  $\leftarrow -1$ 
4: if  $s$  then
5:   TCoeff  $\leftarrow 1$ 
6: for  $i = 1$  to  $i = \text{LENGTH}(y)$  do ▷ One iteration per row of context matrix
7:   IthRowFound  $\leftarrow \text{false}$  ▷ Indicate whether we could add the  $i$ -th row
8:   for  $\gamma \in \Gamma \setminus \text{MatrixRowTerms}$  do
9:     ▷ 1. Verifying Kaykobad pair: sign of nonzero matrix terms
10:    TermIsInvalid  $\leftarrow \text{false}$ 
11:    for  $j = 1$  to  $j = \text{LENGTH}(y)$  do
12:      if  $\text{COEFF}(\gamma, y_j) \neq 0$  and  $\text{SIGN}(\text{COEFF}(\gamma, y_j)) \neq \text{SIGN}(q_j) \cdot \text{TCoeff}$  then
13:        TermIsInvalid  $\leftarrow \text{true}$ 
14:        break
15:      ▷ 2. Verifying Kaykobad pair: matrix diagonal terms
16:      if  $\text{COEFF}(\gamma, y_i) = 0$  or TermIsInvalid then
17:        next
18:      ▷ 3. Verifying Kaykobad pair: relationship between matrix and vector entries
19:      Residuals  $\leftarrow \text{zeros}(\text{LENGTH}(y))$ 
20:      for  $j = 1$  to  $j = \text{LENGTH}(y)$  do
21:        if  $j \neq i$  then
22:          Residuals[ $j$ ]  $\leftarrow \text{SIGN}(q_j) \cdot \text{TCoeff} \cdot \text{COEFF}(\gamma, y_j) \cdot \frac{q_i}{\text{COEFF}(\gamma, y_i)}$ 
23:        if  $|q_j| \cdot \text{TCoeff} \leq \text{PartialSums}[j] + \text{Residuals}[j]$  then
24:          TermIsInvalid  $\leftarrow \text{true}$ 
25:          break
26:        if not TermIsInvalid then
27:          ▷ Resulting matrix is meeting Kaykobad pair conditions at  $i$ -th row
28:          IthRowFound  $\leftarrow \text{true}$ 
29:          for  $j = 1$  to  $j = \text{LENGTH}(y)$  do
30:            PartialSums[ $j$ ]  $\leftarrow \text{PartialSums}[j] + \text{Residuals}[j]$ 
31:          MatrixRowTerms.append( $\gamma$ )
32:          break
33:   if not IthRowFound then
34:     return Error: Cannot transform term
35:  $B \leftarrow \text{MATRIXFROMTERMS}(\text{MatrixRowTerms}, y)$ 
36:  $b \leftarrow \text{VECTORFROMTERMS}(\text{MatrixRowTerms}, y)$ 
37: return  $p^\top x + q^\top B^{-1}b \leq r$ 

```

of x and z .

Lemma III.3. Let $a \in \mathbb{R}^m$ and $c \in \mathbb{R}^o$.

- Suppose $g^-(a, c)$ is finite and the optimum of the LP (4) (with $x = a$ and $z = c$) is attained at y^* . Let $J = \left\{ i \mid b_i(a, c) - \sum_{j=1}^n \beta_{ij} y_j^* = 0 \right\}$ and assume that $|J| = n$, where n is the number of optimization variables y in g^- . If B_J is invertible, then (B_J, q) is a refining pair.
- Similarly, suppose $g^+(a, c)$ is finite and the optimum of the LP (5) (with $x = a$ and $z = c$) is attained at y^* . Let $J = \left\{ i \mid b_i(a, c) - \sum_{j=1}^n \beta_{ij} y_j^* = 0 \right\}$ and assume that

$|J| = n$. If B_J is invertible, then (B_J, q) is a relaxing pair.

