A Generic Projection Operator
for Partial Cylindrical Algebraic Decomposition

Andreas Seidl[*]
FMI, University of Passau
Germany
seidl@fmi.uni-passau.de

Thomas Sturm[†]
FMI, University of Passau
Germany
sturm@uni-passau.de

ABSTRACT

This paper provides a starting point for generic quantifier elimination by partial cylindrical algebraic decomposition (PCAD). On input of a first-order formula over the reals generic PCAD outputs a theory and a quantifier-free formula. The theory is a set of negated equations in the free variables of the input formula. The quantifier-free formula is equivalent to the input for all parameter values satisfying the theory. For obtaining this generic elimination procedure, we derive a generic projection operator from the standard Collins–Hong projection operator. Our operator particularly addresses formulas with many parameters thus filling a gap in the applicability of PCAD. It restricts decomposition to a reasonable subset of the entire space. The above-mentioned theory describes this subset. The approach is compatible with other improvements in the framework of PCAD. It turns out that the theory contains assumptions that are easily interpretable and that are most often non-degeneracy conditions. The applicability of our generic elimination procedure significantly extends that of the corresponding regular procedure. Our procedure is implemented in the computer logic system REDLOG.

Categories and Subject Descriptors

G.4. [**Mathematics of Computing**]: Mathematical Software—*Algorithm design and analysis*

General Terms

Algorithms,Performance,Experimentation,Theory

Keywords

Partial CAD,Projection Operator,REDLOG

[*]http://www.fmi.uni-passau.de/~seidl/

[†]http://www.fmi.uni-passau.de/~sturm/

1. INTRODUCTION

It is generally agreed that real quantifier elimination is an important and powerful tool in symbolic computation. The most elaborate implemented real quantifier elimination approach is that of partial cylindrical algebraic decomposition (PCAD), on which research is going on for 30 years now. Another important implemented approach is *virtual substitution methods* [22], which have been thoroughly investigated by Weispfenning and his group during the last 15 years. The advantages and disadvantages of these different approaches can be summarized as follows:

Virtual substitution is restricted to input formulas that obey certain restrictions on the maximal degree of the quantified variables. They can be in theory generalized to arbitrary degrees [23], but this does not appear to be feasible in practice. Much effort has been spent in developing and efficiently implementing heuristic approaches to relax the degree restrictions with the virtual substitution approach—polynomial factorization is the most obvious idea. Though this has been surprisingly successful, the principal problem remains. The computed quantifier-free equivalents are often quite redundant and hard to interpret, although there has been considerable progress with the introduction of sophisticated simplification techniques [8]. Virtual substitution is doubly exponential in the number of quantifier changes, but for fixed quantifier type only singly exponential in the number of quantifiers; most importantly, the number of parameters, i.e. free variables, does not contribute to complexity in a relevant way.

PCAD is a complete real quantifier elimination method. That is, there are no restrictions on the possible input formulas. The computed quantifier-free equivalents are simple to understand, i.e., there are not many algebraic or logical redundancies. On the negative side, PCAD has a higher theoretical complexity than virtual substitution methods: It is doubly exponential in *all* variables. This makes it particularly inferior to virtual substitution methods for input with many parameters.

From this point of view, there is a significant gap in the applicability of real quantifier elimination, which affects input problems where there are on one hand quantified variables of comparatively high degree and on the other hand comparatively many parameters. More precisely, it can be sufficient for virtual substitution to fail that the total degree in the quantified variables exceeds 1, and on the other hand there is a famous example by Hong [14] with one existential quantifier and three parameters where current implementations of PCAD use to fail.

In [9] *generic quantifier* elimination has been introduced on the basis of real quantifier elimination by virtual substitution. This works as follows: On input of a first-order formula φ over the reals, the generic quantifier elimination procedure has two return values:

1. A *theory* Θ, i.e., a list of nontrivial negated equations, also called *assumptions*, in the parameters of φ.

2. A quantifier-free formula φ' in the parameters of φ.

The specification of the algorithm is that φ' is a quantifier-free equivalent to φ for all choices of parameters satisfying Θ; formally

$$\mathbb{R} \models \bigwedge \Theta \longrightarrow (\varphi' \longleftrightarrow \varphi).$$

Note that there are never equalities, trivial negated equalities, or ordering inequalities assumed, and that there are no Boolean connections other than conjunction possible between assumptions. As a consequence, the exception set, for which φ' is *not* correct, has measure zero within the parameter space. The idea behind generic quantifier elimination is that the assumption of Θ supports the construction of φ' to such an extent that the range of practically feasible problems is significantly extended.

With virtual substitution methods this has without doubt been the case. This has been demonstrated in particular in the area of automated geometry proving [9, 20] and physical network analysis [21]. In all applications examined so far there have been two more most interesting observations made:

• The assumptions Θ use to have a straightforward interpretation within the real system modeled by the input. For instance, conditions on an electric circuit would not compare voltages with resistances.

• Even more strikingly, the assumptions often provide additional *non-degeneracy* assumptions that are actually necessary to make the input a sufficiently precise model of the real world. For instance, conditions on a geometric configuration would state that a triangle does not degenerate to a line.

We shall see a concrete examples for both these points in Section 5.

Regarding the second point, in his famous monograph on geometry proving [4], Chou has convincingly demonstrated that for geometric configurations it is not practicable to determine all necessary non-degeneracy conditions in advance.

