Cylindrical Algebraic Decompositions for Boolean Combinations

Russell Bradford
University of Bath
R.J.Bradford@bath.ac.uk

James H. Davenport
University of Bath
J.H.Davenport@bath.ac.uk

Matthew England
University of Bath
M.England@bath.ac.uk

Scott McCallum
Macquarie University
Scott.McCallum@mq.edu.au

David Wilson
University of Bath
D.J.Wilson@bath.ac.uk

ABSTRACT

This article makes the key observation that when using cylindrical algebraic decomposition (CAD) to solve a problem with respect to a set of polynomials, it is not always the signs of those polynomials that are of paramount importance but rather the truth values of certain quantifier free formulae involving them. This motivates our definition of a Truth Table Invariant CAD (TTICAD). We generalise the theory of equational constraints to design an algorithm which will efficiently construct a TTICAD for a wide class of problems, producing stronger results than when using equational constraints alone. The algorithm is implemented fully in MAPLE and we present promising results from experimentation.

Categories and Subject Descriptors

I.1.2 [**Symbolic and Algebraic Manipulation**]: Algorithms—*Algebraic algorithms, Analysis of algorithms*

General Terms

Algorithms, Experimentation, Theory

Keywords

cylindrical algebraic decomposition; equational constraint

1. INTRODUCTION

Cylindrical algebraic decompositions (CADs) are a key tool in real algebraic geometry, both for their original motivation, solving quantifier elimination problems, but also for use in many other applications ranging from robot motion planning [22, etc.] to programming with complex functions [12, etc.]. Traditionally CADs are produced sign-invariant to a given set of polynomials, (the signs of the polynomials do not vary on the cells of the decomposition). However, this gives far more information than required for most problems. The idea of a truth invariant CAD (the truth of a formula does not vary on each cell) was defined in [2] for use in simplifying CADs. The key contribution of this paper is an approach to construct CADs which are truth invariant without having to first build a sign-invariant CAD. Actually, we directly build CADs which are truth table invariant, (the truth values of various quantifier free formulae do not vary).

We present an algorithm to efficiently produce TTICADs for a wide class of problems, utilising the theory of equational constraints [19]. The algorithm goes further than equational constraints by allowing the creation of smaller CADs in a wider variety of cases; for example disjunctive normal form where each individual conjunction has an equational constraint but no single explicit equational constraint is present for the formula. The problem of decomposing complex space according to a set of branch cuts for the purpose of algebraic simplification ([21, etc.]) is of this case.

ISSAC'13, June 26–29, 2013, Boston, Massachusetts, USA.

1.1 Background on CAD

We briefly remind the reader about the theory of CAD, first proposed by Collins in [9].

DEFINITION 1. *A* Tarski formula $F(x_1, \ldots, x_n)$ *is a Boolean combination* $(\wedge, \vee, \neg)$ *of statements about the signs,* $(= 0, > 0, < 0$*, but therefore* $\neq 0, \geq 0, \leq 0$ *as well), of certain integral polynomials* $f_i(x_1, \ldots, x_n)$*. We use* QFF *to denote a quantifier free Tarski formula.*

CAD was developed as a tool for the problem of quantifier elimination over the reals: given a quantified Tarski formula

$$Q_{k+1}x_{k+1} \ldots Q_n x_n F(x_1, \ldots, x_n) \tag{1}$$

(where $Q_i \in \{\forall, \exists\}$ and F is a QFF), produce an equivalent QFF $\psi(x_1, \ldots, x_k)$. Collins proposed to decompose $\mathbb{R}^n$ cylindrically such that each cell was sign-invariant for all f_i occurring in F. Then ψ would be the disjunction of the defining formulae of those cells c_i in $\mathbb{R}^k$ such that (1) was true over the whole of c_i, which is the same as saying that (1) is true at any one "sample point" of c_i.

Collins' algorithm has two phases. The first, *projection*, applies a projection operator repeatedly to a set of polynomials, each time producing another set in one fewer variables. Together these sets contain the *projection polynomials.* These are then used in the second phase, *lifting*, to build the CAD incrementally. First $\mathbb{R}$ is decomposed into cells which are points and intervals corresponding to the real roots of the univariate polynomials. Then $\mathbb{R}^2$ is decomposed by repeating the process over each cell using the bivariate

polynomials at a sample point. The output for each cell consists of *sections* (where a polynomial vanishes) and *sectors* (the regions between). Together these form a *stack* over the cell, and taking the union of these stacks gives the CAD of $\mathbb{R}^2$. This is repeated until a CAD of $\mathbb{R}^n$ is produced.

To conclude that a CAD produced in this way is sign-invariant we need delineability. A polynomial is *delineable* in a cell if the portion of its zero set in the cell consists of disjoint sections. A set of polynomials are *delineable* in a cell if each is delineable and the sections of different polynomials in the cell are either identical or disjoint. The projection operator used must be defined so that over each cell of a sign-invariant CAD for projection polynomials in r variables, the polynomials in $r+1$ variables are delineable.

The output of a CAD algorithm depends on the variable ordering. We usually work with polynomials in $\mathbb{Z}[x_1, \ldots, x_n]$ with the variables, $\mathbf{x}$, in ascending order (so we first project with respect to x_n and continue to reach univariate polynomials in x_1). The *main variable* of a polynomial (mvar) is the greatest variable present with respect to the ordering.

Major directions of work since 1975 includes the following:

1. Improvements in Collins' main algorithms by [17, and many others]. These have focussed on reducing the projection sets required as discussed further later.
2. Complexity theory of CAD [5, 13].
3. Partial CAD, introduced in [11], where the structure of F is used to lift only when required to deduce ψ.
4. The theory of equational constraints, [19, 20, 6] discussed in Section 2.1. This is related to the previous direction but differs by using more efficient projections.
5. CAD via Triangular Decomposition [8]: a radically different approach for computing a sign-invariant CAD which is used for Maple's inbuilt CAD command.

1.2 TTICAD

We define a new type of CAD, the topic of this paper.

Definition 2. *Let $\Phi = \{\phi_i\}_{i=1}^t$ be a list of QFFs. We say a cylindrical algebraic decomposition $\mathcal{D}$ is a* Truth Table Invariant *CAD for Φ (TTICAD) if the Boolean value of each ϕ_i is constant (either true or false) on each cell of $\mathcal{D}$.*

A full sign-invariant CAD for the set of polynomials occurring in the formulae of Φ would clearly be a TTICAD. However, we aim to produce an algorithm that will construct smaller TTICADs for certain Φ. We will achieve this using the theory of equational constraints (first suggested in [10] with the key theory developed in [19]).

