J. Symbolic Computation (1988) **5**, 189–212

A Cluster-Based Cylindrical Algebraic Decomposition Algorithm*

DENNIS S. ARNON

Xerox PARC, 3333 Coyote Hill Road, Palo Alto, California 94304, U.S.A.

(Received 2 April 1985, and in revised form 15 November 1987)

Let $A \subset \mathbf{Z}[x_1, \ldots, x_r]$ be a finite set. An *A-invariant cylindrical algebraic decomposition (cad)* is a certain partition of r-dimensional euclidean space E^r into semi-algebraic cells such that the value of each $A_i \in A$ has constant sign (positive, negative, or zero) throughout each cell. Two cells are *adjacent* if their union is connected. Recently a number of methods have been given for augmenting Collins' cad construction algorithm (1975), so that in addition to specifying the cells that comprise a cad, it identifies the pairs of adjacent cells. Assuming the availability of such an adjacency algorithm, in this paper we give a modified cad construction algorithm based on the utilization of clusters of cells in a cad (a *cluster* is a collection of cells whose union is connected). Preliminary observations indicate that the new algorithm can be significantly more efficient in some cases than the original, although in other examples it is somewhat less efficient.

1 Introduction

Recently a number of methods have been given for augmenting the cad construction algorithm (Collins, 1975), so that in addition to specifying the cells that comprise a cad, it identifies the pairs of adjacent cells (see e.g. Arnon *et al.*, 1988, Prill, 1986, Kozen & Yap 1985, Schwartz & Sharir 1983). A *cluster* of cells in a cad is a collection of cells whose union is connected. Assuming the availability of an adjacency algorithm, in this paper we give a modified cad construction algorithm based on the utilization of clusters. The key idea is that, as a cad of E^{i-1} is extended to a cad of E^i, certain (possibly expensive) computations are performed only once for each cluster, rather than once for each cell as in the original algorithm. Offsetting this saving is the extra cost of adjacency computation. Preliminary observations indicate that the new algorithm can be significantly more efficient in some cases than the original, although in other examples it is somewhat less efficient. In this paper we give both a general framework for cluster-based cad construction, within which any available adjacency algorithm can be used, and a specific cluster-based cad algorithm that uses the 2-space and 3-space adjacency algorithms of Arnon *et al.* (1984b, 1988). The specific algorithm we give has the following properties: (1) it requires no coordinate changes, and (2) in any cad of E^1, E^2,

*This work was supported by the National Science Foundation (Grant MCS-8009357 to the University of Wisconsin-Madison), the Purdue Research Foundation, and the Xerox Corporation. This paper was typeset at Xerox PARC using TEX in the Cedar environment.

0747–1717/88/010189+24 $03.00/0

or E^3 that it builds, the boundary of each cell is a (disjoint) union of lower-dimensional cells. The particular clusters that occur in cluster-based cad construction are of mathematical interest in their own right. For example, if A consists of a single element F, then the (unions of the) r-space clusters are typically the connected components of the hypersurface $F = 0$ and its complement.

In this Introduction we sketch the broad outlines of the clustering strategy for cad construction, give an outline of the paper, and review prior related work.

1.1 Cad graphs and clusters

Let us begin our discussion of clustering by recalling terminology from Arnon *et al.* (1984a, 1984b, 1988). We say that a connected subset of E^r is a *region*. If $A = (A_1, ..., A_n)$ is a subset of $I_r = \mathbf{Z}[x_1, \ldots, x_r]$, if R is an A-invariant region in E^r (i.e. the value of each $A_i \in A$ has constant sign (-1, 0, or $+1$) throughout R), and if σ_i is the sign of A_i on R, then we say that the ordered n-tuple $\sigma = (\sigma_1, ..., \sigma_n)$ is the *signature* of R with respect to A (and also, the signature of A on R). A *cell triple* for a cell c of an A-invariant cad is a triple (I, σ, S), where I is the cell index of c (cell indices are defined in Section 4 of Arnon *et al.*, 1984a), σ is the signature of the cell (with respect to the set A of input polynomials), and S is a sample point for c. We temporarily proceed as though sample points are represented as in Arnon *et al.* (1984a, 1988); we will have more to say about their representation later.

Given $A \subset I_r$, a *graph representation* for an A-invariant cad D of E^r, or *cad graph*, is a quintuple $G = (A, B, V, E, G')$, defined as follows. B is a basis (as defined in Arnon *et al.*, 1988) for $prim(A)$, such that D is a basis-determined cad with basis B. (Recall that $prim(A)$ the set of primitive parts of those elements of A which have positive degree). V is a set of cell triples for the cells that comprise D. E is a set of unordered pairs of (distinct) elements of V, obeying the following condition: if (c_1, c_2) is an element of E, then cells c_1 and c_2 of the cad D are adjacent (thus (V, E) is a certain undirected graph). For any given pair of cells c_1 and c_2 of D, the converse may or may not hold. If for every pair of cells c_1 and c_2 of D the converse does hold, i.e. $(c_1, c_2) \in E$ if and only if c_1 and c_2 are adjacent in D, then we say that G is a *full* graph for D; otherwise, G is *partial*. The reader will notice a certain abuse of notation here: we freely identify a cell c with the triple that represents it. If $r > 1$, then G' is a graph representation for the cad D' of E^{r-1} induced by D, and $G' = \emptyset$ when $r = 1$. Typically the cad graph representations we work with are partial. In case G is full, the undirected graph (V, E) has been called the *connectivity graph* of D (Schwartz & Sharir, 1983, p. 320). We assume the availability of standard graph algorithms, e.g. depth-first search for connected components; see e.g. Aho *et al.* (1974).

It is appropriate to check that a graph representation for a cad supplies the information about that cad called for at the beginning of Section 4 of Arnon *et al.* (1984a). It was stated there that a description of a cad must inform one of the number of cells in the cad, how they are arranged into stacks, and the signature of each cell with respect to the set of input polynomials. Obviously a cad graph gives one the number of cells and each cell's signature. As detailed in Arnon *et al.* (1984a), the indices of the cells comprising a cad tell one how those cells are arranged into stacks.