Consequently, straightforward algebraic formulations of geometric theorems are in most cases "false," and this would in fact be the result of any regular quantifier elimination procedure applied to their universal closure. When leaving the free variables in the sense of Chou unquantified, generic quantifier elimination, in contrast, adds in almost all cases considered so far the missing input specifications to Θ and obtains "true" as φ'. From this point of view, generic quantifier elimination is much more than a weaker form of regular quantifier elimination. It is noteworthy that the Wu–Ritt reduction techniques used by Chou yield conditions in the parameters very similar to our assumptions.

This paper provides a starting point for generic quantifier elimination by PCAD:

1. We define a *generic projection operator* GPROJ, which is derived from the standard Collins–Hong projection operator PROJH [5, 12]. This operator is compatible with the common variants of cell decomposition and solution formula construction. On the other hand, it even allows for a specifically optimized decomposition.

2. Our operator GPROJ particularly addresses formulas with many parameters, and thus fills in the gap mentioned above: It will turn out that projection within parameter space systematically allows for assumptions. In bound variable space, in contrast, this requires certain configurations, which occur in practice with significant frequency but not systematically.

3. Unlike all other improvements of projection operators discussed in the literature so far, we do not only aim at a coarser decomposition of the entire space but at decomposing only a reasonable subset of this space. Our approach is compatible with other improvements in the framework of PCAD.

4. Our projection operator GPROJ allows to restrict the possible form of valid assumptions: One can, e.g., restrict to monomial assumptions.

5. We have implemented quantifier elimination by PCAD using our generic projection operator GPROJ. This allows us to judge the empirical performance of our approach on practical examples.

6. Furthermore, it is possible with our implementation to choose between generic quantifier elimination using our GPROJ and regular quantifier elimination using PROJH. This ensures perfect comparability of computation times as well as of qualities of results between our approach and the regular one. It turns out that generic PCAD dramatically exceeds the capabilities of regular PCAD.

7. As discussed with other methods above, we obtain assumptions Θ that are easily interpretable and that are most often non-degeneracy conditions.

For regular PCAD it is best practice to first try McCallum's projection including Brown's improvement. If then decomposition fails, one would fall back on Collins–Hong projection. Our observations apply to the McCallum–Brown projection as well. Consequently our approach can be extended to adopt exactly the above-mentioned practice. We expect our approach to be beneficial with the McCallum–Brown as well, but further research and implementation will be necessary to demonstrate this.

It is not hard to see that the introduction of our generic projection operator does not lead in general to any better upper worst-case complexity bounds compared to regular PCAD. We thus focus on demonstrating its applicability by solving with our implementation of generic PCAD examples that are not solvable with its non-generic counterpart. Since real quantifier elimination has during the past years been on the edge between academic examples and real world problems, any step further is most promising.

In contrast to related work on strict polynomial inequalities [19, 16], our approach does not require the input to be of any particular syntactic form. Instead, we single out certain

factors during projection when we have already abstracted from the syntactic structure of the input formula.

Our implementations are part of the current development version of the computer logic system REDLOG, which in turn forms an integral part of the REDUCE distribution. So the software described here will be published with the next release REDUCE 3.8.[1] The framework of REDLOG offers the crucial advantage that the work described here can be easily combined with a huge variety of already present services and tools on first-order formulas. This affects the user interface level as well as the implementation level.

The plan of this paper is a follows: Section 2 is the technical core of the paper. Here we define our generic projection operator, which is derived from Hong's modification in [12] of Collins' original projection operator in [5], and prove its correctness. In Section 3 we address the consequences for decomposition and solution formula construction when using our generic projection operator for performing generic quantifier elimination. Section 4 gives a description of the current status of the implementation of PCAD within the computer logic system REDLOG [7, 18]. In Section 5 we give computation examples with this implementation. In Section 6 we finally summarize and evaluate our work.

2. GENERIC PROJECTION OPERATOR

Throughout this section, we consider our prenex input formula to contain variables $x_1, \ldots, x_r$, where $x_1, \ldots, x_k$ are free variables, which we also refer to as *parameters*, and $x_{k+1}, \ldots, x_r$ are variables bound by quantifiers. By I_j we denote $\mathbb{Z}[x_1, \ldots, x_j]$. Consequently, I_r is the set of all polynomials possibly occurring in our formula, and I_k is the set of all polynomials containing only parameters. Let A generally denote a finite subset of I_r.

We recall the definition of Hong's projection operator from [12]:

$$
\begin{aligned}
\text{PROJH}(A) &= \text{PROJ}_1(A) \cup \text{PROJ}_2^*(A) \\
\text{PROJ}_1(A) &= \bigcup_{\substack{f \in A \\ f^* \in \text{RED}(f)}} \left(\{\text{ldcf}(f^*)\} \cup \text{PSC}(f^*, f^{*\prime}) \right) \\
\text{PROJ}_2^*(A) &= \bigcup_{\substack{f, g \in A \\ f < g}} \bigcup_{f^* \in \text{RED}(f)} \text{PSC}(f^*, g).
\end{aligned}
$$

For the motivation of our approach, consider the leading coefficients added in $\text{PROJ}_1(A)$. In contrast to only adding the leading coefficient of each polynomial in A, there are in addition the leading coefficients of all reducta added. The reason for this is that there will in general be choices for variables such that these leading coefficients vanish. For the relevant properties of the projection sets it is, however, crucial to include leading coefficients for all possible choices of variables including degenerate situations.

From this point of view, the construction of the chain of reducta can be stopped as soon as the first constant leading coefficient appears; a fact, which is well-known in the community. Our idea is now to go one step further: We are going to stop this process as soon as a leading coefficient appears, which contains only parameters. We simply assume this parametric leading coefficient to be nonzero. This assumption, which is formally a negated equation, is added to a *theory* Θ.