Definition 3. *Suppose some quantified formula is given:*

$$\phi^* = (Q_{k+1}x_{k+1}) \cdots (Q_n x_n)\phi(\mathbf{x}).$$

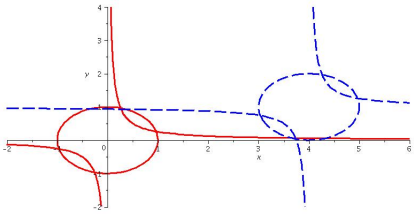
where the Q_i are quantifiers and ϕ is quantifier free. An equation $f = 0$ is called an **equational constraint** *of ϕ^* if $f = 0$ is logically implied by ϕ (the quantifier-free part of ϕ^*). Such a constraint may be either explicit or implicit.*

We suppose that we are given a formula list Φ in which every QFF ϕ_i has a designated explicit equational constraint $f_i = 0$. We will construct TTICADs by generalising McCallum's reduced projection operator for equational constraints (as in [19]) so that we may make use of the equational constraints.

1.3 Worked Example

We will provide details for the following worked example.

Figure 1: The polynomials from Section 1.3.



Consider the polynomials:

$$f_1 := x^2 + y^2 - 1 \qquad g_1 := xy - \tfrac{1}{4}$$
$$f_2 := (x-4)^2 + (y-1)^2 - 1 \qquad g_2 := (x-4)(y-1) - \tfrac{1}{4}$$

which are plotted in Figure 1. We wish to solve the following problem: find the regions of $\mathbb{R}^2$ where the formula

$$\Phi := (f_1 = 0 \wedge g_1 < 0) \vee (f_2 = 0 \wedge g_2 < 0)$$

is true. Assume that we are using the variable ordering $y \succ x$ (so the 1-dimensional CAD is with respect to x).

Both Qepcad [3] and Maple 16 [8] produce a full sign-invariant CAD for the polynomials with 317 cells. At first glance it seems that the theory of equational constraints [19, 20, 6] is not applicable here as neither $f_1 = 0$ nor $f_2 = 0$ is logically implied by Φ. However, while there is no explicit equational constraint we can observe that $f_1 f_2 = 0$ is an *implicit* constraint of Φ. Using Qepcad with this declared gives a CAD with 249 cells. Later, in Section 2.3 we demonstrate how a TTICAD with 105 cells can be produced.

2. PROJECTION OPERATORS

2.1 Equational Constraints

We use two key theorems from McCallum's work on projection and equational constraints. Both theorems use CADs which are not just sign-invariant but have the stronger property of order-invariance. A CAD is *order-invariant* with respect to a set of polynomials if each polynomial has constant order of vanishing within each cell.

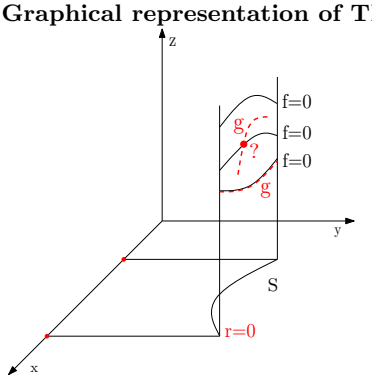
Let P be the McCallum projection operator [17], which produces coefficients, discriminant and cross resultants from a set of polynomials. We assume the usual trivial simplifications such as removal of constants, exclusion of entries identical to a previous entry (up to constant multiple), and using only the necessary coefficients. Recall that a set $A \subset \mathbb{Z}[\mathbf{x}]$ is an *irreducible basis* if the elements of A are of positive degree in the main variable, irreducible and pairwise relatively prime. The main theorem underlying P follows.

Theorem 1 ([18]). *Let A be an irreducible basis in $\mathbb{Z}[\mathbf{x}]$ and let S be a connected submanifold of $\mathbb{R}^{n-1}$. Suppose each element of $P(A)$ is order-invariant in S. Then each element of A either vanishes identically on S or is analytic delineable on S, (a slight variant on traditional delineability, see [18]). The sections of A not identically vanishing are pairwise disjoint, and each element of A not identically vanishing is order-invariant in such sections.*

The main mathematical result underlying the reduction of P in the presence of an equational constraint f is as follows.

Theorem 2 ([19]). *Let $f(\mathbf{x}), g(\mathbf{x})$ be integral polynomials with positive degree in x_n, let $r(x_1, \ldots, x_{n-1})$ be their*

Figure 2: Graphical representation of Theorem 2



resultant, and suppose $r \neq 0$. Let S be a connected subset of $\mathbb{R}^{n-1}$ such that f is delineable on S and r is order-invariant in S. Then g is sign-invariant *in every section of f over S.*

Figure 2 gives a graphical representation of the question answered by Theorem 2. Here we consider polynomials $f(x, y, z)$ and $g(x, y, z)$ of positive degree in z whose resultant r is non-zero, and a connected subset $S \subset \mathbb{R}^2$ in which r is order-invariant. We further suppose that f is delineable on S (noting that Theorem 1 with $n = 3$ and $A = \{f\}$ provides sufficient conditions for this). We ask whether g is sign-invariant in the sections of f over S. Theorem 2 answers this question affirmatively: the real variety of g either aligns with a given section of f exactly (as for the bottom section of f in Figure 2), or has no intersection with such a section (as for the top). The situation at the middle section of f cannot happen. Theorem 2 thus suggests a reduction of the projection operator P relative to an equational constraint $f = 0$ for the first projection step, as in [19].

2.2 A Projection Operator for TTICAD

In [19] the central concept is that of the reduced projection of a set A of integral polynomials relative to a nonempty subset E of A and it is an extension of this which is central here. For simplicity in [19], the concept is first defined for the case when A is an irreducible basis and by analogy we start with a similar special case. Let $\mathcal{A} = \{A_i\}_{i=1}^t$ be a list of irreducible bases A_i and let $\mathcal{E} = \{E_i\}_{i=1}^t$ be a list of nonempty subsets $E_i \subseteq A_i$. Put $A = \bigcup_{i=1}^t A_i$ and $E = \bigcup_{i=1}^t E_i$ (we will use the convention of uppercase Roman letters for sets and calligraphic letters for sequences).