Given $A \subset I_r$, let G denote a full graph for an A-invariant cad D. It is easy to show that the vertices of any connected subgraph of G correspond to a collection of cells of D whose union is a region in E^r. Turning this around, we say that a collection C of

cells of D is a *cluster* (of D) if the subgraph of G induced by C is connected. Clearly C is a cluster if and only if the union of C is a region. The *dimension* of a cluster is the dimension of the largest cell in it. A partition of (the set of cells of) a cad D into clusters is called a *clustering* of D. Obviously any D can be clustered in many ways.

Assume now that G is either partial or full, and suppose given an equivalence relation R on the cells of D. Then R induces a clustering of D, which can be made explicit by computing the connected components of G subject to the constraint that we only "notice" an edge during the computation if the cells it joins belong to R. In this paper, we are exclusively interested in one particular equivalence relation, namely the relation to which a pair of cells belongs if and only if the two cells have the same signature (with respect to the set A of input polynomials). We call this the *sign-invariance* relation. (We will henceforth be using the term "sign-invariant" quite often in place of "A-invariant", to denote the condition that "each input polynomial is sign-invariant", without mentioning the particular set A of input polynomials). We call a clustering induced by the sign-invariance relation in a graph representation for a cad D a *sign-invariant clustering* of D, and the clusters which comprise it *sign-invariant clusters.*

Given two clusterings Γ_1 and Γ_2 of a cad D, we say that Γ_1 is *finer* than Γ_2, if each cluster of Γ_1 is a subset of some cluster of Γ_2. Equivalently, we say that Γ_1 is a *refinement* of Γ_2, and that Γ_2 is *coarser* than Γ_1. We say that a sign-invariant clustering of D is *maximal* if it is the coarsest possible sign-invariant clustering of D; its elements are then *maximal sign-invariant clusters.* Given A, we call the maximal connected A-invariant subsets of E^r the *A-components* of E^r, or in general the *sign-invariant components* of E^r (with respect to this A, of course). Note that this last definition is independent of any particular cad of E^r. Clearly a sign-invariant clustering of D is maximal if and only if the union of each of its clusters is a sign-invariant component of E^r. If G is a full graph for D, then clearly the sign-invariant connected components of G correspond to the maximal sign-invariant clusters of D, however if G is partial, this need not be the case.

If G is partial, then an equivalence relation on the cells of D still induces a clustering of D when (just as above) we compute connected components in the cad graph under the constraint that we only notice edges between equivalent cells. Such a clustering is in general finer than the clustering we get with the same relation applied to a full G, since the edges in a partial G are a subset of the edges in a full G. This observation is important, because in general we will build clusterings using partial graphs, and we will be interested in how closely these clusterings correspond to the clusterings that the same equivalence relation induces in a full graph.

Let us now look at some examples of the notions we have introduced. Consider the sample cad D from Section 5 of Arnon *et al.* (1984b), which we show in Fig. 1. Fig. 2 shows a full graph for D. The figure uses the convention that 0-cells are indicated as solid vertices, 1-cells as half-filled vertices, and 2-cells as unfilled vertices. Edges satisfiying the sign-invariance relation are shown as solid lines, and those not are shown as dotted lines. We see that there are 15 maximal sign-invariant clusters. Fig. 3 shows a partial graph for D. Again, edges satisfiying the sign-invariance relation are drawn as solid lines, and those not satisfiying it are drawn as dotted lines. For this graph, we get 19 sign-invariant clusters, many of which are not maximal.

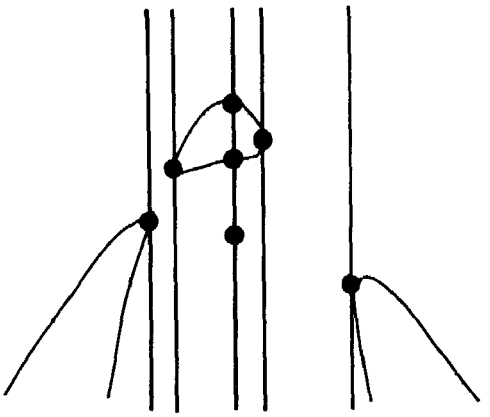

Figure 1: Sample Cad.

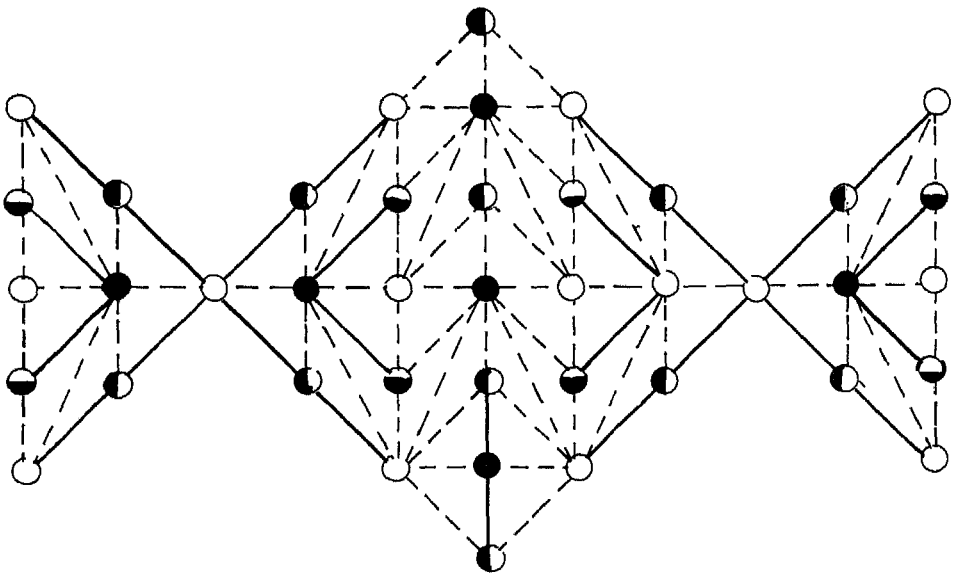

Figure 2: Full graph.