Similar observations hold for the chains of principal subresultant coefficients computed in both PROJ_1 and PROJ_2^*.

After our generic projection we continue PCAD. At the end we have obtained on input of a first order formula φ both a quantifier-free formula φ' and the theory Θ mentioned above. The result φ' is correct for all choices of parameters satisfying Θ. Formally, we have

$$
\mathbb{R} \models \bigwedge \Theta \longrightarrow (\varphi' \longleftrightarrow \varphi).
$$

From this semantical description it is straightforward that we do not want to admit any assumptions on bound variables. Recall that projection proceeds from bound variable space to parameter space. Within the former it is good luck to find parametric leading coefficients or principal subresultant coefficients; within the latter they occur systematically.

We are now going to formalize our idea and prove its correctness.

For the definition of our generic projection operator, we define a generic set of reducta as follows: If there is some j such that $\text{red}^j(f) \neq 0$ and $\text{ldcf}(\text{red}^j(f)) \in I_k$ and there is some $j < j'$ such that $\text{red}^{j'}(f) \neq 0$, then

$$
\mu = \min\{\, j \mid \text{red}^j(f) \neq 0 \text{ and } \text{ldcf}(\text{red}^j(f)) \in I_k \,\}
$$

and

$$
\text{GRED}(f) = \left(\{\text{ldcf}(\text{red}^\mu(f)) \neq 0\}, \{\, \text{red}^i(f) \mid 0 \leq i \leq \mu \,\} \right).
$$

Else $\text{GRED}(f) = (\emptyset, \text{RED}(f))$.

Similarly, we have a generic set of principal subresultant coefficients: If there is $j < \min\{\deg(f), \deg(g)\}$ such that $\text{psc}_j(f, g) \in I_k$ and there is $j < j' < \min\{\deg(f), \deg(g)\}$ such that $\text{PSC}_{j'}(f, g) \neq 0$, then

$$
\mu = \min\{\, j < \min\{\deg(f), \deg(g)\} \mid \text{psc}_j(f, g) \in I_k \,\}
$$

and

$$
\text{GPSC}(f, g) = \left(\{\text{psc}_\mu(f, g) \neq 0\}, \{\, \text{psc}_i(f, g) \mid 0 \leq i \leq \mu \,\} \right).
$$

Else $\text{GPSC}(f, g) = (\emptyset, \text{PSC}(f, g))$.

As a final preparational step, we make the conventions that

$$
\text{gldcf}(f) = \left(\emptyset, \{\text{ldcf}(f)\} \right)
$$

and that

$$
(\Theta, S) \sqcup (\Theta', S') = (\Theta \cup \Theta', S \cup S').
$$

This allows us to define our generic projection operator as follows:

$$
\begin{aligned}
\text{GPROJ}(A) &= \text{GPROJ}_1(A) \sqcup \text{GPROJ}_2^*(A) \\
\text{GPROJ}_1(A) &= \bigsqcup_{\substack{f \in A \\ f^* \in \text{GRED}(f)}} \left(\{\text{gldcf}(f^*)\} \sqcup \text{GPSC}(f^*, f^{*\prime}) \right) \\
\text{GPROJ}_2^*(A) &= \bigsqcup_{\substack{f, g \in A \\ f < g}} \bigsqcup_{f^* \in \text{GRED}(f)} \text{GPSC}(f^*, g).
\end{aligned}
$$

Note that all polynomials occurring as the left hand sides of assumptions are as well part of the projection set. This way, after computation of a sign-invariant decomposition, every assumption will be either constantly valid or constantly invalid over each cell.

We recall further definitions. We denote by $V(A)$ the real variety of A. For $0 \leq j < r$ let S be a connected subset of $\mathbb{R}^j$. Then $Z(S) = S \times \mathbb{R}$ is the *cylinder* over S.

We adopt the definition of a *section* of a cylinder from [1]. Then A is *delineable* on S if the portion of $V(A)$ lying in $Z(S)$ consists of n disjoint sections of $Z(S)$ for some $n \geq 0$. This notion of delineability, which is a bit weaker than the original one by Collins, has been introduced in [1]. We shall allow ourselves in the sequel to briefly say *invariant* instead of sign-invariant.

For a set of conditions Θ in variables $x_1, \ldots, x_r$ and $0 \leq j \leq r$ let

$$\mathbb{R}^j_\Theta = \left\{ \underline{x} \in \mathbb{R}^j \;\middle|\; \mathbb{R} \models \bigwedge \Theta(\underline{x}, \underline{y}) \text{ for some } \underline{y} \in \mathbb{R}^{r-j} \right\}.$$

LEMMA 1. *Let A be a finite subset of I_j for $j \geq 2$, and say $\mathrm{GPROJ}_1(A) = (\Theta, P)$. Let S be a connected subset of $\mathbb{R}^{j-1}_\Theta$ such that every element of P is invariant on S. Then every element of A is either delineable or identically zero on S.*

PROOF. Our assertion is a modification of Lemma 2 in [12], for which the proof is given in Theorem 4 of [5]. We modify this proof to derive a proof of our claim as follows: Fix an element $f(x_1, \ldots, x_j) = \sum_{i=0}^m f_i x_j^i$ of A and a connected subset S of $\mathbb{R}^{j-1}_\Theta$ such that f is not identically zero on S.