Definition 4. *We define the* reduced projection of $\mathcal{A}$ with respect to $\mathcal{E}$, *denoted by $P_{\mathcal{E}}(\mathcal{A})$, as follows:*

$$P_{\mathcal{E}}(\mathcal{A}) := \bigcup_{i=1}^t P_{E_i}(A_i) \cup \mathrm{Res}^{\times}(\mathcal{E}) \tag{2}$$

where

$$P_{E_i}(A_i) = P(E_i) \cup \{\mathrm{res}_{x_n}(f, g) \mid f \in E_i, g \in A_i, g \notin E_i\}$$

$$\mathrm{Res}^{\times}(\mathcal{E}) = \{\mathrm{res}_{x_n}(f, \hat{f}) \mid \exists i, j : f \in E_i, \hat{f} \in E_j, i < j, f \neq \hat{f}\}$$

In Section 3.1 we build Algorithm 1 to apply the reduced projection operator for less special input sets by considering contents and irreducible factors of positive degree.

Definition 5. *The* excluded projection polynomials *of (A_i, E_i) are those in $P(A)$ but excluded from $P_{\mathcal{E}}(\mathcal{A})$:*

$$\begin{aligned}\mathrm{ExclP}_{E_i}(A_i) &:= P(A_i) \setminus P_{E_i}(A_i) \\ &= \{\mathrm{coeffs}(g), \mathrm{disc}_{x_n}(g), \mathrm{res}_{x_n}(g, \hat{g}) \mid g, \hat{g} \in A_i \setminus E_i, g \neq \hat{g}\}.\end{aligned} \tag{3}$$

The total set of excluded polynomials, denoted $\mathrm{ExclP}_{\mathcal{E}}(\mathcal{A})$, *consists of all the* $\mathrm{ExclP}_{E_i}(A_i)$, *along with the cross resultants of g_i with all of A_j for $i \neq j$.*

The following theorem is an analogue of Theorem 2.3 of [19], and provides the foundation for our algorithm in Section 3.1.

Theorem 3. *Let S be a connected submanifold of $\mathbb{R}^{n-1}$. Suppose each element of $P_{\mathcal{E}}(\mathcal{A})$ is order invariant in S. Then each $f \in E$ either vanishes identically on S or is analytically delineable on S, the sections over S of the $f \in E$ which do not vanish identically are pairwise disjoint, and each element $f \in E$ which does not vanish identically is order-invariant in such sections.*

Moreover, *for each i, with $1 \leq i \leq t$, every $g \in A_i \setminus E_i$ is sign-invariant in each section over S of every $f \in E_i$ which does not vanish identically.*

Proof. The crucial observation is that $P(E) \subseteq P_{\mathcal{E}}(\mathcal{A})$. To see this, recall equation (2) and note that we can write

$$P(E) = \textstyle\bigcup_i P(E_i) \cup \mathrm{Res}^{\times}(\mathcal{E}).$$

We can therefore apply Theorem 1 to the set E and obtain the first three conclusions immediately.

There remains the final conclusion to prove. Let i be in the range $1 \leq i \leq t$, let $g \in A_i \setminus E_i$ and let $f \in E_i$; suppose f does not vanish identically on S. Now $\mathrm{res}_{x_n}(f, g) \in P_{\mathcal{E}}(\mathcal{A})$, and so is order-invariant in S by hypothesis. Further, we already concluded that f is delineable. Therefore by Theorem 2, g is sign-invariant in each section of f over S. □

In the following section we can use Theorem 3 as the key tool for our implementation of TTICAD, so long as the equational constraint f does not vanish identically on the lower dimensional manifold, S. When working with a polynomial f considered in r variables that vanishes identically at a point $\alpha \in \mathbb{R}^{r-1}$ we say that f is *nullified* at α.

Remark 4. *It is clear that the reduced projection $P_{\mathcal{E}}(\mathcal{A})$ will lead to fewer (or the same) projection polynomials than the full projection P. One may consider instead using the reduced projection $P_E(A)$ of [19], (with $E = \cup_i E_i$ and $A = \cup_i A_i$ as above). In the context of Section 1.2 this corresponds to using $\prod_i f_i$ as an implicit equational constraint for a single formula. Note that $P_{\mathcal{E}}(\mathcal{A})$ also contains fewer polynomials than $P_E(A)$ in general since $P_E(A)$ contains all resultants $\mathrm{res}(f, g)$ where $f \in E_i, g \in A_j$ (and $g \notin E$), while $P_{\mathcal{E}}(\mathcal{A})$ contains only those with $i = j$ (and $g \notin E_i$).*

2.3 Worked Example

In Section 3 we will discuss how to use these results to define an algorithm for TTICAD. First we illustrate the potential savings with our worked example from Section 1.3.

In the notation introduced above we have:

$$A_1 := \{f_1, g_1\},\ E_1 := \{f_1\};\ A_2 := \{f_2, g_2\},\ E_2 := \{f_2\}.$$

We construct the reduced projection sets for each ϕ_i,

$$P_{E_1}(A_1) = \left\{x^2 - 1, x^4 - x^2 + \tfrac{1}{16}\right\},$$

$$P_{E_2}(A_2) = \left\{x^2 - 8x + 15, x^4 - 16x^3 + 95x^2 - 248x + \tfrac{3841}{16}\right\}$$

and the cross-resultant set

$$\mathrm{Res}^{\times}(\mathcal{E}) = \{\mathrm{res}_y(f_1, f_2)\} = \{68x^2 - 272x + 285\}.$$

Figure 3: The polynomials from the worked example along with the solutions to the projection sets.

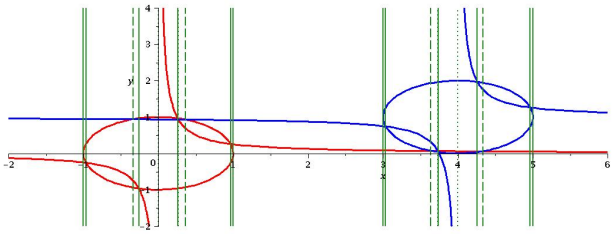
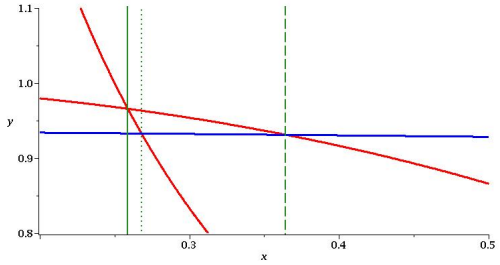


Figure 4: Magnified region of Figure 3



$P_{\mathcal{E}}(\mathcal{A})$ is then the union of these three sets. In Figure 3 we plot the polynomials (solid curves) and identify the 12 real solutions of $P_{\mathcal{E}}(\mathcal{A})$ (solid vertical lines). We can see the solutions align with the asymptotes of the fs and the important intersections (those of f_1 with g_1 and f_2 with g_2).