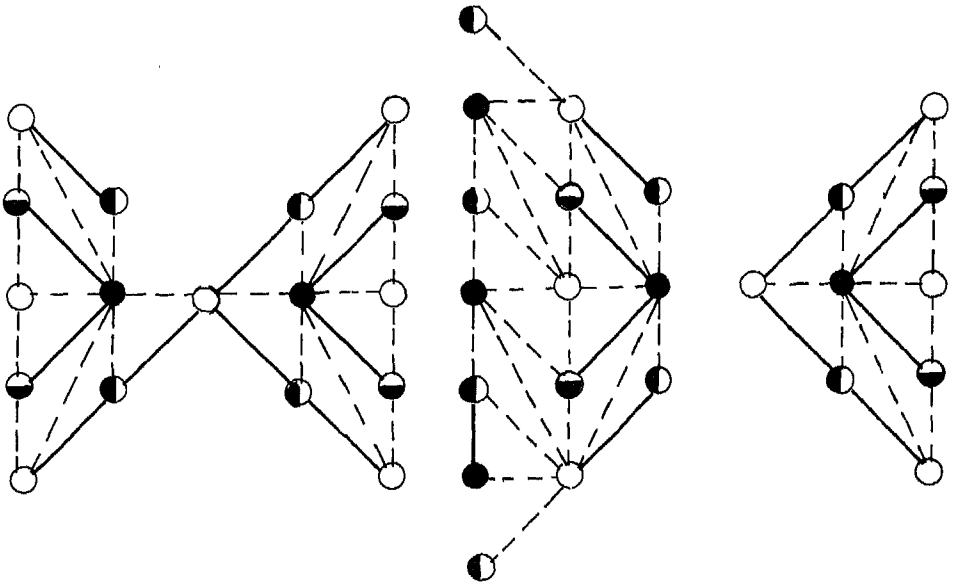

Figure 3: Partial graph.

1.2 Cluster-based cad construction

Let us now describe the basic idea of cluster-based cad construction. Essentially what we do is make the extension of a cad of E^{i-1} to E^i more efficient, by building stacks over sign-invariant clusters in E^{i-1}, rather than over individual cells in E^{i-1}. This general strategy requires the availability of adjacency algorithms, but does not require the use of any particular adjacency algorithm. We now explain in detail the formal basis for the strategy.

Assume given $A \subset I_r$. In Section 3 of Arnon *et al.* (1984a), a map $PROJ$, which takes a subset of I_r to a subset of I_{r-1}, is defined, and it is proved (Theorem 3.4) that over any $PROJ(A)$-invariant region in E^{r-1}, there exists an A-invariant stack. In applying this Theorem 3.4 to extend a cad of E^{r-1} to a cad of E^r in algorithm CAD of Arnon *et al.* (1984a), the $PROJ(A)$-invariant regions in E^{r-1} are the cells of the induced cad of E^{r-1}. However, given an arbitrary $PROJ(A)$-invariant decomposition $\hat{D}$ of E^{r-1}, Theorem 3.4 tells us that if we have a sample point for each region of $\hat{D}$, then we can extend it to a decomposition D^* of E^r consisting of the stacks over regions of $\hat{D}$, by exactly the steps used in CAD for extension over a single cell. Note that D^* is not necessarily cylindrical in the sense of Arnon *et al.* (1984a), *i.e.* it may not be the case that $\hat{D}$ consists of stacks over the regions of some decomposition of E^{r-2}. However, it is the case that if $\hat{D}$ is algebraic, *i.e.* its regions are semi-algebraic sets, then so is D^*.

Suppose now that for some $A \subset I_r$, we have (a graph for) a $PROJ(A)$-invariant cad D' of E^{r-1}, and a clustering of it into $PROJ(A)$-invariant clusters. Then forming the regions we get by taking the union of each cluster, we get a $PROJ(A)$-invariant, decomposition $\hat{D}$ of E^{r-1}. As above, let us extend $\hat{D}$ to a decomposition D^* of E^r by building stacks over $\hat{D}$'s regions, and let D denote the (A-invariant) cad of E^r that we

get by extending D'. Then it is not hard to see that each element of D^* is the union of certain elements of D. In particular, if C is a cluster of D', and if R is the union of C, then for any $i \geq 1$, the i^{th} element of the stack over R (this stack is part of D^*) is the union of the i^{th} elements of the stacks over the cells of C (these stacks are each part of D). Furthermore, if cells c_1 and c_2 of C are adjacent elements of D', then for any $i \geq 1$, the i^{th} element of the stack over c_1, and the i^{th} element of the stack over c_2, are adjacent elements of D. We call such clusters and adjacencies in E^r *induced clusters* and *induced adjacencies*, because they are "induced" in a cad D of E^r by a cluster or adjacency in the cad D' of E^{r-1}.

Given D', a $PROJ(A)$-invariant clustering of D', and a sample point for (one cell in) each cluster, our observations above imply that we can build D as follows. For each cluster, we determine a stack over its union R using its sample point, as just discussed. Having determined the number of elements, i.e. sections and sectors, of this stack, and assuming that we have determined the signature of each element of the stack with respect to A, we next look to see which cells of D' comprise C (i.e. what their cell indices are), and we know immediately (i.e. we can write down the cell indices and signatures for) the cells of D which comprise each element of the stack over R. When we have processed all clusters of D' in this way, we will have compiled the indices and signatures of all cells of D. Furthermore, for each cluster C of D', each adjacency $\{c, d\}$ of elements of C induces an adjacency between the i^{th} elements of the stacks over c and d. Clearly a graph for D which contains exactly these induced adjacencies gives rise to an induced clustering of D (into induced clusters).

Clearly the induced adjacencies of D are sign-invariant adjacencies. They are likely to be only a proper subset of the set of all of D's sign-invariant adjacencies, however. In particular, if we do a sign-invariant components computation in the graph for D in the form that it has after the steps we have described, the sign-invariant clusters we construct are likely not maximal. To get the most benefit from the use of clusters, we would like to have the largest possible sign-invariant clusters. Hence, the next step of our general strategy is to build larger sign-invariant clusters than those given to us by the induced adjacencies. As one might guess, we do this by computing further adjacencies in E^r using the adjacency algorithms that we assume are available to us.