Choose $m \geq 1$ maximal such that $f_m \neq 0$ on S, and choose k such that

$$g := \mathrm{red}^k(f) = \sum_{i=0}^m f_i x_j^i.$$

Denote $\mathrm{GRED}(f) = (\Theta_1, B)$, and assume $\Theta_1 \neq \emptyset$, say

$$\Theta_1 = \{\mathrm{ldcf}(\mathrm{red}^\mu(f)) \neq 0\}, \quad B = \{f, \mathrm{red}(f), \ldots, \mathrm{red}^\mu(f)\}.$$

Because of $\Theta_1 \subseteq \Theta$ and our assumption on S we have $k \leq \mu$. Thus certainly $g \in B$ and $\mathrm{ldcf}(g) \in P$.

Choose $l \geq 1$ minimal such that $\mathrm{psc}_l(g, g') \neq 0$ on S. Denote $\mathrm{GPSC}(g, g') = (\Theta_2, S_1)$. Assume $\Theta_2 \neq \emptyset$, say

$$\Theta_2 = \{\mathrm{psc}_\nu(g, g') \neq 0\}, \quad S_1 = \{\mathrm{psc}_0(g, g'), \ldots, \mathrm{psc}_\nu(g, g')\}.$$

In the same way as above we conclude that $l \leq \nu$, hence $psc_l(g, g') \in S_1$.

We have shown that whenever we make assumptions and cease to include further polynomials, then all polynomials needed for the original proof that we are modifying are contained in our generic projection set. $\square$

LEMMA 2. *Let A be a finite subset of I_j for $j \geq 2$, and say $\mathrm{GPROJ}_1(A) = (\Theta, P)$. Let S be a connected subset of $\mathbb{R}^{j-1}_\Theta$ such that every element of P is invariant on S. Let f and g be any two different polynomials in A. If the least integer k such that*

$$\mathrm{psc}_k\big(f(\underline{\alpha}, x_j), g(\underline{\alpha}, x_j)\big) \neq 0$$

does not vary for $\underline{\alpha} \in S$, then the sections of $Z(S)$ belonging to f and g are either disjoint or identical. $\square$

Our Lemma 2 is a modification of Lemma 3 in [12]. The basic proof ideas for the latter are given in Theorem 5 of [5]. Using similar modifications as in the proof of Lemma 1 one obtains a proof also for our Lemma 2.

THEOREM 3 (GPROJ IS A PROJECTION OPERATOR). *Let A be a finite subset of I_j for $j \geq 2$, and say $\mathrm{GPROJ}_1(A) = (\Theta, P)$. Let S be a connected subset of $\mathbb{R}^{j-1}_\Theta$ such that every element of P is invariant on S. Then the following two conditions hold:*

1. *Every element of A is either delineable or identically zero on S.*

2. *The sections of $Z(S)$ belonging to different $f, g \in A$ are either disjoint or identical.* $\square$

Again, a proof can be obtained by adaption of the proof of Theorem 1 in [12].

It is not hard to see that our generic projection remains correct if we impose restrictions on the form of possible assumptions. From an application point of view this is a most interesting feature. The user might, e.g., wish to obtain only monomial assumptions, require at least one variable to occur only linearly, impose other degree restrictions, or wish to prohibit assumptions on certain parameters.

By successively liberating restrictions on possible assumptions, our generic projection set scales from the Collins–Hong projection set towards Brown's projection set [3]. In fact, Brown's projection can be regarded as implicitly making assumptions even on bound variables without explicitly creating any theory. Any mistakes resulting from this are, however, detected during extension. From our point of view, Brown pays the price that his decomposition, in contrast to ours, can fail.

3. DECOMPOSITION AND QUANTIFIER ELIMINATION

Starting with a prenex first-order formula φ, we extract the polynomials obtaining a set A. Then we repeatedly apply our generic projection operator until only one variable is left. During this process, the projection set grows monotonically. All returned theories are united. The final result of this projection phase is a pair (Θ, P) consisting of a theory and a projection set wrt. this theory.

We now turn to the necessary modifications of the decomposition of r-space following the projection. For this we use the basic decomposition algorithm of Collins [5], which is known to be compatible with Collins–Hong projection. We are going to comment on the compatibility of our modification with PCAD [6] at the end of this section. The crucial feature of any such decomposition is that all polynomials from the projection set P are invariant on all cells of the decomposition.

Decomposition as known in the literature works as follows: As a starting point, the trivial decomposition D_0 of $\mathbb{R}^0$ is used. Let $0 < j \leq r$, and let D_{j-1} be a decomposition already computed for $\mathbb{R}^{j-1}$. For each *base cell* $B \in D_{j-1}$ there is a decomposition of $B \times \mathbb{R}$ generated. This is called the *stack* over B. Then the decomposition D_j of $\mathbb{R}^j$ is obtained as the union of all stacks.

For our needs we modify this algorithm as follows: Each cell of the stack over a base cell B lies either fully within $\mathbb{R}^j_\Theta$ or fully within its complement. This follows from our observation right after the definition of GPROJ in Section 2 that all left hand side polynomials of the assumptions are contained in the generic projection set. We drop all cells that do not lie within $\mathbb{R}^j_\Theta$. This has the same effect as applying the common trial evaluation to $\varphi \wedge \bigwedge \Theta$ instead of φ, where φ is the matrix of the prenex input formula.