If we were to instead use a projection operator based on an implicit equational constraint $f_1 f_2 = 0$ then in the notation above we would construct $P_E(A)$ from $A = \{f_1, f_2, g_1, g_2\}$ and $E = \{f_1, f_2\}$. This set provides an extra 4 solutions (the dashed vertical lines) which align with the intersections of f_1 with g_2 and f_2 with g_1. Finally, if we were to consider $P(A)$ then we gain another 4 solutions (the dotted vertical lines) which align with the intersections of g_1 and g_2 and the asymptotes of the gs. In Figure 4 we magnify a region to show explicitly that the point of intersection between f_1 and g_1 is identified in $P_{\mathcal{E}}(\mathcal{A})$, whereas the intersection points of g_2 with both f_1 and g_1 are ignored.

Hence the 1-dimensional CAD produced using $P_{\mathcal{E}}(\mathcal{A})$ has 25 cells compared to 33 when using $P_E(A)$ and 41 when using $P(A)$. However, it is important to note that this reduction is amplified after lifting (using Theorem 3 and and Algorithm 1). The full dimensional TTICAD has 105 cells, the CAD invariant with respect to the implicit equational constraint has 249 cells and the full sign-invariant CAD has 317.

3. IMPLEMENTATION

3.1 Algorithm Description and Proof

We describe carefully Algorithm 1. This will create a TTICAD of $\mathbb{R}^n$ for a list of QFFs, $\Phi = \{\phi_i\}_{i=1}^t$, in variables $\mathbf{x} = x_1 \prec x_2 \prec \cdots \prec x_n$ where each ϕ_i has a designated equational constraint $f_i = 0$ of positive degree. We use a subalgorithm `CADW`, fully specified and validated in [18]. The input of `CADW` is: r, a positive integer and A, a set of r-variate integral polynomials. The output is a Boolean w which if true is accompanied by an order-invariant CAD for A (a list of indices I and sample points S).

Let A_i be the set of all polynomials occurring in ϕ_i, put $E_i = \{f_i\}$, and let $\mathcal{A}$ and $\mathcal{E}$ be the lists of the A_i and E_i, respectively. Our algorithm effectively defines the reduced projection of $\mathcal{A}$ with respect to $\mathcal{E}$ using the special case of this definition from the previous section. The definition amounts to using $\mathfrak{P} := C \cup P_{\mathcal{F}}(\mathcal{B})$ for $P_{\mathcal{E}}(\mathcal{A})$, where C is the set of contents of all the elements of all the A_i, $\mathcal{B}$ is the list $\{B_i\}_{i=1}^t$, such that B_i is the finest squarefree basis for the set $\text{prim}(A_i)$ of primitive parts of elements of A_i which have positive degree, and $\mathcal{F}$ is the list $\{F_i\}_{i=1}^t$, such that F_i is the finest squarefree basis for $\text{prim}(E_i)$. (The reader will notice that this notation and the definition of $P_{\mathcal{E}}(\mathcal{A})$ is analogous to the work in Section 5 of [19].)

Algorithm 1: TTICAD Algorithm

Input : A list of quantifier-free formulae $\Phi = \{\phi_i\}_{i=1}^t$ in variables $x_1, \ldots, x_n$. Each ϕ_i has a designated equational constraint $f_i = 0$.

Output: Either • $\mathcal{D}$: A TTICAD of $\mathbb{R}^n$ for Φ (described by lists I and S of cell indices and sample points, respectively); or • **FAIL**: If Φ is not well oriented (Def. 6).

for $i = 1 \ldots t$ **do**
  Set $E_i \leftarrow \{f_i\}$. Compute the finest squarefree basis F_i for $\text{prim}(E_i)$;
Set $F \leftarrow \cup_{i=1}^t F_i$;
if $n = 1$ **then**
  Isolate in (I, S) the real roots of the product of the polynomials in F;
  return I and S for $\mathcal{D}$;
else
  for $i = 1 \ldots t$ **do**
    Extract the set A_i of polynomials in ϕ_i ;
    Compute the set C_i of contents of the elements of A_i; Compute the set B_i, the finest squarefree basis for $\text{prim}(A_i)$;
  Set $C \leftarrow \cup_{i=1}^t C_i$, $\mathcal{B} \leftarrow (B_i)_{i=1}^t$ and $\mathcal{F} \leftarrow (F_i)_{i=1}^t$;
  Construct the projection set: $\mathfrak{P} \leftarrow C \cup P_{\mathcal{F}}(\mathcal{B})$;
  Attempt to construct a lower-dimensional CAD: $w', I', S' \leftarrow \texttt{CADW}(n-1, \mathfrak{P})$;
  if $w' = false$ **then**
    return FAIL ($\mathfrak{P}$ not well oriented);
  $I \leftarrow \emptyset$; $S \leftarrow \emptyset$;
  for *each cell* $c \in \mathcal{D}'$ **do**
    $L_c \leftarrow \{\}$;
    for $i = 1, \ldots t$ **do**
      if f_i *is nullified on* c **then**
        if $\dim(c) > 0$ **then**
          return FAIL (Φ not well oriented);
        else
          $L_c \leftarrow L_c \cup B_i$;
      else
        $L_c \leftarrow L_c \cup F_i$;
    Lift over c using L_c: construct cell indices and sample points for the stack over c of the polynomials in L_c, adding them to I and S;
  return I and S for $\mathcal{D}$;

We shall prove that, provided $\mathcal{A}$ and $\mathcal{E}$ satisfy the condition of well-orientedness given in Definition 6, the output

of Algorithm 1 is indeed a TTICAD for Φ. Note that this condition is specialised and new, introduced for this paper. Its requirement is due to both the use of `CADW` from [18] and the introduction of our new reduced projection operator.