Let us consider a simple example of these ideas. Let $A = \{u^2 + z^2 + y^2 + x^2 - 1\}$, i.e. A consists of the polynomial which defines the (three-dimensional) unit sphere in 4-space. We have $PROJ(A) = \{z^2 + y^2 + x^2 - 1\}$, $PROJ^2(A) = \{y^2 + x^2 - 1\}$, and $PROJ^3(A) = \{x^2 - 1\}$. The cad of 1-space clearly has five cells; recall from Section 4 of Arnon *et al.* (1984a) that we write the indices for these cells as (1), (2), (3), (4), (5). The maximal sign-invariant clusters for this cad of E^1 are just the five singleton sets. In 2-space, we have 13 cells, which can be partitioned into three maximal sign-invariant clusters: the unit circle (consisting of four cells, with indices (2,2), (3,2), (3,4), and (4,2)), its interior (consisting of one cell, with index (3,3)), and its exterior (consisting of eight cells, with indices (1,1), (2,1), (2,3), (3,1), (3,5), (4,1), (4,3), and (5,1)). The discussion of the previous paragraphs tells us that to determine the cad of 3-space, it suffices to have a sample point for each of these three 2-space clusters. When we extend over the unit circle, for example, we get a stack in E^3 consisting of three elements (i.e. two sectors and one sections), that corresponds to four stacks in the cad of 3-space that each have three elements. The adjacencies among the cells in E^2 that comprise the unit circle induce certain adjacencies (and three clusters) among the cells of these four stacks in 3-space. Similarly, extending into E^3 over the interior of the unit circle in E^2, we get

a stack with five elements, and extending over the exterior of the circle, a stack in E^3 with one element. The latter stack corresponds to eight one-element stacks, with the obvious induced adjacencies and a single induced cluster, of the cad of 3-space. Clearly, the induced clusters in 3-space that we have described are not maximal sign-invariant clusters. Using a 3-space adjacency algorithm, we can compute additional adjacencies among the 3-space cells that enable us to obtain the three maximal sign-invariant clusters that there clearly are in 3-space (which correspond to the unit sphere, its interior, and its exterior). We can then build stacks in 4-space over these three 3-space clusters to determine a sign-invariant cad of 4-space.

Altogether the three steps of the cluster-based cad algorithm are: (1) If $r > 1$, call the algorithm recursively to build a graph for the induced cad of $r-1$ space, (2) If $r > 1$, extend, over the maximal sign-invariant clusters of the induced cad, to a graph for the cad of r-space, or if $r = 1$, build a graph directly, (3) Construct additional adjacencies in r-space. The simplest, trivial, case of cluster-based cad construction is the "original" cad algorithm, i.e. no adjacency computation at all, which means that we generally have just singleton clusters in the cad's of E^1, E^2, ... that we build.

1.3 Outline of the paper and prior work

Sections 2-4 fill in the details of the cluster-based cad construction strategy, by partitioning it into algorithms of four kinds: basis (Section 2), projection (Section 2), extension (Section 3), and adjacency (Section 4). Section 3 begins by defining several possible representations for sample points in cad graphs. This is fundamental material for this paper: careful management of sample point representations is an important reason why cluster-based cad construction is more efficient than previous cad algorithms in those cases that it is. Section 4 presents the particular adjacency algorithms for E^2 and E^3 that we currently use; these rely on certain adjacency subalgorithms from Arnon *et al.* (1984b) and Arnon *et al.* (1988). Section 5 presents a main algorithm CLCAD for cluster-based cad construction, which has procedure parameters for the four key subalgorithms. Also in Section 5 we specify the values of these procedure parameters that we use in our current implementation of CLCAD. Section 6 reports the comparative performance of algorithms CLCAD and CAD on a number of examples.

The work we report in this paper was done between 1979 and 1981. The use of adjacencies and clusters in cad construction was presaged by Arnon (1979), where it was shown that incidence of cells is decidable. A first version of CLCAD was presented, and some examples of its use and comparative performance with CAD given, in Arnon (1981). Applications of cluster-based cad construction can be found in Arnon & McCallum (1988), and Arnon (1988).

Defining formula construction is an important part of the cad algorithm, especially for applications to quantifier elimination (see e.g. Arnon & Mignotte, 1988). Constructing a defining formula for each cell of a cad is easily accomplished in cluster-based cad construction, by constructing such formulas for certain cells, and then inferring formulas for the remaining cells by much the same inference process as used for induced clusters and adjacencies in Section 1.2. See Arnon (1981) or Collins (1975) for details of Collins' original algorithm for cell defining formula construction.

As mentioned above, in the present paper we only make use of one equivalence relation of cells, namely the sign-invariance relation. The order-invariance relation of McCallum (1988) is another equivalence relation of cells in a cad whose use for cluster-

based cad construction is attractive.

2 Basis, Projection, and Base steps

In this Section we discuss the first few steps of the cluster-based cad construction algorithm. We have relatively little to say about them. The reader may wish either to look ahead to algorithm CLCAD in Section 5, or skip this Section for the moment.

In general, it doesn't matter what type of basis (e.g. coarsest or finest squarefree basis) our basis procedure computes (see Arnon *et. al.*, 1988, and Collins, 1975, for basis-related definitions). The actual projection operator we currently use is determined by the adjacency algorithms of Arnon *et al.* (1984b, 1988) that we use (cf. Section 4). We want to build the same cad's as these algorithms do. Hence rather than use $PROJ(A)$ as a projection operator, as we did in Section 1, we henceforth assume that we have computed a basis B for $prim(A)$, and that we use $PROJ(B) \cup cont(A)$ as our projection operator ($cont(A)$ is the set of non-zero non-unit contents of elements of A; see Arnon *et al.*, 1988, for futher discussion). For $r \leq 3$, we could use McCallum (1988) projection instead; it would then be necessary that our basis procedure compute a finest squarefree basis. The projection operator would then be the P operator as defined in McCallum (1988). The resulting cad's of E^r, $1 \leq r \leq 3$, would still have the boundary property, i.e. the boundary of each cell would be a (disjoint) union of lower-dimensional cells, and if $r > 1$, then the induced cad of E^{r-1} would also have the boundary property.