Our modified algorithm applied to a finite set $P \subseteq I_r$ results in a cylindrical decomposition of $\mathbb{R}^r_\Theta$ that is invariant wrt. P. It is not too hard to derive from Theorem 3 the following result, which states this observation more formally:

LEMMA 4. *Let (Θ, P) be the generic projection result of $A \subseteq I_r$, let D_r be the decomposition of $\mathbb{R}^r_\Theta$ as computed by our modified decomposition algorithm, and let $D_{r-1}, \ldots, D_0$ be the induced decompositions of $\mathbb{R}^{r-1}_\Theta, \ldots, \mathbb{R}^0_\Theta$. For each $0 \le j \le r$, let P_j denote the polynomials of P that contain at most the variables $x_1, \ldots, x_j$. Then for each $0 \le j \le r$, we have that each polynomial of P_j is invariant on each cell $C \in D_j$.* □

Our next task is to derive from a cylindrical decomposition a quantifier-free formula φ' that is equivalent to our prenex input formula

$$\varphi(x_1, \ldots, x_k) = \mathsf{Q}_{k+1}x_{k+1} \ldots \mathsf{Q}_r x_r \psi$$

with $0 \le k \le r$, $\mathsf{Q}_{k+1}, \ldots, \mathsf{Q}_r \in \{\forall, \exists\}$, and ψ quantifier-free. Let A denote the polynomials of φ. We fix these naming conventions for the remainder of this section.

LEMMA 5. *Let (Θ, P) denote the generic projection result of A. Let D_r be an invariant CAD of $\mathbb{R}^r_\Theta$ wrt. A. Then for each j with $k \le j \le r$ the truth value of*

$$\varphi_j(x_1, \ldots, x_j) := \mathsf{Q}_{j+1}x_{j+1} \ldots \mathsf{Q}_r x_r \psi$$

is constant throughout all cells of D_j, where D_j denotes the decomposition induced by D_r of $\mathbb{R}^j_\Theta$.

PROOF. Induction on j. For $j = r$ we have $\varphi_r = \psi$. According to Lemma 4 all polynomials of ψ are invariant on all cells of D_r. The truth value of ψ is invariant as well. Now assume that $k \le j < r$ and that the assumption holds for $j + 1$. Let C be a cell in D_j. All cells of the stack above C have constant truth value on each of these cells by assumption. According to whether Q_{j+1} is a universal or an existential quantifier, φ_j has constant truth value on C, viz. the conjunction or disjunction of the truth values of the stack, respectively. □

The above Lemma 5 guarantees for each cell $C \in D_k$ a constant truth value v_C. As each cell $C \in D_k$ is semi-algebraic, there exists a quantifier-free defining formula δ_C for this cell.

THEOREM 6. *Let D_k denote the decomposition induced by D_r of $\mathbb{R}^k_\Theta$. Then we have*

$$\mathbb{R} \models \bigwedge \Theta \longrightarrow \left(\bigvee_{\substack{C \in D_k \\ v_C = \text{true}}} \delta_C \longleftrightarrow \varphi \right). \quad \square$$

As shown in [13], it is often successful to first try Hong's method for constructing a solution formula. This method can fail, if the CAD turns out *not* to be projection-definable. In this case, by carefully adding derivatives of projection factors, the resulting recomputed CAD will eventually be projection-definable [2]. Thus our method can be considered complete.

Concentrating on essentials, we have described our generic quantifier elimination in terms of traditional, i.e. non-partial, CAD. Our approach is obviously fully compatible with PCAD as described by Collins and Hong in [6] including all improvements discussed there.

We conclude this section with an important observation: With generic quantifier elimination by virtual substitution it is straightforward to extend the approach to allow starting with an external theory instead of the empty set [9]. The

same holds for theory-based simplification techniques [8]. These external conditions are used in the same way as the theory implicitly generated by the procedures. It is most tempting but *not* correct to straightforwardly extend this idea to our framework here. The crucial assumption for our modified decomposition that the assumptions are truth invariant on each cell would be violated by this.

4. IMPLEMENTATION

Our generic PCAD is implemented within the current development version of the computer logic system REDLOG [7] based on REDUCE. REDLOG is a regular part of the REDUCE distribution. So the software described here will be published with the next release of REDUCE. The framework of REDLOG offers the crucial advantage that the work described here can be easily combined with a huge variety of already present algorithms on and utilities for first-order formulas. This affects the user interface level as well as the implementation level.

While REDLOG exists for already 10 years now, the development of PCAD within this framework is rather new. It started with the efficient implementation of algebraic numbers in 2001 [18]. We now give an overview on the current status of our PCAD implementation. This implementation forms the basis for the generic variant described in this article.

For the projection phase, we have the operators available proposed by Collins [5], Hong [12], McCallum [17], and Brown [3]. We use a modularized approach such that it is easy to make available possible variants of projection operators discussed in the literature. In addition we can easily define *combined projection operators*: Such combined projection operators apply various projection operators choosing them according to the levels of polynomials. It is possible to interactively manipulate these projection sets. The decomposition phase employs *partial decomposition* as proposed by Hong, including his improvements, which are *trial evaluation* and *propagation below free variable space* [6]. For the solution formula construction we use Hong's version described in [13].

We finally recall that REDLOG contains powerful simplifiers [8]. We use these simplifiers in two different ways: First, instead of blindly making assumptions, we conservatively check whether a desired non-zeroness follows from assumptions already made before. Second, we obtain by simplification a more concise theory for output at the end.

5. COMPUTATION EXAMPLES

All computations have been performed on a 933 MHz Intel Pentium III under Linux using 128 MB RAM. All timings are CPU times including garbage collection times.

