Simple Truth Invariant CAD's and Solution Formula Construction

Christopher W. Brown
George E. Collins

July 30, 1996

Abstract

Given a prenex formula F of elementary real algebra having k free variables, the CAD method of quantifier elimination produces a truth invariant CAD, D, for F, that is, a CAD of k-dimensional real space in each cell of which F is truth-invariant, together with an assignment to each cell of its truth value. The method also produces a set P of irreducible projection polynomials, the real zeros of which form the boundaries of the cells. Typically there exists a coarser truth-invariant CAD, D', for F and a subset, P', of P whose real zeros form the boundaries of D'. We describe an efficient method for deriving such a D' and P' from D and P. The original set P may not suffice for construction of a solution formula (a quantifier-free formula equivalent to F), but P' will suffice if P does. We describe a method for augmenting P' in this case to a larger set that suffices. In either case the task of solution formula construction is much reduced by the smaller size of P'.

1 Introduction

The method of quantifier elimination by cylindrical algebraic decomposition takes a formula from the elementary theory of real closed fields as input, and constructs a cylindrical algebraic decomposition (CAD) of the space of unquantified variables. This decomposition is *truth invariant* with respect to the input formula, meaning that the formula is either identically true or identically false in each cell of the decomposition. The method determines the truth of the input formula for each cell, and then uses the CAD to construct a solution formula — a quantifier free formula that is equivalent to the input formula. It is this final phase of the method, solution formula construction, that this paper concerns.

In his 1975 paper, Collins [Col75] constructed solution formulas by producing a defining formula for each cell of the decomposition in which the input formula was true, and taking the disjunction of these defining formulas. He introduced the augmented projection operator to ensure that a defining formula can always be produced for each cell of the truth invariant CAD his method produces. Augmented projection, however, is prohibitively expensive. Hong [Hon90b, Hon92] abandoned the idea of the augmented projection operator with the observation that a solution formula can usually be constructed without it, using the polynomials computed during projection. His method for solution formula construction uses multiple-valued logic minimization to produce a *simple* solution formula when the CAD allows it, and otherwise to produce simple necessary and sufficient conditions.

Hong's method produces simple solution formulas in most cases. However, in some cases it is impractically slow, and in others it is unable to construct a formula because the projection factor set does not suffice for it. This paper seeks to remedy both of these problems. A method is given for constructing a "simple" CAD of free variable space from the CAD of free variable space computed by the methods of [Col75, Hon90b, CH91]. This simple CAD can be used as input for Hong's solution formula construction algorithm, which can result in considerable speed-ups. Also, a method is given for constructing a truth invariant CAD of free variable space that is guaranteed to allow solution formula construction. This method, like the augmented projection, uses derivatives of projection polynomials to ensure that a solution formula can be constructed. However, it will usually use few derivatives, and thus avoids the prohibitive inefficiency of the augmented projection.

The paper is organized as follows: Section 2 introduces some terminology and assumptions that will be used throughout the paper. The construction of a "simple" CAD and its application to simple solution formula construction is discussed in section 3. Finally, a method for constructing a CAD that ensures that a solution formula can be constructed is given in section 4.

2 Terminology and assumptions

The following assumptions and terminology will be used throughout this paper:

- A truth invariant CAD will be referred to as *self-definable* if there is a quantifier free formula using exclusively the polynomials in the CAD's projection factor set that is satisfied by all true cells in the CAD, and no false cells.
- Both algorithms described in this paper assume that the projection and CAD construction phases described in [Col75] or [CH91] have already occurred. It will be assumed that the quantified input formula contains the free variables $x_1, \ldots, x_n$, and that the projection operator, call it $PROJ$, projects from x_n down to x_1. The projection factor set produced by the

method will be called P, and the truth invariant CAD D.