The base step of our cad algorithm, i.e. the algorithm for construction of cad's of E^1, is essentially that of Arnon *et al.* (1984a). It is easy to make the graph for the cad of E^1 full, since we trivially know what its adjacencies are. The reader may consult algorithm CLCAD in Section 5 for details of the base step.

3 Extension step

Our task in this section is to develop the method (algorithm *ExtendCadClusters* of Fig. 6) that we use for the extension step of cluster-based cad construction. We begin by considering the issue of sample point representation. Assume throughout this section that our cad input polynomials have $r \geq 1$ variables.

So far we have assumed that cell sample points are represented as in Arnon *et al.* (1984a, 1984b, 1988). In fact, cell sample points in the cluster-based cad algorithm may have one of three representations: (a) *null* (no information), or (b) *extended*, consisting of a real algebraic number α, an $r-1$ tuple of elements of $Q(\alpha)$, a nonzero squarefree polynomial $g(x) \in Q(\alpha)[x]$, and an isolating interval for a (real) root of $g(x)$ (this root is the r^{th} coordinate of the sample point), or (c) *primitive*, consisting of a real algebraic number α (the primitive element) and an r-tuple of elements of $Q(\alpha)$.

In fact, this extended representation is present in passing in the extension step of the cad algorithms in Arnon *et al.* (1984a, 1984b, 1988), although ultimately all cell sample points become primitive in these algorithms. To be specific, when we have a sample point for the base of a stack, and we isolate the real roots of a squarefree univariate algebraic polynomial to determine the sections of the stack, the base sample point, the algebraic polynomial, and each isolating interval for one of its roots give us an extended sample point representation for a section of the stack. As described in Section

5 of Arnon *et al.* (1984a), we can use the NORMAL and SIMPLE algorithms of Loos (1982) to convert an extended representation to a primitive one. This conversion process has been observed to often be expensive, and avoiding it whenever possible is a major goal of the cluster-based cad algorithm. Working with extended rather than primitive representations whenever possible is one step towards that goal; another such step is to make do with null sample points, whenever possible, which we also will do.

As we have noted, however, it is a required of a cad algorithm to construct input polynomial signatures for each of its cells. Previous cad algorithms (such as those in Arnon *et al.*, 1984a, 1984b, 1988) have done so by evaluating the input polynomials at primitive cell sample points. We now show that it is possible to compute the signature (with respect to the input polynomials) of a cell in a basis-determined cad of E^r given an extended representation for the cell's sample point. In fact, the method we give could be used in the original as well as the cluster-based cad algorithm, to avoid extended-to-primitive conversions of section sample points in dimension r, i.e. the highest dimension.

We proceed in two steps. First, we show how to get the signatures of cells in E^r with respect to the basis polynomials from extended representations for their sample points. Second, we infer input polynomial signatures from these basis signatures plus signatures for the contents of the input polynomials. Here is a sketch of the first step. Suppose that $r \geq 2$, that s is a cell of a cad D of E^r, and that c is the unique cell of the induced cad D' of E^{r-1} for which $s \in Z(c)$. Suppose that we have already determined (i.e. found the number of sections of) the stack $S(c) \subset D$, by isolating the real roots of some suitable $g(x_r) \in Q(\alpha)[x_r]$. Thus, about each real root of $g(x_r)$, we have an open isolating interval with rational number endpoints. For each basis polynomial B_i, we compute the greatest squarefree divisor $d(x_r)$ of $B_i(\alpha, x_r)$. $d(x_r)$ has the same roots as $B_i(\alpha, x_r)$, but only simple roots; see Kaltofen (1982) for more information on greatest squarefree divisors. Since we are assuming $S(c)$ to be B-invariant, any root of $B_i(\alpha, x_r)$ is a root of $g(x_r)$. Hence for any section s of $S(c)$, B_i vanishes on s if and only if d has opposite signs at the endpoints of the isolating interval for the unique root of $g(x_r)$ that corresponds to s. If B_i doesn't vanish on s, then it has the same sign on s, on the sector immediately above s, and on the sector immediately below s. We can determine the sign of B_i on sectors of $S(c)$ as follows. The endpoints of the isolating intervals for the roots of $g(x_r)$ give us sample points of the form $< \alpha, b >$, b rational, for the sectors of $S(c)$ (much as we got sample points for the sectors of stacks in Section 5 of Arnon *et al.*, 1984a). By evaluating each $B_i(\alpha, b)$, we determine the sign of B_i on each sector of $S(c)$. Fig. 4 gives the algorithm *BasisSignaturesOverCell* that embodies this strategy. The map *gsfd* in the algorithm is "greatest squarefree divisor".

To infer input polynomial signatures from basis signatures, we need only a few more observations. Suppose $C(x) = \mathit{content}(A_i)$. If $C(\alpha) = 0$ then A_i vanishes on every element of $S(c)$, and we are done. If not, we use the sign of $\mathit{content}(A_i)$, and the factorization of $pp(A_i)$ ($pp(A_i)$ denotes the primitive part of A_i) as a power product of basis polynomials, to "infer" the sign of A_i on each element of the stack. Algorithm *InputSignaturesOverCell* in Fig. 5 has the details.