5.1 Inverting Operation Amplifier

We revisit an example from [10, 21]. The inverting operation amplifier in Figure 1 is described by the following set of equations. This description has been automatically generated from a graph-like description of the circuit using
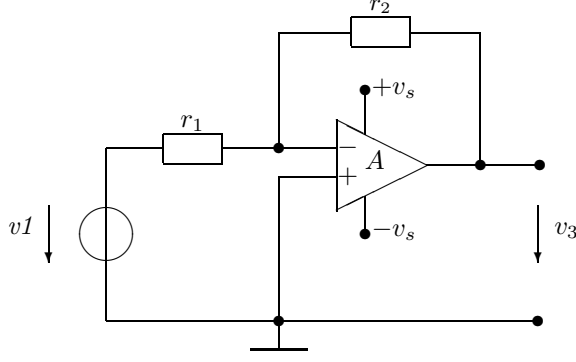
Figure 1: An inverting operation amplifier circuit

the MATHEMATICA-based system Analog Insydes [11]:

$$v_1 = v1$$
$$v_2 = -v_{\text{pm\_op1}}$$
$$v_3 = v_{\text{og\_op1}}$$
$$v_1 + i_{\text{v0}}r_1 = v_2$$
$$v_2 r_1 + v_2 r_2 = v_3 r_1 + v_1 r_2 + i_{\text{pm\_op1}} r_1 r_2$$
$$v_3 + i_{\text{og\_op1}} r_2 = v_2$$
$$v_{\text{og\_op1}} = v_{\text{pm\_op1}} x_{\text{op1}}^2$$
$$v_s^2 x_{\text{op1}}^2 + A v_{\text{og\_op1}}^2 = A v_s^2$$
$$i_{\text{pm\_op1}} = 0.$$

The aim is to determine the output voltage v_3 as a function of the input voltage $v1$. The amplification factor A, the supply voltage v_s, and the resistances r_1 and r_2 are parameters. All other variables

$$v_{\text{pm\_op1}},\ i_{\text{og\_op1}},\ i_{\text{pm\_op1}},\ i_{\text{v0}},\ v_1,\ v_2,\ v_{\text{og\_op1}},\ x_{\text{op1}}$$

have to be existentially quantified.

In [10] it had been tried to determine a solution via computation of an elimination ideal basis, where it was for principal reasons not possible to exclude certain so-called parasitic solutions. The problem has then been satisfactorily solved in [21] using generic quantifier elimination by virtual substitution. For our purposes here, we consider it an excellent benchmark example for the decomposition of a 14-dimensional space. The 14 variables partition into 6 existentially quantified variables and 8 parameters; it should be mentioned that the elimination of one of the quantified variables, viz. v_1, is in fact trivial.

The projection order for all computation variants on this example discussed in the sequel is as follows:

$$\overset{v_{\text{pm\_op1}}}{\to}\ \overset{i_{\text{og\_op1}}}{\to}\ \overset{i_{\text{pm\_op1}}}{\to}\ \overset{i_{\text{v0}}}{\to}\ \overset{v_1}{\to}\ \overset{v_2}{\to}$$

$$\overset{v_{\text{og\_op1}}}{\to}\ \overset{x_{\text{op1}}}{\to}\ \Big|\ \overset{A}{\to}\ \overset{v_s}{\to}\ \overset{v1}{\to}\ \overset{r_2}{\to}\ \overset{v_3}{\to}\ \overset{r_1}{\to}.$$

By placing "|" we indicate the beginning of the parameter space.

Using PROJH, the size of the projection set develops as follows:

$$9 \to 12 \to 14 \to 16 \to 17 \to 20 \to$$
$$30 \to 42 \to\ |\ 78 \to 375 \to \perp \qquad (> 104\ \text{min}).$$

That is, there are 9 input factors before the projection of $v_{\text{pm\_op1}}$. This first projection step results in 12 factors altogether before the projection of $i_{\text{og\_op1}}$, etc. After 104 min the computation aborts during the projection of v_s due to exceeding the chosen memory size of 128 MB.

Using GPROJ, we successively obtain projection set cardinalities as follows:

$$9 \to 12 \to 13 \to 15 \to 15 \to 16 \to$$
$$18 \to 26 \to\ |\ 32 \to 35 \to 35 \to 35 \to 35 \qquad (870\ \text{ms}).$$

During this projection, there is in addition to the projection factors the following theory of 13 negated equations generated:

$$\Big\{\ 4A^2 r_1^2 v_3^2 + 8A^2 r_1 r_2 v1 v_3 + 4A^2 r_2^2 v1^2 + r_1^2 v_s^2 +$$
$$2r_1 r_2 v_s^2 + r_2^2 v_s^2 \neq 0,$$
$$Ar_1 v_3^3 - Ar_1 v_3 v_s^2 + Ar_2 v1 v_3^2 - Ar_2 v1 v_s^2 +$$
$$r_1 v_3 v_s^2 + r_2 v_3 v_s^2 \neq 0,$$
$$Ar_1 v_3^3 - Ar_1 v_3 v_s^2 + Ar_2 v1 v_3^2 - Ar_2 v1 v_s^2 -$$
$$r_1 v_3 v_s^2 - r_2 v_3 v_s^2 \neq 0,$$
$$Ar_1 v_3 + Ar_2 v1 + r_1 v_3 + r_2 v_3 \neq 0,$$
$$A \neq 0,$$
$$r_1 v_3 + r_2 v1 \neq 0,$$
$$r_1 + r_2 \neq 0,$$
$$r_1 \neq 0,$$
$$r_2 \neq 0,$$
$$v_3 + v_s \neq 0,$$
$$v_3 - v_s \neq 0,$$
$$v_3 \neq 0,$$
$$v_s \neq 0\ \Big\}.$$

Recall that the theories which we give here have undergone some simplification. There have been possibly more than 13 assumptions made during projection.