We first recall the more general notion of well-orientedness from [18]. A set A of n-variate polynomials is said to be *well oriented* if whenever $n > 1$, every $f \in \text{prim}(A)$ is nullified by at most a finite number of points in $\mathbb{R}^{n-1}$, and (recursively) $P(A)$ is well-oriented. The Boolean output of `CADW` is false if the input set was not well-oriented in this sense. Now we define our new notion of well-orientedness for the set lists $\mathcal{A}$ and $\mathcal{E}$ defined above, and hence Φ.

DEFINITION 6. *We say $\mathcal{A}$ is* well oriented *with respect to $\mathcal{E}$ (and that Φ is* well oriented*) if whenever $n > 1$, every constraint polynomial f_i is nullified by at most a finite number of points in $\mathbb{R}^{n-1}$, and $P_{\mathcal{E}}(\mathcal{A})$ (hence $\mathfrak{P}$ in the algorithm) is well-oriented in the sense of [18].*

THEOREM 5. *The output of Algorithm 1 is as specified.*

PROOF. We must show that when Φ is well-oriented the output is a Truth Table Invariant CAD, (each ϕ_i has constant truth value in each cell of $\mathcal{D}$), and **FAIL** otherwise.

If the input was univariate then it is trivially well-oriented. The algorithm will construct a CAD $\mathcal{D}$ of $\mathbb{R}^1$ using the roots of the irreducible factors of the constraint polynomials (steps 5 to 6). At each 0-cell all the polynomials in each ϕ_i trivially have constant signs, and hence every ϕ_i has constant truth value. In each 1-cell no constraint polynomial has a root, so every ϕ_i has constant truth value $false$.

Now suppose $n > 1$. If $\mathfrak{P}$ is not well-oriented in the sense of [18] then `CADW` returns w' as false. In this case the input Φ is not well oriented in the sense of Definition 6 and Algorithm 1 correctly returns **FAIL**. Otherwise, $\mathfrak{P}$ is well-oriented and at step 13 we have $w' = true$. Further, I' and S' specify a CAD, $\mathcal{D}'$, order-invariant with respect to $\mathfrak{P}$. Let c, a submanifold of $\mathbb{R}^{n-1}$, be a cell of $\mathcal{D}'$.

Suppose first that the dimension of c is positive. If any constraint polynomial f_i vanishes identically on c then Φ is not well oriented in the sense of Definition 6 and the algorithm correctly returns **FAIL** at step 22. Otherwise, we know that Φ is certainly well-oriented. Since no constraint polynomial f_i vanishes then no element of the basis F vanishes identically on c either. Hence, by Theorem 3, applied with $\mathcal{A} = \mathcal{B}$ and $\mathcal{E} = \mathcal{F}$, each element of F is delineable on c, and the sections over c of the elements of F are pairwise disjoint. Thus the sections and sectors over c of the elements of F comprise a stack Σ over c. Furthermore, Theorem 3 assures us that, for each i, every element of $B_i \setminus F_i$ is sign-invariant in each section over c of every element of F_i.

Let $1 \leq i \leq t$. Consider first a section σ of the stack Σ. We shall show that ϕ_i has constant truth value in σ. Now the constraint polynomial f_i is a product of its content $\text{cont}(f_i)$ and some elements of the basis F_i. But $\text{cont}(f_i)$, an element of $\mathfrak{P}$, is sign-invariant in the whole cylinder $c \times \mathbb{R}$ which includes σ. Moreover all of the elements of F_i are sign-invariant in σ, as noted previously. Therefore f_i is sign-invariant in σ. If f_i is positive or negative in σ then ϕ_i has constant truth value $false$ in σ.

Suppose that $f_i = 0$ throughout σ. It follows that σ must be a section of some element of the basis F_i. Let $g \in A_i \setminus E_i$ be a non-constraint polynomial in A_i. Now, by the definition of B_i, we see g can be written as $g = \text{cont}(g)h_1^{p_1} \cdots h_k^{p_k}$ where $h_j \in B_i, p_j \in \mathbb{N}$. But $\text{cont}(g)$, in $\mathfrak{P}$, is sign-invariant in the whole cylinder $c \times \mathbb{R}$ including σ. Moreover each h_j is sign-invariant in σ, as noted previously. Hence g is sign-invariant in σ. (Note that in the case where g does not have main variable x_n then $g = \text{cont}(g)$ and the conclusion still holds). Since g was an arbitrary element of $A_i \setminus E_i$, it follows that all polynomials in A_i are sign-invariant in σ, and hence that ϕ_i has constant truth value in σ.

Next consider a sector σ of the stack Σ, and notice that at least one such sector exists. As observed above, $\text{cont}(f_i)$ is sign-invariant in c, and f_i does not vanish identically on c. Hence $\text{cont}(f_i)$ is non-zero throughout c. Moreover each element of the basis F_i is delineable on c. Hence the constraint polynomial f_i is nullified by no point of c. It follows from this that the algorithm does not return **FAIL** during the lifting phase. It follows also that $f_i \neq 0$ throughout σ. Therefore ϕ_i has constant truth value $false$ in σ.

It remains to consider the case in which the dimension of c is 0. In this case the roots of the polynomials in the lifting set L_c constructed by the algorithm determine a stack Σ over c. Each ϕ_i trivially has constant truth value in each section (0-cell) of this stack, and the same can routinely be shown for each sector (1-cell) of this stack. □

REMARK 6. *When the input to Algorithm 1 is a single QFF then it produces a CAD which is invariant with respect to the sole equational constraint. This may be shown using the results of [19] alone. However, we note that Algorithm 1 is actually more efficient in the lifting stage than the modified QEPCAD algorithm discussed in [19] since the lifting set excludes some non-equational constraint input polynomials.*

Algorithm 1 and Definition 6 have been kept conceptually simple to aid readability. However in practice the algorithm may sometimes be unnecessarily cautious. In [4], several cases where non-well oriented input can still lead to an order-invariant CAD are discussed. Similarly here, we can sometimes allow the nullification of an equational constraint on a positive dimensional cell.

LEMMA 7. *Let f_i be an equational constraint which vanishes identically on a cell $c \in \mathcal{D}'$ constructed during Algorithm 1. If all polynomials in $\text{ExclP}_{E_i}(A_i)$ are constant on c then any $g \in A_i \setminus E_i$ will be delineable over c.*

PROOF. Suppose first that A_i and E_i satisfy the simplifying conditions from Section 2.2. Rearranging (3) we see

$$P(A_i) = P_{E_i}(A_i) \cup \text{ExclP}_{E_i}(A_i).$$

However, given the conditions of the lemma, this is equivalent (after the removal of constants which do not affect CAD construction) to $P_{E_i}(A_i)$ on c. So here $P(A_i)$ is a subset of $P_{\mathcal{E}}(\mathcal{A})$ and we can conclude by Theorem 1 that all elements of A_i vanish identically on c or are delineable over c.