- For ease of exposition it will be assumed that D is sign-invariant. This means that in each cell of the decomposition every projection factor has invariant sign. In some cases, the partial CAD method of Hong produces a CAD of free variable space for which this is not the case. Also the use of equational constraints [Col94] may result in a CAD of free variable space that is not sign-invariant. The algorithms presented here can be straightforwardly generalized to handle these cases, and indeed our implementations do. The examples given in this paper make use of partial CAD construction and equational constraints.
- The set of polynomials in P with positive degree in x_i, but of degree 0 in all x_j such that $j > i$ will be denoted by P_i. Polynomials in P_i will also be referred to as being of *level* i.
- "The derivative of a projection factor p", will always mean the derivative with respect to the variable in p of largest index.

3 Simple truth invariant CAD's

We begin with an example to illustrate what we mean by a simple truth invariant CAD. Consider the following quantified formula:

$$(\exists z)\,[\quad x^2 + y^2 + (z-3)^2 < 9 \quad \wedge \quad 18z - x + 7y < 15 \quad]$$

This formula defines a sphere and a half-space and asks for the (x, y) pairs which are images of points in the their intersection. Figure 1 shows the CAD produced for this problem. The shaded region is the solution space, i.e. the (x, y) pairs which satisfy the quantified formula.

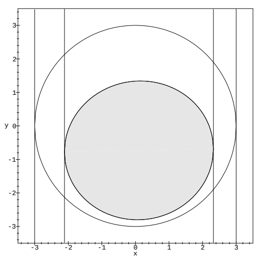

Figure 1: A CAD which could be simpler

Removing the outer circle and the two vertical lines tangent to it, will clearly leave a CAD which is truth invariant with respect to the input formula. We consider this CAD to be a simpler CAD, and whenever possible, we would like to make this kind of simplification. We will say that CAD D_1 is simpler that CAD D_2, if the projection factor set of D_1 is a proper subset of the projection factor set of D_2.

3.1 What are we looking for?

Our search for a simple truth-invariant CAD is restricted to CAD's defined by $PROJ$-closed subsets of P. These CAD's are interesting because they are easy to compute from the original CAD. Finding the $PROJ$-closure of a subset of P amounts to little more than book keeping, since all the resultant computation and factorization was done during the original projection. Furthermore, all of the sample points, the root isolation, and other computations that are needed for constructing a CAD from the new projection factor set can be taken directly from the original CAD.

3.2 Constructing a simpler truth invariant CAD

We present an algorithm that delivers a "minimal" truth invariant CAD. The CAD is minimal in the sense that removing any projection factor from its projection factor set either destroys the $PROJ$-closure property, or results in a CAD that is not truth invariant. The method removes projection factors from P, starting at level n and proceeding down to level 1, but ensures that P remains a $PROJ$-closed set defining a truth invariant CAD.

First, we determine which level n projection factors to remove. Let a be a section cell in D, and let b and c be its two neighbors in the same stack. If $a \cup b \cup c$ is not a truth invariant region, then P must retain at least one polynomial that is identically zero in a. Thus, for every section cell a that has a stack neighbor with a different truth value, we have the condition on P that $P \cap S_a \neq \emptyset$, where S_a is the set of all polynomials in P_n that are identically zero in a. This condition is also sufficient to ensure that P determines a truth invariant CAD. We construct the set $S = \{S_a \mid a$ is a section cell with a stack neighbor of opposite truth value $\}$, and let Q be a minimal hitting set for S. (Note: If A is a set of objects, and T is a set of subsets of A, a subset of A is called a hitting set of T if it has a non-empty intersection with every element of T.) All level n polynomials which are not in Q are removed from P. The CAD D is changed accordingly,i.e. to that defined by the new P. From the construction of P, it is clear that each cell in the new CAD will be the union of consecutive elements of a stack in the old CAD, all of which have the same truth value. The truth value and sample points of any one of these cells can be taken for the new cell.