With algorithm *InputSignaturesOverCell* available to us, we have the following situation. We have no need to convert any extended sample point representations to primitive form in the cad of r-space. In dimensions less than r, we need primitive sample points for any cell or cluster that we are going to "extend over", i.e. build a stack over, as we extend our cad to the next higher-dimensional space. Thus the first step of *ExtendCadClusters* is to compute the sign-invariant connected components of

$\Sigma \leftarrow$ **BasisSignaturesOverCell** (B, p, J)

Inputs Given $A = (A_1, \ldots, A_n) \subset I_r$, $r \geq 1$, $B = (B_1, \ldots, B_m)$ is a basis for $prim(A)$, $p = (p_1, \ldots, p_{r-1})$ is a primitive sample point for a $PROJ(B)$-invariant and $cont(A)$-invariant cell c in a cad of E^{r-1}, i e each p_i is an element of $Q(\gamma)$ for some real algebraic number γ If $r = 1$, then $p = \emptyset$ and $c = E^0$ $J = (J_1, J_2, \ldots, J_k)$, $k \geq 0$, is a list of open isolating intervals for the k real roots $\lambda_1 < \cdots < \lambda_k$ of some nonzero univariate real polynomial $g = g(x_r)$, such that for each j, $1 \leq j \leq k$, the point $(p_1, \ldots, p_{r-1}, \lambda_j)$ lies on the j^{th} section of a B-invariant (and hence also A-invariant) stack $S(c)$ over c

Output $\Sigma = (\sigma_1, \ldots, \sigma_{2k+1})$, such that $\sigma_j = (\sigma_{1,j}, \ldots, \sigma_{m,j})$ is the signature of the j^{th} element of $S(c)$ with respect to B

(1) [Do it] For $i = 1, \ldots, m$, do set $h(x_r) \leftarrow B_i(p_1, \ldots, p_{r-1}, x_r)$, set $d(x_r) \leftarrow gsfd(h(x_r))$; set $d = 0$ if $h = 0$, set $\rho_{i,2k+1} \leftarrow sign(d) = sign(leadingCoefficient(d))$, set $\sigma_{i,2k+1} \leftarrow sign(h)$, for $j = k, k-1, \ldots, 1$ do Let $J_j = (u_j, v_j)$, set $\rho_{i,2j-1} \leftarrow sign(d(u_j))$, if $\rho_{i,2j-1} \neq \rho_{i,2j+1}$, then $\sigma_{i,2j} \leftarrow 0$, and $\sigma_{i,2j-1} \leftarrow sign(h(u_j))$, else $\sigma_{i,2j-1} \leftarrow \sigma_{i,2j} \leftarrow \sigma_{i,2j+1}$ □

Figure 4 Algorithm BasisSignaturesOverCell

$T \leftarrow$ **InputSignaturesOverCell** (A, B, p, J)

Inputs $A = (A_1, \ldots, A_n) \subset I_r$, $r \geq 1$, and the remaining inputs are as for algorithm *BasisSignaturesOverCell*

Output $T = (\tau_1, \ldots, \tau_{2k+1})$, such that $\tau_j = (\tau_{1,j}, \ldots, \tau_{n,j})$ is the signature of the j^{th} element of $S(c)$ with respect to A

(1) [Get basis signatures] $\Sigma \leftarrow BasisSignaturesOverCell(B, p, J)$

(2) [Infer input polynomial signatures] Recall that we follow the convention that $sign(content(F))$ is chosen to be $sign(F)$, for any $F \in I_r$ For each $A_i \in A$, there exist nonnegative integers $e_{i,1}, \ldots, e_{i,m}$ such that $A_i = content(A_i) \prod_{u=1}^{m} B_u{}^{e_{i,u}}$. For $i = 1, \ldots, n$, and for $j = 1, \ldots, 2k+1$ do $\tau_{i,j} \leftarrow sign(content(A_i)) \prod_{u=1}^{m} \sigma_{u,j}^{e_{i,u}}$ □

Figure 5 Algorithm InputSignaturesOverCell

G', and for each (i.e. for each sign-invariant cluster of D'), insure that at least one of its constituent cells has a primitive sample point. Needing a primitive sample point only for one cell in each sign-invariant cluster, rather than for each cell of the cad, is a key reason why the cluster-based cad algorithm is faster than the original cad algorithm in those cases that it is. For cad's of E^3, however, the saving realized here in the extension step are somewhat offset by the fact that our current E^3 adjacency algorithm (given in Section 4) needs primitive sample points for certain additional cells of the induced cad of E^2. In general we should expect that adjacency algorithms may require us to perform certain addition extended-to-primitive conversions of sample point representations.

Fig. 6 gives the complete algorithm *ExtendCadClusters*. The reader will see that it is essentially a formalization of our discussion in Section 1.2. We say that a cad graph is *initial* if the "initial" adjacencies are present in it, where these are (1) the intrastack adjacencies of each stack of that cad, and (2) the induced adjacencies as defined in Section 1. Note that even though we use a basis B for $prim(A)$ to determine the stacks of our cad's, *ExtendCadClusters* constructs A-invariant clusters prior to extending. In general, A-invariant clusters will be coarser than B-invariant clusters, and we want the extend over the coarsest possible clusters to minimize the number of primitive sample points that we need.

Suppose $r = 2$. Since maximal sign-invariant clusters in a cad of E^1 each consist of a single cell, by Step (2.1) of *ExtendCadClusters*, we see that we get primitive sample points for all 1-cells and all 2-cells, and an extended or primitive sample point for each 0-cell, of the cad of E^2 that we build.

4 Adjacency step

For each i, the role of what we call the "adjacency" subalgorithm of cluster-based cad construction is to add non-initial (interstack) adjacencies to the graph for the cad of E^i. It is not necessary to actually have such an adjacency algorithm for each i. If we wish, we need compute no adjacencies beyond initial adjacencies, for any value(s) of $i \leq r$. For example, since at present we only have implemented adjacency algorithms for $i = 1, 2, 3$, the adjacency step in our implementation of the cluster-based cad algorithm is currently null for $i \geq 4$. As might be expected, if we have a null adjacency algorithm for the cad of E^i, then our graph for that cad is almost certainly partial, and the sign-invariant components of that graph give us a clustering of the cad that is almost certainly not maximal.

In Fig. 7 we give our 2-space adjacency algorithm, which is an adaptation of algorithm CADA2 of Arnon *et al.* (1984b). Note that when $r = 2$, the maximal sign-invariant clusters in the induced cad (of 1-space) are just singleton clusters, i.e. the individual cells. Hence when $r = 2$ we construct all adjacencies of the cad of E^2 that we build, and so clearly the sign-invariant components of the graph for this cad correspond to maximal sign-invariant clusters of D.