Among the assumptions in the theory, we find typical non-degeneracy conditions: resistances like r_1, r_2 can, of course, never be zero in reality; the output voltage v_3 can naturally never reach the supply voltage v_s.

The entire generic PCAD finishes after 12 min yielding the quantifier-free result "false." This is a bit surprising. The reason is that the algebraic equation describing the behavior of the circuit is actually the negation of the third assumption in the theory.

Instead of going into details about the physical facts that could be derived in this situation, we continue seeking for a straightforward result. For this we once more use GPROJ but now make use of the important feature to restrict the form of admissible assumptions for the theory: We admit only monomial assumptions. This way, we successively obtain the following cardinalities of projection sets:

$$9 \to 12 \to 13 \to 15 \to 15 \to 16 \to$$
$$19 \to 26 \to\ |\ 32 \to 35 \to 35 \to 35 \to 35 \qquad (230\ \text{ms}).$$

In addition, we obtain the following theory with 5 negated equations:

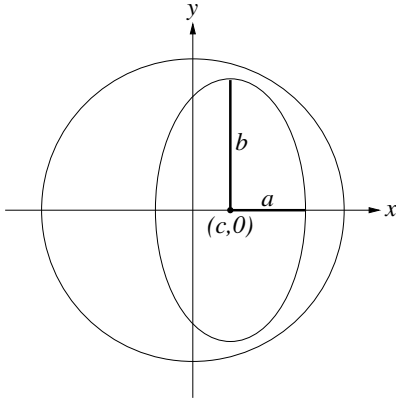$$\{A \neq 0, r_1 \neq 0, r_2 \neq 0, v_3 \neq 0, v_s \neq 0\}.$$

Figure 2: The x-axis ellipse problem

On the basis of this projection, generic PCAD finishes after 12 min. This time we obtain a quantifier-free result containing 408 atomic formulas. Here we obviously suffer from our still preliminary solution formula construction code. Substitution of reasonable values for the parameters A, r_1, r_2 and subsequent simplifications indicate, however, that this formula actually describes the desired solution derived in [21].

5.2 X-Axis Ellipse Problem

The problem, which has been suggested by Lazard in [15], is to write down conditions such that the ellipse

$$\frac{(x-c)^2}{a^2} + \frac{(y-d)^2}{b^2} - 1 = 0$$

is inside the circle $x^2 + y^2 - 1 = 0$. We treat the special case $d = 0$; compare Figure 2. Our input formula reads as follows:

$$\forall x \forall y \big(b^2(x-c)^2 + a^2 y^2 - a^2 b^2 = 0 \longrightarrow x^2 + y^2 - 1 \le 0\big).$$

Due to clearing denominators, this description includes degenerate cases of exactly the sort observed by Chou [4] to be inevitable in general.

The projection order is $\overset{y}{\to} \overset{x}{\to} \mid \overset{c}{\to} \overset{b}{\to} \overset{a}{\to}$. With the regular projection operator PROJH, we successively obtain intermediate projection sets of the following sizes:

$$2 \to 9 \to \mid 18 \to 28 \to 32 \qquad \text{(20 ms)}.$$

On the basis of this projection set, regular PCAD succeeds after 1 min with a quantifier-free equivalent containing 4234 atomic formulas.

With our generic projection operator GPROJ, we obtain, in contrast, the following intermediate projection set sizes:

$$2 \to 9 \to \mid 16 \to 24 \to 24 \qquad \text{(10 ms)}.$$

In addition, we obtain the following theory:

$$\{a + b \ne 0, a - b \ne 0, a \ne 0, b \ne 0\}.$$

On this basis, generic PCAD returns after 7 s a quantifier-free formula containing 448 atomic formulas.

We next analyze the situation when using GPROJ but admitting only monomial assumptions. Here we obtain the

projection set sizes

$$2 \to 9 \to \mid 17 \to 25 \to 25 \qquad \text{(10 ms)}$$

together with the theory

$$\Theta = \{a \ne 0, b \ne 0, c \ne 0\}.$$

On this basis, generic PCAD yields after 8 s a quantifier-free formula φ' with 578 atomic formulas. Note that $a \ne 0$ and $b \ne 0$ are obviously non-degeneracy conditions.

We finally use Θ and φ' obtained by this last generic PCAD, and complete it to a quantifier-free equivalent of the input formula. For this we additionally treat the three cases excluded by Θ. Applying regular PCAD to the original problem with 0 substituted for a, we obtain "false" in less than 10 ms. The same holds for the case $b = 0$. For the case $c = 0$ we obtain in 240 ms a quantifier-free equivalent φ'_c with 76 atomic formulas. Combining these results, we have

$$\mathbb{R} \models \left(\left(\bigwedge \Theta \wedge \varphi'\right) \vee (c = 0 \wedge \varphi'_c)\right) \longleftrightarrow \varphi.$$

Our quantifier-free formula obtained by one generic PCAD plus three regular PCAD's on special cases contains $3 + 578 + 1 + 76 = 658$ atomic formulas. It requires an overall computation time of less than 9 s. This is a considerable improvement compared to the straightforward PCAD with 4234 atomic formulas in 1 min.