We now consider removing level $n-1$ polynomials from P. Since we require P to be $PROJ$-closed, any level $n-1$ projection factor in the $PROJ$-closure of

P_n must be retained. However, we may have to keep other level $n-1$ projection factors as well. Once again we consider some section cell a, but this time in the CAD of $n-1$ space defined by $P \backslash P_n$. If some level $n-1$ element of the $PROJ$-closure of P_n is zero in a, then we consider a no further. If not, we know that the sections of the elements of P_n form a stack over $a \cup b \cup c$, where b and c are again the neighbors of a in the same stack. Therefore, a, b and c could be merged into a single cell, and D would still be a CAD. However, it might not be truth invariant. It is easy to check whether or not this is the case by simply comparing the truth values of cells in the three stacks. If merging the stacks would create a CAD which is not truth invariant, then we have once again the condition on P that it must retain at least one level $n-1$ polynomial that is identically zero in a. In other words, we have $P \cap S_a \neq \emptyset$, where S_a is the set of all level $n-1$ polynomials that are zero in a. Once again we collect all the S_a's in a set S, and construct Q, a minimal hitting set for S. Any level $n-1$ polynomial which is neither in Q nor in the $PROJ$-closure of P_n is removed from P. Once again the CAD D is changed accordingly.

The above method for removing level $n-1$ polynomials from P suggests an iterative procedure for removing lower level polynomials from P. After every step, P remains a $PROJ$-closed set, and D, the CAD defined by P, remains truth invariant. Because we always choose our hitting sets to be minimal (we can even require them to be of minimum size), we can be sure that removing any polynomial from the final P will result either in a set of polynomials that is not $PROJ$-closed, or in a set of polynomials that defines a CAD which is not truth invariant.

3.3 Examples of minimal truth invariant CAD's

With the help of QEPCAD, an implementation of the quantifier elimination by partial CAD algorithm based on the ideas found in [Hon90a, Hon90b, CH91, Hon92] , we can look at some non-trivial examples of the CAD produced during quantifier elimination by CAD. We have implemented the algorithm for constructing a minimal CAD outlined in the previous section and integrated it into QEPCAD, allowing us to compare the minimal CAD with the original CAD. Note: The McCallum [McC84] projection is used in all examples, with equational constraints where possible.

3.3.1 The complex product of an edge and a square

Consider the complex segment $L = \{x + i \mid x \in [0, 2]\}$, and the complex square $S = \{x + iy \mid x \in [2, 4], y \in [-1, 1]\}$. We use quantifier elimination to find the complex product of S and L. Since we have the easy necessary conditions that the product lies in the box $[-1, 9] \times i[0, 6]$, the product can be expressed as all pairs (x, y) satisfying:

$$(\exists x_1)(\exists x_2)(\exists y_2)\left[\begin{array}{r} x = x_1x_2 - y_2 \quad \wedge \quad y = x_1y_2 + x_2 \quad \wedge \\ 0 \leq x_1 \leq 2 \quad \wedge \quad 2 \leq x_2 \leq 4 \quad \wedge \quad -1 \leq y_2 \leq 1 \\ \wedge \quad -1 \leq x \leq 9 \quad \wedge \quad 0 \leq y \leq 6 \end{array}\right]$$

We add these necessary conditions to the quantified formula because QEPCAD will use them to speed up the computation. Figure 2 shows two CAD's that are truth invariant with respect to this input formula - the shaded regions are those in which the formula is satisfied. The first CAD is the original one produced by the quantifier elimination process. The second is the minimal truth invariant CAD constructed from the original CAD by the algorithm outlined above. Table 1 gives a quantitative comparison of the two CAD's. Construction of the