As for cells, we say that two (distinct) clusters (of a given clustering of a given cad) are *adjacent* if their union is connected. It is not hard to show that clusters C_1 and C_2 are adjacent if and only if there is a cell c_1 of C_1, and a cell c_2 of C_2, such that c_1 and c_2 are adjacent. We call an adjacency of cells belonging to different clusters an *intercluster* adjacency, whereas an adjacency of cells in the same cluster is an *intracluster* adjacency.

One possible adjacency algorithm for E^3 would be to simply build (all interstack) adjacencies in E^3 over all intercluster adjacencies of the induced cad of the plane, using

$G \leftarrow$ **ExtendCadClusters** $(A, B, G', ExtendCellToStack)$

Inputs: $A \subset I_r$. $B \subset I_r$ is a basis for $prim(A)$. $G' = (A', B', V', E', G'')$ is a graph for a cad D' of E^{r-1} such that an A-invariant stack exists over each cell c of D'. $ExtendCellToStack(c, B', B; g, J, I, L)$ is a procedure with the following specifications. For inputs: c is a cell in a basis-determined cad D' of E^{r-1}, $r \geq 2$. $B' \subset I_{r-1}$ is a basis for D'. $B \subset I_r$ is a basis, such that each element of B is either delineable or nullified on c. For outputs: Let p be the sample point for c, and suppose the real algebraic number γ is a primitive element for p (see Arnon *et al.*, 1984a, for this terminology). g is a nonzero squarefree univariate polynomial $g(x_r)$ with coefficients in the field $Q(\gamma)$ whose real roots are in one-one correspondence with the sections of a B-invariant stack S over c. J is a list of isolating intervals for the real roots of g. I is a list of cell indices for the elements of S (since we know the cell index of c, we know the indices of elements of S). L is a list of the intrastack adjacencies of S.

Output: $G = (A, B, V, E, G')$ is an initial graph for an A-invariant cad of E^r.

(1) [Get sign-invariant clusters.] Do a sign-invariant connected components computation in G', to get a certain sign-invariant clustering of D'.

(2) [Process each cluster.] Initialize V and E to the empty set. For each sign-invariant cluster C of D', do the following steps (2.1) - (2.3).

(2.1) [Build a stack over the representative cell of the cluster.] Find a primitive sample point p for an (arbitrary) "representative" cell c of C; if none currently exists, construct one (by extended-to-primitive conversion) for some cell c of C, of highest possible dimension. Call $ExtendCellToStack(c, B', B; g, J, I, L)$. Set $T \leftarrow InputSignaturesOverCell(A, B, p, J)$. Using I, T, p, g, and J, we make a cell triple for each cell of the stack, as follows. We know the indices of all cells in the stack (from I), and their signatures (from T). Make a primitive sample point for each sector of the stack, and an extended sample point for each section (using p, g, and J). However, if the primitive element for p is of degree one, i.e. a rational number, then g has rational number coefficients, and so make a primitive rather than an extended sample point for each section of the stack. Add all cell triples to V. Add the intrastack adjacencies of L to E.

(2.2) [Infer stacks over the remaining cells of the cluster.] For all cells of C other than the one just used, do the following three steps: (1) make up cell triples for a stack over it, each triple consisting of an index inferred from the triple for the corresponding cell of the stack over c, a signature copied from the triple for the corresponding cell of the stack over c, and a null sample point; (2) add the triples for this stack to V; and (3) add the intrastack adjacencies for this stack to E.

(2.3) [Induced adjacencies of each induced cluster.] Let $2k + 1$ be the number of elements of the stack over the representative cell of C (thus each stack over an element of C also has $2k+1$ elements). For each intracluster adjacency $\{d, e\}$ of C, and for $i = 1, ..., 2k + 1$, record in E that the i^{th} element of the stack over d is adjacent to the i^{th} element of the stack over e □

Figure 6: Algorithm ExtendCadClusters

AdjacenciesTwoSpace (G)

Inputs G ($1, B, 1, E, G'$) is a graph for a basis-determined 1-invariant cad D of E^2 (with basis B).

Output G is modified so that it contains additional adjacencies among cells of D, in particular, if G is initial at input, then it is a full graph for D at output

(1) [Interstack adjacencies] Set $B^* \leftarrow \coprod B$ Let $a_1 < a_2 < \quad < a_{2k} < a_{2k+1}$, $k \geq 0$, be the sample points for the cells of D' (Each a_{2i+1} is a rational sample point for a 1-cell, each a_{2i} is an algebraic sample point for a 0-cell) For $i = 1, \quad , k$, call algorithm SSADJ2 of Arnon *et al* (1984b) with inputs B^*, a_{2i}, a_{2i-1}, and a_{2i+1}, add the contents of its outputs L_1 and L_2 to G, i e to E Note that the section numbers which occur in the adjacencies returned by SSADJ2 must first be converted into the indices of the corresponding cells of D, for example, if the list L_1 returned by the i^{th} call to SSADJ2 contains the adjacency $\{3, 2\}$, it must be converted to $\{(2i, 6), (2i-1, 4)\}$ before being added to L. Infer the remaining interstack adjacencies between $S(c_{2i})$ and $S(c_{2i-1})$, and between $S(c_{2i})$ and $S(c_{2i+1})$, as described at the end of Section 2 of Arnon *et al* (1984b), and add them to G □

Figure 7· Algorithm AdjacenciesTwoSpace.

the algorithms of Arnon *et al* (1988) Assuming that we started with an initial graph for the cad of E^3, we clearly would end up with a full graph for this cad A sign-invariant components computation in this graph would then obviously yield maximal sign-invariant clusters of the cad D of E^3. We now show that there is a proper subset Λ of the set of all intercluster adjacencies of the induced cad of E^2, such that given an initial graph for D, if we then build (all interstack) adjacencies over each element of Λ, then a sign-invariant connected components computation in the resulting graph yields maximal sign-invariant clusters of D Besides the obvious reduction in the amount of adjacency determination we have to do in E^3, it turns out that the particular Λ that we show is sufficient allows us to often avoid the most costly extended-to-primitive sample point conversions in E^2