In the more general case we can still draw the same conclusion because $P(A_i) = C_i \cup P_{F_i}(B_i) \cup \text{ExclP}_{F_i}(B_i) \subseteq \mathfrak{P}$. □

Hence we can use Lemma 7 to safely extend step 24 to also apply in such cases. In particular, we can allow equational constraints f_i which do not have main variable x_n in such cases. We have included this in our implementation discussed in Section 3.3. In theory, we may be able to go further and allow step 24 to apply in cases where the polynomials in $\text{ExclP}_{E_i}(A_i)$ are not necessarily all constant, but have no real roots within the cell c. However, identifying such cases would require answering a separate quantifier elimination question, which may not be trivial.

3.2 TTICAD via the ResCAD Set

In Algorithm 1 the lifting stage (steps 16 to 27) varies according to whether an equational constraint is nullified. When this does not occur there is an alternative implementation of TTICAD which would be simpler to introduce into existing CAD algorithms. Define the *ResCAD Set* of Φ as

$$\mathcal{R}(\Phi) = E \cup \bigcup_{i=1}^{t} \{\text{res}_{x_n}(f,g) \mid f \in E_i, g \in A_i, g \notin E_i\}.$$

THEOREM 8. *Let $\mathcal{A} = (A_i)_{i=1}^t$ be a list of irreducible bases A_i and let $\mathcal{E} = (E_i)_{i=1}^t$ be a list of non-empty subsets $E_i \subseteq A_i$. For the McCallum projection operator P, [17] we have:*

$$P(\mathcal{R}(\Phi)) = P_{\mathcal{E}}(\mathcal{A}).$$

The proof is straightforward and so omitted here.

COROLLARY 9. *If no f_i is nullified by a point in $\mathbb{R}^{n-1}$ then inputting $\mathcal{R}(\Phi)$ into any algorithm which produces a sign-invariant CAD using McCallum's projection operator, will result in the TTICAD for Φ produced by Algorithm 1.*

Hence Corollary 9 gives us a simple way to compute TTICADs using existing CAD implementations, such as QEPCAD, but this cannot be applied as widely as Algorithm 1.

3.3 Implementation in Maple

There are various implementations of CAD available but none guarantee order-invariance, required for proving the validity of our TTICAD algorithm. Hence we needed to construct our own implementation to obtain experimental results. We built an implementation of McCallum projection, so that we could reproduce `CADW` and modified the existing stack generation commands in MAPLE from [8] so they could be used more widely. Together these allowed us to fully implement Algorithm 1. The CAD implementation grew to a MAPLE package `ProjectionCAD` which gathers together algorithms for producing CADs via projection and lifting to complement the existing CAD commands in MAPLE which use triangular decomposition, giving the same representation of sample points using regular chains. For further details (along with free access to the code) see [15].

3.4 Formulating a Problem for TTICAD

When formulating a problem for TTICAD there may be choices for the input, such as choosing which equational constraint to designate in a QFF when more than one is present. Other possibilities include choosing whether conjunctions of formulae should be split into separate QFFs. Usually it will be preferable to minimise the number of QFFs, but if for example a designated equational constraint has many intersections with another polynomial which could be ignored by using separate QFFs, then the cost of the extra polynomials in the projection set may be outweighed by the complexity of those removed. Hence it is worth taking care in how we formulate the TTICAD. A simple problem of the form

$$f_1 = 0 \wedge f_2 = 0 \wedge g_1 < 0 \wedge g_2 < 0$$

has six acceptable choices for the composition of Φ.

We have started exploring heuristics for choosing the best composition. The metric `sotd` (sum of total degrees) as defined in [14] may be used to approximate the complexity of polynomials. We first considered using $\texttt{sotd}(\mathfrak{P})$ and found that while it was fairly well correlated with the number of cells produced by Algorithm 1 it was not always fine enough to separate compositions leading to TTICADs with significantly different numbers of cells. Hence we prefer a stronger heuristic, $\texttt{sotd}(\mathfrak{P} \cup \overline{P}(\mathfrak{P})))$ where $\overline{P}$ is the complete set of projection polynomials obtained by repeatedly applying P.

For the problems in Section 4 we used the QFFs imposed by the disjunctions of formulae using this heuristic to choose which equational constraints are designated when there was a choice. For these problems the heuristic computation time was negligible compared to the overall time, but for larger problems this would not be the case. Work on heuristics is ongoing with a more detailed report available in [1].

4. EXPERIMENTAL RESULTS

4.1 Description of experiments

Our timings were obtained on a Linux desktop (3.1GHz Intel processor, 8.0Gb total memory) with MAPLE 16 (command line interface), MATHEMATICA 9 (graphical interface) and QEPCAD-B 1.69. For each experiment we produce a CAD and give the time taken and number of cells (cell count). The first is an obvious metric while the second is crucial for applications performing operations on each cell.

For QEPCAD the options `+N500000000` and `+L200000` were provided, the initialization included in the timings and explicit equational constraints declared when present with the product of those from the individual QFFs declared otherwise. In MATHEMATICA the output is not a CAD but a formula constructed from one [24], with the actual CAD not available to the user. Cell counts for the algorithms were provided by the author of the MATHEMATICA code.

TTICADs are calculated using our implementation described in Section 3.3, which is simple and not optimized. The results in this section are not presented to claim that our implementation is state of the art, but to demonstrate the power of the TTICAD theory over the the conventional theory, and how it can allow even a simple implementation to compete. Hence the cell counts are of most interest.

The time is measured to the nearest tenth of a second, with a time out (T/O) set at 5000 seconds. When **F** occurs it indicates failure due to a theoretical reason such as not well-oriented (in either sense). The occurrence of Err indicates an error in an internal subroutine of MAPLE's `RegularChains` package, used by `ProjectionCAD`. This error is not theoretical but a bug, beyond our control.

We considered examples originating from [7]. However these problems (and most others in the literature) involve conjunctions of conditions, chosen as such to make them amenable to existing technologies. These problems can be tackled using TTICAD, but they do not demonstrate its full strength. Hence we introduced new examples, denoted with a †, which are adapted from [7] to have disjuncted QFFs.