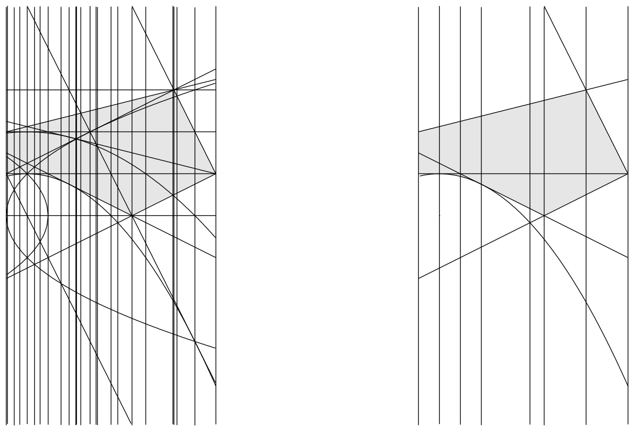

Figure 2: The original and minimal CAD's

	Proj. fac.'s		Cells	
	level 1	*level 2*	*level 1*	*level 2*
Original CAD	70	17	135	1693
Minimal CAD	14	6	27	179

Table 1: Complex product of an edge and a square

original CAD was accomplished by QEPCAD in 838.131 seconds (times do not include garbage collections). The minimal truth invariant CAD was constructed in 0.533 seconds.

3.3.2 The x-axis ellipse problem

[AM88] Consider an ellipse with center $(c, 0)$ and semi-major and semi-minor axes parallel to the coordinate axes with lengths a and b. We can express as a quantified formula the condition that this ellipse lies completely inside the unit disc. By symmetry it suffices to consider non-negative values for a, b, and c, and it is also clearly necessary that each of these values be at most 1. We obtain the following quantified formula:

$$(\forall x)(\forall y)\left[\begin{array}{l} 0 \le a \le 1 \quad \wedge \quad 0 \le b \le 1 \quad \wedge \quad 0 \le c \le 1 \quad \wedge \\ \left[(x-c)^2/a^2 + y^2/b^2 = 1 \Rightarrow x^2 + y^2 \le 1\right] \end{array}\right]$$

There are three free variables in this formula, so we don't provide pictures of the CAD's. However, Table 2 gives a quantitative comparison of the original and minimal CAD's. Computing the minimal CAD required 0.08 seconds, while

	Proj. fac.'s			Cells		
	level 1	*level 2*	*level 3*	*level 1*	*level 2*	*level 3*
Original CAD	7	9	7	15	93	471
Minimal CAD	3	6	3	7	29	75

Table 2: x-axis ellipse problem

60.75 seconds were used in constructing the original truth invariant CAD.

3.4 Simple Solution Formula Construction

In the introduction, the search for a simple CAD was motivated by the observation that Hong's simple solution formula construction algorithm sometimes requires a prohibitive amount of time, and with the claim that using Hong's method with a simpler CAD as input would improve this situation. Both examples from the previous section are in accord with this claim. For the first example, QEPCAD's solution formula construction algorithm returned a solution formula after 412 seconds when the minimal CAD was used as input. With the original CAD as input, no solution formula was constructed after more than 10 times that length of time, at which point the computation was terminated. In the second example a solution formula was computed in 1.3 seconds using the minimal CAD as input, and 50.3 seconds using the original CAD. The same formula was produced for the two different inputs.

In these examples one sees that a minimal truth invariant CAD can significantly simplify the task of solution formula construction. Unfortunately this is not always the case. The process of constructing a simpler CAD does not preserve self-definability. That is, it may happen that the original CAD is self-definable while the minimal CAD is not. For solution formula construction a minimal *self-definable* CAD is more appropriate than a minimal truth invariant CAD. However, a minimal truth invariant CAD can help in constructing a minimal self-definable CAD.

Self-definability says something about the projection factor set of a CAD. Since a solution formula can be constructed from the projection factor set of a self-definable CAD, it is clear that for any pair of cells in the CAD, one true and one false, there must be at least one projection factor that has different signs in the two cells. This obvious necessary condition is also sufficient.