6. CONCLUSIONS

We have defined a generic projection operator for PCAD and described how to perform generic quantifier elimination on the basis of this operator. Our techniques are implemented within the computer logic system REDLOG. By means of two highly non-trivial quantifier elimination examples, we have demonstrated that our generic approach has a significantly extended application range compared to regular PCAD. This particularly affects input formulas with polynomials of high degree and many parameters thus filling a gap in the applicability of real quantifier elimination techniques.

7. ACKNOWLEDGMENTS

8. REFERENCES

[1] ARNON, D. S., COLLINS, G. E., AND MCCALLUM, S. Cylindrical algebraic decomposition I: The basic algorithm. *SIAM Journal on Computing 13*, 4 (Nov. 1984), 865–877.

[2] BROWN, C. W. Guaranteed solution formula construction. In *Proceedings of the ISSAC 99*, S. Dooley, Ed. ACM Press, New York, NY, July 1999, pp. 137–144.

[3] BROWN, C. W. Improved projection for cylindrical algebraic decomposition. *Journal of Symbolic Computation 32*, 5 (Nov. 2001), 447–465.

[4] CHOU, S.-C. *Mechanical Geometry Theorem Proving.* Mathematics and its applications. D. Reidel Publishing Company, Dordrecht, Boston, Lancaster, Tokyo, 1988.

[5] COLLINS, G. E. Quantifier elimination for the elementary theory of real closed fields by cylindrical algebraic decomposition. In *Automata Theory and Formal Languages. 2nd GI Conference*, H. Brakhage, Ed., vol. 33 of *Lecture Notes in Computer Science.* Springer-Verlag, Berlin, Heidelberg, New York, 1975, pp. 134–183.

[6] COLLINS, G. E., AND HONG, H. Partial cylindrical algebraic decomposition for quantifier elimination. *Journal of Symbolic Computation 12*, 3 (Sept. 1991), 299–328.

[7] DOLZMANN, A., AND STURM, T. Redlog: Computer algebra meets computer logic. *ACM SIGSAM Bulletin 31*, 2 (June 1997), 2–9.

[8] DOLZMANN, A., AND STURM, T. Simplification of quantifier-free formulae over ordered fields. *Journal of Symbolic Computation 24*, 2 (Aug. 1997), 209–231.

[9] DOLZMANN, A., STURM, T., AND WEISPFENNING, V. A new approach for automatic theorem proving in real geometry. *Journal of Automated Reasoning 21*, 3 (1998), 357–380.

[10] HENNIG, E. Rechnergestützte Dimensionierung analoger Schaltungen auf der Basis symbolischer Analyseverfahren. Tech. rep., Zentrum für Mikroelektronik, Universität Kaiserslautern, Dec. 1995. Annual report on a research project.

[11] HENNIG, E., AND HALFMANN, T. Analog Insydes tutorial. ITWM, Kaiserslautern, Germany, 1998.

[12] HONG, H. An improvement of the projection operator in cylindrical algebraic decomposition. In *Proceedings of the International Symposium on Symbolic and Algebraic Computation (ISSAC 90)*, S. Watanabe and M. Nagata, Eds. ACM Press, New York, Aug. 1990, pp. 261–264.

[13] HONG, H. Simple solution formula construction in cylindrical algebraic decomposition based quantifier elimination. In *Proceedings of the International Symposium on Symbolic and Algebraic Computation (ISSAC 92)*, P. S. Wang, Ed. ACM Press, New York, July 1992, pp. 177–188.

[14] HONG, H. Quantifier elimination for formulas constrained by quadratic equations via slope resultants. *THE Computer Journal 36*, 5 (1993), 440–449. Special issue on computational quantifier elimination.

[15] LAZARD, D. Quantifier elimination: Optimal solution for two classical examples. *Journal of Symbolic Computation 5*, 1&2 (Feb. 1988), 261–266.

[16] MCCALLUM, S. Solving polynomial strict inequalities using cylindrical algebraic decomposition. *THE Computer Journal 36*, 5 (1993), 432–438. Special issue on computational quantifier elimination.

[17] MCCALLUM, S. An improved projection operation for cylindrical algebraic decomposition. In *Quantifier Elimination and Cylindrical Algebraic Decomposition*, B. F. Caviness and J. R. Johnson, Eds. Springer-Verlag, Wien, 1998, pp. 242–268.

[18] SEIDL, A. An efficient representation of algebraic numbers and polynomials for cylindrical algebraic decomposition (extendend abstract). In *Proceedings of the 8th Rhine Workshop on Computer Algebra* (Mannheim, Germany, 2002), pp. 167–172.

[19] STRZEBOŃSKI, A. Solving systems of strict polynomial inequalities. *Journal of Symbolic Computation 29*, 3 (Mar. 2000), 471–480.

[20] STURM, T. *Real Quantifier Elimination in Geometry.* Doctoral dissertation, Department of Mathematics and Computer Science. University of Passau, Germany, D-94030 Passau, Germany, Dec. 1999.

[21] STURM, T. Reasoning over networks by symbolic methods. *Applicable Algebra in Engineering, Communication and Computing 10*, 1 (Sept. 1999), 79–96.

[22] WEISPFENNING, V. The complexity of linear problems in fields. *Journal of Symbolic Computation 5*, 1&2 (Feb.–Apr. 1988), 3–27.

[23] WEISPFENNING, V. Quantifier elimination for real algebra—the quadratic case and beyond. *Applicable Algebra in Engineering Communication and Computing 8*, 2 (Feb. 1997), 85–101.