Two examples came from the application of branch cut analysis for simplification. These problems require a decomposition according to branch cuts of the form $f = 0 \wedge g < 0$, and then go on to test the validity of a simplification on each cell, [21, etc.]. We need to consider the disjunction of the branch cuts making such problems suitable for Algorithm 1. We included a key example from Kahan [16], along with the problem induced by considering the validity of the double angle formulae for arcsin. Finally we considered the worked example from Section 1.3 and its generalisation to three dimensions. Note that A and B following the problem name

indicate different variable orderings. Full details for all examples can all be found in the CAD repository [25] available freely online at `http://opus.bath.ac.uk/29503`.

4.2 Results

We present our results in Table 1. For each problem we give the name used in the repository, n the number of variables, d the maximum degree of polynomials involved and t the number of QFFs used for TTICAD. We then give the time taken and number of cells produced by each algorithm.

We first compare our TTICAD implementation with the sign-invariant CAD generated using `ProjectionCAD` with McCallum's projection operator [15]. Since these use the same architecture the comparison makes clear the benefits of the TTICAD theory. The experiments confirm the fact that the cell count for TTICAD will always be less than or equal to that of a sign-invariant CAD produced using the same implementation. Ellipse† A is not well-oriented in the sense of [18], and so both methods return **FAIL**. Solotareff† A and B are well-oriented in this sense but not in the stronger sense of Definition 6 and hence TTICAD fails while the full sign-invariant CADs can be produced. The only example with equal cell counts is Collision† A in which the non-equational constraints were so simple that the projection polynomials were unchanged. Examining the results for the worked example and its generalisation we start to see the true power of TTICAD. In 3D Example A we see a 759-fold reduction in time and a 50-fold reduction in cell count.

We next compare our implementation of TTICAD with the state of the art in CAD: QEPCAD [3], MAPLE [8] and MATHEMATICA [23, 24]. MATHEMATICA is the quickest, however TTICAD often produces fewer cells. We note that MATHEMATICA's algorithm uses powerful heuristics and so actually used Gröbner bases on the first two problems, causing the cell counts to be so low. When all implementations succeed TTICAD usually produces far fewer cells than QEPCAD or MAPLE, especially impressive given QEPCAD is producing partial CADs for the quantified problems, while TTICAD is only working with the polynomials involved. For Collision† A the TTICAD theory offers no benefit allowing the better optimized alternatives to have a lower cell count.

Reasons for the TTICAD implementation struggling to compete on speed in general are that the MATHEMATICA and QEPCAD algorithms are largely implemented directly in `C`, have had far more optimization, and in the case of MATHEMATICA use validated numerics for lifting [23]. However, the strong performance in cell counts is very encouraging, both due its importance for applications where CAD is part of a wider algorithm (such as branch cut analysis) and for the potential if TTICAD theory were implemented elsewhere.

5. CONCLUSIONS

We have defined Truth Table Invariant CADs, which can be more closely aligned to the needs of problems than traditional sign-invariant CADs. Theorem 3 extended the theory of equational constraints allowing us to develop Algorithm 1 to construct TTICADs efficiently for a large range of problems. The algorithm has been implemented in MAPLE giving promising experimental results. TTICADs in general have less cells than full sign-invariant CADs using the same implementation and we showed that this allows even a simple implementation of TTICAD to compete with the state of the art CAD implementations. It is anticipated that future implementations of TTICAD could be far better optimized leading to lower times for the same cell counts. We also note that the benefits of TTICAD increase with the number of QFFs in a problem and so larger problems may be susceptible to TTICAD when other approaches fail.

We hope that these results inspire other implementations of TTICAD, with Corollary 9 showing a particularly easy way to adapt existing CAD implementations.

5.1 Future Work

There is scope for optimizing the algorithm and extending it to allow less restrictive input. Lemma 7 gives one extension that is included in our implementation while other possibilities include removing some of the caution implied by well-orientedness, analogous to [4]. Also, work developing heuristics for composing the input is underway in [1].

Of course, the implementation of TTICAD used here could be improved in many ways, but perhaps more desirable would be for TTICAD to be incorporated into existing state of the art CAD implementations. In particular, we would like to use the existing MAPLE CAD commands [8] but this requires first understanding when they give order-invariance, a key question currently under consideration. We see several possibilities for the theoretical development of TTICAD:

- Can we apply the theory recursively instead of only at the top level? For example by widening the projection operator to conclude order-invariance, as in [20].
- Can we apply TTICAD to forms of QFF other than "one equality and other items"? For example, can we generalise the theory of bi-equational constraints?
- Can we make use of the ideas behind partial CAD to avoid unnecessary lifting once the truth value of a QFF on a cell is determined?
- Can anything be done when Φ is not well oriented?
- Can we implement the lifting algorithm in parallel?

Acknowledgements

We are grateful to A. Strzeboński for assistance in performing the Mathematica tests and to the anonymous referees for useful comments. We also thank the rest of the Triangular Sets seminar at Bath (A. Locatelli, G. Sankaran and N. Vorobjov) for their input, and the team at Western University (C. Chen, M. Moreno Maza, R. Xiao and Y. Xie) for access to their MAPLE code and helpful discussions. This work was supported by the EPSRC grant: EP/J003247/1.

6. REFERENCES

[1] R. Bradford, J.H. Davenport, M. England, and D. Wilson. Optimising Problem Formulation for Cylindrical Algebraic Decomposition. In Press: *Proc. CICM '13*. Preprint: http://opus.bath.ac.uk/34373.

[2] C.W. Brown. Simplification of truth-invariant cylindrical algebraic decompositions. *Proc. ISSAC '98*, pages 295–301, 1998.

[3] C.W. Brown. QEPCAD B: A program for computing with semi-algebraic sets using CADs. *ACM SIGSAM Bulletin*, 37:4, pages 97–108, 2003.

[4] C.W. Brown. The McCallum projection, lifting, and order-invariance. Technical report, U.S. Naval Academy, Computer Science Department, 2005.

[5] C.W. Brown and J.H. Davenport. The Complexity of Quantifier Elimination and Cylindrical Algebraic Decomposition. *Proc. ISSAC '07*, pages 54–60, 2007.

Table 1: Comparing TTICAD to the full CAD built with the same architecture and other CAD algorithms.