We restate the condition as follows: For a pair of cells (a, b) in D, denote by $S_{a,b}$ the set of all projection factors that have different signs in the two cells. If S is the set $\{S_{a,b} | a \text{ is true and } b \text{ false}\}$, then the CAD defined by a subset of P is self-definable if and only if that subset is a hitting set for S. We can construct a self-definable CAD by constructing a hitting set for S, and using its $PROJ$-closure as a new projection factor set. Unfortunately this requires considering all pairs of true and false cells. This has worst case quadratic complexity in the number of cells. In tests, finding a minimal hitting set for S has proven to be much more expensive than minimal truth invariant CAD construction. For the complex product of the edge and a square problem, for example, the set S has 759934 elements. Just constructing it is expensive.

However, if one begins with the projection set Q defining a minimal truth invariant CAD, one can just decide which polynomials of P need to be added to Q in order to restore self-definability. The procedure is the same as that outlined above, except that S is defined as all $S_{a,b}$ such that a is a true cell and b is false and all elements of Q have the same sign in a as in b. This will, in general, be much smaller than the S defined previously. Additionally, the set can be constructed quickly. By sorting, one can partition the cells according to the signs of the elements of Q, and then construct the sets $S_{a,b}$ only for cells in the same block. The time required to construct S in this way is dominated by the cardinality of S plus $n \log n$, where n is the total number of cells.

So, in cases where the minimal CAD loses the property of self-definability, we have a simple and fast procedure for constructing from it a self-defineable CAD.

4 When solution formula construction fails

Without the augmented projection, it is possible that the quantifier elimination process produces a CAD that is truth invariant, but not self-definable. If this occurs, a self-definable CAD must be produced before solution formula construction can continue. In this section we present an algorithm for doing this.

The strategy behind the augmented projection is to guarantee a priori that the CAD produced during quantifier elimination is self-definable by introducing all possible derivatives into the projection factor set. To produce a self-definable CAD from a CAD that is merely truth invariant we will also introduce derivatives of the projection factor set. However, we introduces derivatives of only a subset (hopefully small) of P, and we introduce them one at a time until the CAD is self-definable. Thus, we hope usually to introduce few new projection factors.

4.1 Choosing polynomials to introduce

We will define the notion of a *conflicting pair* of cells in a CAD, and see from three simple propositions that they contain the information necessary to determine which derivatives should be added to the projection set.

Definition 1 *Two level i cells a and b in the same stack are a* conflicting pair *if there exist cells a' and b' of level at least i such that:*

1. *the projections onto i-space of a' and b' are a and b respectively,*
2. *each projection factor that is sign invariant in both a' and b' has the same sign in both cells (i.e. the signatures of a' and b' are compatible), and*
3. *the input formula is true in a' and false in b'.*

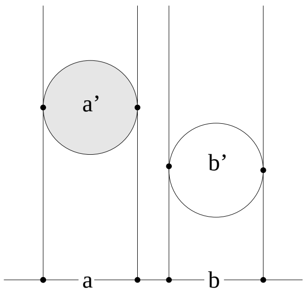

Figure 3: A conflicting pair

Figure 3 gives an example of a conflicting pair a and b. The shaded region satisfies the input formula, the unshaded regions do not. In this example we have exactly one level one and one level two projection factor, so a' and b' do indeed have the same signatures. Since they are not in the same stack, however, they are not a conflicting pair. By requiring the cells to be in the same stack, we reduce the problem of distinguishing between two level two cells to distinguishing between two level one cells. If we can distinguish between a and b, then we can distinguish between a' and b'.

The following proposition explains the relevance of conflicting pairs to constructing self-definable CAD's.

Proposition 1 *D is self-definable if and only if it contains no conflicting pair.*

The next proposition says in essence that adding a polynomial to the projection factor set cannot create "new" conflicting pairs.