Proof. We prove the first part. Consider the following problems:

$$\begin{aligned} \text{primal} & \begin{cases} \underset{y}{\text{minimize}} & -q^\top y \\ \text{subject to} & By \leq b(a, c) \end{cases} \\ \text{dual} & \begin{cases} \underset{\lambda}{\text{maximize}} & -b(a, c)^\top \lambda \\ \text{subject to} & B^\top \lambda - q = 0 \\ & \lambda \geq 0 \end{cases} \end{aligned}$$

Since $g^-(a, c)$ is finite, the primal is feasible. By strong duality, so is the dual. Let λ^* be the value of λ where the dual attains its optimum. Then $\lambda^* \geq 0$ and $0 = B^\top \lambda^* - q = B_J^\top \lambda_J^* + B_{\hat{J}}^\top \lambda_{\hat{J}}^* - q$, where $\hat{J} = \{1, \dots, N\} \setminus J$. Due to complementary slackness, we know that $\lambda_{\hat{J}}^* = 0$. Thus, $0 = B_J^\top \lambda_J^* - q$. By assumption, B_J is invertible. Then (B_J, q) is a refining pair. The proof of the second part is similar. \square

Algorithm 2 Antecedents and consequents for linear inequality constraints through linear programming

Input: Term to transform $p^\top x + q^\top y \leq r$, context Γ , $a \in \mathbb{R}^m$, $c \in \mathbb{R}^o$, transform instruction s (**true** for antecedents and **false** for consequents)

Output: Transformed term t' lacking any y variables

```

1:  $B \leftarrow \text{MATRIXFROMTERMS}(\Gamma, y)$ 
2:  $b \leftarrow \text{VECTORFROMTERMS}(\Gamma, y)$ 
3:  $b_e \leftarrow \text{EVALUATE}(b, a, c)$ 
4: if  $s$  then
5:    $(\text{success}, y^*) \leftarrow \text{LINEARPROGRAMMING}(-q, B, b_e)$ 
6: else
7:    $(\text{success}, y^*) \leftarrow \text{LINEARPROGRAMMING}(q, B, b_e)$ 
8: if not success then
9:   return Error: LP is unfeasible
10:  $S \leftarrow b_e - B y^*$ 
11:  $J \leftarrow \emptyset$ 
12: for  $j = 1$  to  $j = \text{LENGTH}(b)$  do
13:   if  $S_j = 0$  then
14:      $J \leftarrow J \cup \{j\}$ 
15:  $(\text{success}, \hat{B}_J) \leftarrow \text{MATRIXINV}(B_J)$ 
16: if not success then
17:   return Error: cannot invert  $B_J$ 
18: return  $p^\top x + q^\top \hat{B}_J b_j \leq r$ 

```

Lemma 2 allows us to obtain the solution to a linear programming problem with symbolic constraints $By \leq b(x, z)$ in a reduced context $B_J y \leq b_J(x, z)$, where we identify J by solving a numerical LP. Lemma 2 and Corollary III.2 yield a method for computing Y -antecedents and consequents. This method is reflected in Algorithm 2. As before, $\text{MATRIXFROMTERMS}(\Gamma, y)$ and $\text{VECTORFROMTERMS}(\Gamma, y)$ extract from the context Γ the matrix B and symbolic vector $b(x, z)$ of the constraints $By \leq b(x, z)$. $\text{EVALUATE}(b, a, c)$ returns the vector $b(a, c) \in \mathbb{R}^N$. $\text{LINEARPROGRAMMING}(q, B, b_e)$ solves the LP $\min_y q^\top y$ subject to $By \leq b_e$ and returns a success variable and the value y^* where the minimum is attained. The success variable is true when the LP is feasible and has a finite solution. MATRIXINV computes matrix inverses. Its success variable is false when the matrix is not invertible.

IV. DISCUSSION AND CONCLUDING REMARKS

To the best of our knowledge, the identification of the problems of variable elimination via antecedent and consequent synthesis as relevant to the computation of specifications for

requirement engineering is new. We provided two efficient algorithms for the solutions of these problems. Both algorithms are sound but incomplete.

The synthesis problems we considered are closely related to quantifier elimination, of which there is a large body of work. In fact, the universal solutions to the synthesis problems are expressed as quantifications—see Proposition II.1.

Quantifier-elimination algorithms for real arithmetic are given by Ferrante and Rackoff [6], Monniaux [7], Nipkow [8], John and Chakraborty [9], and others. Audemard et al. [10] and Bjørner [11] discuss linear quantifier elimination methods in the context of DPLL-based search. Cousot and Halbwachs [12] and Monniaux [13] apply it in the context of abstract interpretation. The earliest means for carrying out quantifier elimination for linear inequalities is Fourier-Dines-Motzkin elimination [14], [15], [16], to which many improvements have been made—see [17], [18], [19], [20], [21]. All known algorithms have at least exponential worst case complexity, but can be extremely performant on many typical problems—see [7] for details.

An important use of the computation of specifications in requirement engineering is the support of design-space exploration and tradeoff analysis. When designing complex systems, engineers may need to navigate a large design space. To do this effectively, the computation of specifications has to be supported by efficient algorithms. This motivated us to look for efficient, though incomplete, algorithms for antecedent/consequent synthesis. Moreover, in requirement engineering, we want to produce outputs which are syntactically similar to the formulas from which variables are eliminated—see the discussion in Section II. This aspect motivated our results of Propositions II.2 and II.3.

Some next steps we perceive in contextual variable elimination for requirement engineering include better algorithms for polyhedral constraint synthesis and support for temporal logic. The algorithms we proposed to synthesize Y -antecedents and consequents are based on Corollary III.1, which yields optimal solutions. However, the algorithms we presented for solving (4) and (5) have room to improve. Per Lemma III.1, given a set of N linear equations in n variables ($N > n$), it would be very useful to research methods to efficiently identify a set of n linear equations with a nonnegative solution. Finally, to support the computation of Y -antecedents and consequents of formulas in a context for temporal logic specifications, we believe the decomposition enabled by Proposition II.5 could have a useful role.