Problem				Full-CAD		TTICAD		QEPCAD		MAPLE		MATHEMATICA	
Name	n	d	t	Time	Cells	Time	Cells	Time	Cells	Time	Cells	Time	Cells
Intersection A	3	2	1	360.1	3707	1.7	269	4.5	825	—	Err	0.0	3
Intersection B	3	2	1	332.2	2985	1.5	303	4.5	803	50.2	2795	0.0	3
Random A	3	3	1	268.5	2093	4.5	435	4.6	1667	23.0	1267	0.1	657
Random B	3	3	1	442.7	4097	8.1	711	5.4	2857	48.1	1517	0.0	191
Intersection† A	3	2	2	360.1	3707	68.7	575	4.8	3723	—	Err	0.1	601
Intersection† B	3	2	2	332.2	2985	70.0	601	4.7	3001	50.2	2795	0.1	549
Random† A	3	3	2	268.5	2093	223.4	663	4.6	2101	23.0	1267	0.2	808
Random† B	3	3	2	442.7	4097	268.4	1075	142.4	4105	48.1	1517	0.2	1156
Ellipse† A	5	4	2	—	**F**	—	**F**	291.6	500609	1940.1	81193	11.2	80111
Ellipse† B	5	4	2	T/O	—	T/O	—	T/O	—	T/O	—	2911.2	16603131
Solotareff† A	4	3	2	677.6	54037	46.1	**F**	4.9	20307	1014.2	54037	0.1	260
Solotareff† B	4	3	2	2009.2	154527	123.8	**F**	6.3	87469	2951.6	154527	0.1	762
Collision† A	4	4	2	264.6	8387	267.7	8387	5.0	7813	376.4	7895	3.6	7171
Collision† B	4	4	2	—	Err	—	Err	T/O	—	T/O	—	591.5	1234601
Kahan A	2	4	7	10.7	409	0.3	55	4.8	261	15.2	409	0.0	72
Kahan B	2	4	7	87.9	1143	0.3	39	4.8	1143	154.9	1143	0.1	278
Arcsin A	2	4	4	2.5	225	0.3	57	4.6	225	3.3	225	0.0	175
Arcsin B	2	4	4	6.5	393	0.2	25	4.5	393	7.8	393	0.0	79
2D Example A	2	2	2	5.7	317	1.2	105	4.7	249	6.3	317	0.0	24
2D Example B	2	2	2	6.1	377	1.5	153	4.5	329	7.2	377	0.0	175
3D Example A	3	3	2	3795.8	5453	5.0	109	5.3	739	—	Err	0.1	44
3D Example B	3	3	2	3404.7	6413	5.8	153	5.7	1009	—	Err	0.1	135

[6] C.W. Brown and S. McCallum. On using bi-equational constraints in CAD construction. *Proc. ISSAC '05*, pages 76–83, 2005.

[7] B. Buchberger and H. Hong. Speeding-up Quantifier Elimination by Gröbner Bases. RISC Technical Report 91-06, 1991.

[8] C. Chen, M. Moreno Maza, B. Xia, and L. Yang. Computing Cylindrical Algebraic Decomposition via Triangular Decomposition. *Proc. ISSAC '09*, pages 95–102, 2009.

[9] G.E. Collins. Quantifier Elimination for Real Closed Fields by Cylindrical Algebraic Decomposition. *Proc. 2nd. GI Conference Automata Theory & Formal Languages*, pages 134–183, 1975.

[10] G.E. Collins. Quantifier elimination by cylindrical algebraic decomposition — twenty years of progess. *Quantifier Elimination and Cylindrical Algebraic Decomposition*, pages 8–23, 1998.

[11] G.E. Collins and H. Hong. Partial Cylindrical Algebraic Decomposition for Quantifier Elimination. *J. Symbolic Comp.*, 12:3, pages 299–328, 1991.

[12] J.H. Davenport, R. Bradford, M. England, and D. Wilson. Program verification in the presence of complex numbers, functions with branch cuts etc. *Proc. SYNASC '12*, pages 83–88, 2012.

[13] J.H. Davenport and J. Heintz. Real Quantifier Elimination is Doubly Exponential. *J. Symbolic Comp.*, 5:1-2, pages 29–35, 1988.

[14] A. Dolzmann, A. Seidl, and Th. Sturm. Efficient Projection Orders for CAD. *Proc. ISSAC '04*, pages 111–118, 2004.

[15] M. England. An implementation of CAD utilising McCallum projection in Maple. *University of Bath, Dept. Computer Science Technical Report Series*, 2013:2. http://opus.bath.ac.uk/33180, 2013.

[16] W. Kahan. Branch Cuts for Complex Elementary Functions. A. Iserles and M.J.D. Powell, editorss, *Proc. The State of the Art in Numerical Analysis*, pages 165–211, 1987.

[17] S. McCallum. An Improved Projection Operation for Cylindrical Algebraic Decomposition of Three-dimensional Space. *J. Symbolic Comp.*, 5:1-2, pages 141–161, 1988.

[18] S. McCallum. An Improved Projection Operation for Cylindrical Algebraic Decomposition. *Quantifier Elimination and Cylindrical Algebraic Decomposition*, pages 242–268, 1998.

[19] S. McCallum. On Projection in CAD-Based Quantifier Elimination with Equational Constraints. *Proc. ISSAC '99*, pages 145–149, 1999.

[20] S. McCallum. On Propagation of Equational Constraints in CAD-Based Quantifier Elimination. *Proc. ISSAC '01*, pages 223–230, 2001.

[21] N. Phisanbut, R.J. Bradford, and J.H. Davenport. Geometry of Branch Cuts. *Communications in Computer Algebra*, 44:132–135, 2010.

[22] J.T. Schwartz and M. Sharir. On the "Piano-Movers" Problem: II. General Techniques for Computing Topological Properties of Real Algebraic Manifolds. *Adv. Appl. Math.*, 4:298–351, 1983.

[23] A. Strzeboński. Cylindrical algebraic decomposition using validated numerics. *Journal of Symbolic Computation*, 41:9, pages 1021–1038, 2006.

[24] A. Strzeboński. Computation with semialgebraic sets represented by cylindrical algebraic formulas. *Proc. ISSAC '10*, pages 61–68. ACM, 2010.

[25] D.J. Wilson, R.J. Bradford, and J.H. Davenport. A Repository for CAD Examples. *ACM Communications in Computer Algebra*, 46:3 pages 67–69, 2012.