Proposition 2 *Let $\overline{P}$ be a PROJ-closed set of polynomials containing P, and let $\overline{D}$ be the CAD defined by $\overline{P}$. If $(\overline{a}, \overline{b})$ is a conflicting pair of level j cells in $\overline{D}$, then there is a conflicting pair (a, b) in D of level at least j such that $\overline{a}$ and $\overline{b}$ are contained respectively in the projections onto j-space of a and b.*

If there is a conflicting pair (a, b) in D, we would like to remove it by adding derivatives of projection factors into the projection factor set. The following proposition gives a criterion for choosing a projection factor whose derivatives suffice to remove (a, b).

Proposition 3 *Let (a, b) be a level i conflicting pair in D. Let p be a level i projection factor that is zero in a or b or some cell that is in the same stack as a and b and lies between them. Let A be the set of all derivatives of p with respect to x_i. Finally, let $\overline{P}$ be the PROJ-closure of $P \cup A$, and $\overline{D}$ the CAD $\overline{P}$ defines. There is no conflicting pair $(\overline{a}, \overline{b})$ in $\overline{D}$ such that $\overline{a} \subseteq a$ and $\overline{b} \subseteq b$.*

These propositions are enough to give us a strategy for choosing which derivatives to add to the projection factor set. Let i be the highest level of any conflicting pair, and Q be a subset of the level i projection factors such that for every level i conflicting pair there is an element of Q satisfying the conditions of proposition 3. Propositions 2 and 3 tell us that adding all derivatives of polynomials in Q to the projection factor set results in a CAD that has no conflicting pairs of level i or higher.

Following our strategy, we remove all conflicting pairs from the top level down. Since we hope that only a few derivatives will be required, we add them one at a time and check after each addition to see if the resulting CAD is self-definable. An outline of an algorithm based on this strategy is given below. For a level i conflicting pair (a, b), denote by $T_{a,b}$ the set of level i projection factors satisfying Proposition 3.

1. Set i to the highest level in which a conflicting pair occurs, if there are no conflicting pairs then stop.
2. Set Q to a minimum hitting set for $\{T_{a,b} \mid (a, b) \text{ a level } i \text{ conflicting pair}\}$.
3. Set S to the set of all factors of all derivatives of the elements of Q.
4. Add to P some element of S not already in P, take the closure under the projection operation, and build the corresponding CAD.
5. If there are still level i conflicting pairs, goto step 4, otherwise goto Step 1.

4.2 Examples of extending the projection factor set

We have implemented the procedure outlined above and integrated it into QEPCAD. We give two examples here, examples which simply show that it does happen that the quantifier elimination by CAD method constructs a truth invariant CAD which is not definable, and which show that (at least for these examples) our method for constructing a definable CAD from a truth invariant CAD is not expensive compared to other phases of the quantifier elimination by CAD algorithm. In our outline of the algorithm, we did not specify the order in which we added the elements of S. In this implementation we introduce factors of lower order derivatives first. It is likely that there are better or worse strategies for this, and evaluation of different strategies is a possible direction for future work.

4.2.1 The two "spheres" problem

We gave Figure 3 as an example to illustrate the definition of conflicting pair. That CAD is a truth invariant CAD for the following quantifier elimination problem:

$$(\exists z)\left[\begin{array}{l} 10 - 12xz - 2x - 6y - 6z + 2xy^2 + 2x^3 + 3y^2 - x^2 + 2x^2y^2 + \\ 2x^2z^2 - 12xy + 2y^2z^2 - 7zy + x^4 + y^4 + z^4 + 3z^2 + 2xz^2 < 0 \\ \quad \wedge \;\; x + y + 2z > 0 \end{array}\right]$$

This problem was manufactured to produce that particular CAD. The point that it is meant to illustrate is that in choosing a projection factor whose derivative we will introduce, we should not only look at projection factors of the highest level. Our definition of a conflicting pair gives us a way to determine which level of projection factors we should consider.

QEPCAD constructed a truth invariant CAD for this example in 36.6 seconds. This CAD is not self-definable. A self-definable CAD was constructed by the method outlined above in 3.7 seconds. Two factors of derivatives were added to the projection factor set, both univariate.