REFERENCES

- [1] A. Benveniste, B. Caillaud, D. Nickovic, R. Passerone, J.-B. Ralet, P. Reinkemeier, A. Sangiovanni-Vincentelli, W. Damm, T. A. Henzinger, and K. G. Larsen, “Contracts for system design,” *Foundations and Trends® in Electronic Design Automation*, vol. 12, no. 2-3, pp. 124–400, 2018.
- [2] I. Incer, *The Algebra of Contracts*. PhD thesis, EECS Department, University of California, Berkeley, May 2022.
- [3] A. L. Sangiovanni-Vincentelli, W. Damm, and R. Passerone, “Taming Dr. Frankenstein: Contract-based design for cyber-physical systems,” *Eur. J. Control*, vol. 18, no. 3, pp. 217–238, 2012.

- [4] S. Boyd and L. Vandenberghe, *Convex Optimization*. Cambridge university press, 2004.
- [5] M. Kaykobad, "Positive solutions of positive linear systems," *Linear Algebra and its Applications*, vol. 64, pp. 133–140, 1985.
- [6] J. Ferrante and C. Rackoff, "A decision procedure for the first order theory of real addition with order," *SIAM Journal on Computing*, vol. 4, no. 1, pp. 69–76, 1975.
- [7] D. Monniaux, "A quantifier elimination algorithm for linear real arithmetic," in *Logic for Programming, Artificial Intelligence, and Reasoning: 15th International Conference, LPAR 2008, Doha, Qatar, November 22-27, 2008. Proceedings 15*, pp. 243–257, Springer, 2008.
- [8] T. Nipkow, "Linear quantifier elimination," in *Automated Reasoning* (A. Armando, P. Baumgartner, and G. Dowek, eds.), (Berlin, Heidelberg), pp. 18–33, Springer Berlin Heidelberg, 2008.
- [9] A. K. John and S. Chakraborty, "A layered algorithm for quantifier elimination from linear modular constraints," *Formal Methods in System Design*, vol. 49, pp. 272–323, 2016.
- [10] G. Audemard, P. Bertoli, A. Cimatti, A. Kornilowicz, and R. Sebastiani, "Integrating boolean and mathematical solving: Foundations, basic algorithms, and requirements," in *Artificial Intelligence, Automated Reasoning, and Symbolic Computation* (J. Calmet, B. Benhamou, O. Caprotti, L. Henocque, and V. Sorge, eds.), (Berlin, Heidelberg), pp. 231–245, Springer Berlin Heidelberg, 2002.
- [11] N. Bjørner, "Linear quantifier elimination as an abstract decision procedure," in *Automated Reasoning* (J. Giesl and R. Hähnle, eds.), (Berlin, Heidelberg), pp. 316–330, Springer Berlin Heidelberg, 2010.
- [12] P. Cousot and N. Halbwachs, "Automatic discovery of linear restraints among variables of a program," in *Proceedings of the 5th ACM SIGACT-SIGPLAN Symposium on Principles of Programming Languages*, POPL '78, (New York, NY, USA), p. 84–96, Association for Computing Machinery, 1978.
- [13] D. P. Monniaux, "Automatic modular abstractions for linear constraints," *ACM SIGPLAN Notices*, vol. 44, no. 1, pp. 140–151, 2009.
- [14] J. Fourier, "Solution d'une question particulière du calcul des inégalités," *Nouveau Bulletin des sciences par la Société philomathique de Paris*, p. 99, pp. 317–319, 1826.
- [15] L. L. Dines, "Systems of linear inequalities," *Annals of Mathematics*, pp. 191–199, 1919.
- [16] T. S. Motzkin, *Beiträge zur Theorie der linearen Ungleichungen*. PhD thesis, University of Basel, 1936.
- [17] G. B. Dantzig and B. Curtis Eaves, "Fourier-motzkin elimination and its dual," *Journal of Combinatorial Theory, Series A*, vol. 14, no. 3, pp. 288–297, 1973.
- [18] R. J. Duffin, "On Fourier's analysis of linear inequality systems," in *Pivoting and Extension: In honor of A.W. Tucker* (M. L. Balinski, ed.), pp. 71–95, Berlin, Heidelberg: Springer Berlin Heidelberg, 1974.
- [19] J.-L. Lassez and M. J. Maher, "On Fourier's algorithm for linear arithmetic constraints," *Journal of Automated Reasoning*, vol. 9, pp. 373–379, 1992.
- [20] V. Chandru, "Variable elimination in linear constraints," *The Computer Journal*, vol. 36, no. 5, pp. 463–472, 1993.
- [21] J. Imbert, "Fourier's elimination: Which to choose?," in *Principles and Practice of Constraint Programming*, pp. 117–129, 1993.