4.2.2 The fixed size ellipse problem

Here, as in the second example of Section 3.3, we consider a special case of the ellipse problem posed by Kahan [Kah75]. We fix a size and shape for an ellipse (semi-major axis of length 2/3 that is parallel to the x-axis and a semi-minor axis of length 1/3) and ask for necessary and sufficient conditions for its center such that the ellipse is tangent to the unit circle, and inside the circle. Because of symmetry it suffices to consider only positions for the ellipse's center and for the tangency point in the first quadrant. We use this in order to speed up the computation. Thus we arrive at the following quantified formula:

$$(\exists x)(\exists y)\left[\begin{array}{l} c^2+d^2<1 \quad \wedge \quad 36(y-d)^2+9(x-c)^2=4 \quad \wedge \\ y^2+x^2=1 \quad \wedge \quad (3x+c)y-4xd=0 \quad \wedge \\ 0<c<1 \quad \wedge \quad 0<d<1 \quad \wedge \quad 0<x<1 \end{array}\right]$$

For this problem also (using the McCallum projection) a truth invariant CAD is produced that is not self-definable. Approximately 190 seconds were required by QEPCAD to construct this CAD. Slightly less than 33 seconds were required to construct a self-definable CAD. Two factors of derivatives were introduced.

4.3 Other improvements

We have seen in some examples that unnecessary polynomials are often generated during the projection phase. By unnecessary we mean polynomials that may be removed from the projection set, leaving a smaller $PROJ$-closed set which still defines a truth invariant CAD. When the CAD produced during quantifier elimination is self-definable, we considered the strategy of removing some of these unnecessary polynomials before attempting to construct a solution formula. When the original CAD is not self-definable, we might also consider doing the same thing before introducing a new polynomial into the projection factor set — especially because we will have to proceed by computing the $PROJ$-closure of the projection factor set augmented by the new polynomial, and by constructing the CAD that the $PROJ$-closure defines. Both of these operations take more time if more projection factors are present.

References

[AM88] D. S. Arnon and M. Mignotte. On mechanical quantifier elimination for elementary algebra and geometry. *Journal of Symbolic Computation*, 5(1,2):237–260, 1988.

[CH91] G. E. Collins and H. Hong. Partial cylindrical algebraic decomposition for quantifier elimination. *Journal of Symbolic Computation*, 12(3):299–328, Sep 1991.

[Col75] G. E. Collins. Quantifier elimination for the elementary theory of real closed fields by cylindrical algebraic decomposition. In *Lecture Notes In Computer Science*, pages 134–183. Springer-Verlag, Berlin, 1975. Vol. 33.

[Col94] G. E. Collins. Quantifier elimination by cylindrical algebraic decomposition - 20 years of progress. In B. Caviness and J. Johnson, editors, *Quantifier Elimination and Cylindrical Algebraic Decomposition*, Texts and Monographs in Symbolic Computation. Springer-Verlag, 1994. To appear.

[Hon90a] H. Hong. An improvement of the projection operator in cylindrical algebraic decomposition. In *International Symposium of Symbolic and Algebraic Computation*, pages 261–264, 1990.

[Hon90b] H. Hong. *Improvements in CAD–based Quantifier Elimination*. PhD thesis, The Ohio State University, 1990.

[Hon92] H. Hong. Simple solution formula construction in cylindrical algebraic decomposition based quantifier elimination. In *International Conference on Symbolic and Algebraic Computation ISSAC-92*, pages 177–188, 1992.

[Kah75] W. Kahan. Problem no. 9: An ellipse problem. *SIGSAM Bulletin of the Assoc. Comp. Mach.*, 9(35):11, 1975.

[McC84] S. McCallum. *An Improved Projection Operator for Cylindrical Algebraic Decomposition*. PhD thesis, University of Wisconsin-Madison, 1984.