Let us first determine what kinds of intercluster adjacencies can occur among maximal sign-invariant clusters of a cad of the plane. We assume that this cad has the boundary property Clearly there can be no intercluster adjacency between 0-clusters It is also clear that there can be no intercluster adjacencies between two maximal 2-clusters, since by the Intermediate Value Theorem, all 2-clusters have the same signature with respect to the input polynomials So that leaves us with the possibility of adjacencies between 0- and 1-clusters, 0- and 2-clusters, 1- and 1-clusters, and 1- and 2-clusters For the first of these cases, clearly all intercluster adjacencies are $\{0, 1\}$, i e involving a 0-cell of the 0-cluster and a 1-cell of the 1-cluster. In the second case, there can be both $\{0, 1\}$ and $\{0, 2\}$ intercluster adjacencies In the third case, there can only be $\{0, 1\}$ intercluster adjacencies, since (by the boundary property) adjacent cells in the cad of the plane have different dimensions In the fourth case, by the boundary property, there can be $\{0, 1\}$, $\{0, 2\}$, and $\{1, 2\}$ intercluster adjacencies

In point of fact, adjacencies of two 1-clusters seem to occur rarely, for certain "peculiar" sorts of inputs For example, let $F(x, y) = y + x$, $G(x, y) = y - x$, and $H(x, y) = y$, and let $A = \{FGH, FG, FH\}$ Fig 8 shows an A-invariant cad of the plane that illustrates both intercluster adjacencies of a 1-cluster and a 1-cluster, and an adjacent

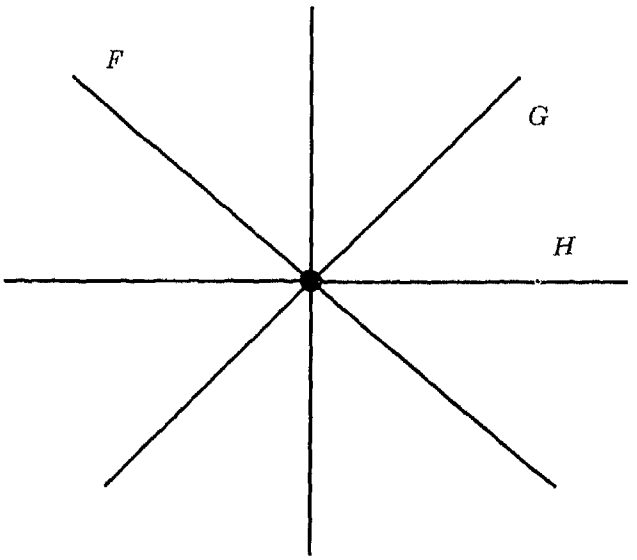

Figure 8: Sample cylindrical algebraic decomposition of the plane.

1-cluster and 2-cluster for which their only intercluster adjacency is a $\{0, 2\}$ adjacency of cells.

As for cells, the *boundary* of a cluster C is the set of all limit points of $R = \cup C$ which are not contained in R. It is not hard to see that clusters "do not have the boundary property", i.e. if two clusters are adjacent, then it is not necessarily the case that one is contained in the boundary of the other. For example, the tacnode curve is defined by the equation:

$$F(x, y) = y^4 - 2y^3 + y^2 - 3x^2y + 2x^4 = 0.$$

Fig. 9 shows an F-invariant cad of the plane. The curve itself is a maximal sign-invariant 1-cluster of this cad, and clearly, for any sign-invariant 2-cluster C, neither C nor the curve is contained in the boundary of the other.

We now prove a theorem that points the way to our actual 3-space adjacencies algorithm. The basic idea, given that clusters in the plane may fail to have the boundary property, is that for a pair of adjacent clusters of a cad of the plane, we find subclusters of each that are as large as possible while still having the property that one is contained in the boundary of the other. It then follows that it is sufficient to build adjacencies in E^3 over just one of the intercluster adjacencies between each such pair of subclusters.

We now define the central notion for our theorem. Given adjacent clusters C_1 and C_2 of some cad, we say that subclusters $Q_1 \subset C_1$ and $Q_2 \subset C_2$ are *cobounding subclusters* for C_1 and C_2 if

1. Each cell of Q_1 is in the boundary of one or more cells of Q_2, and
2. For each cell of Q_2, there are one or more cells of Q_1 contained in its boundary, and

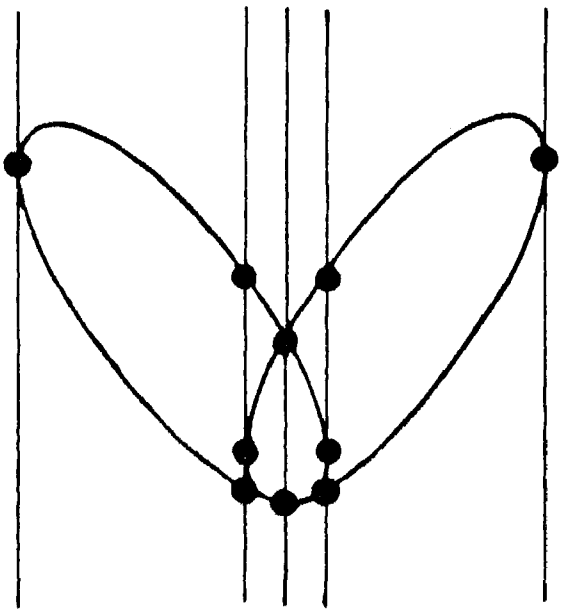

Figure 9: Cylindrical algebraic decomposition of tacnode curve.

3. If cells c_1 and c_2 of Q_2 are adjacent, then there are cells d_1 and d_2 of Q_1 such that $d_1 \subset \partial c_1$, $d_2 \subset \partial c_2$, and either $d_1 = d_2$, or d_1 and d_2 are adjacent.

Clearly if Q_1 and Q_2 are cobounding subclusters for C_1 and C_2, if $R_1 = \cup Q_1$, and if $R_2 = \cup Q_2$, then $R_1 \subset \partial R_2$. If Q_1 and Q_2 are cobounding subclusters for C_1 and C_2, and if for any other cobounding subclusters O_1 of C_1 and O_2 of C_2, it is the case that either $O_1 \cap Q_1 = \emptyset$, or $O_2 \cap Q_2 = \emptyset$, or $O_1 \subset Q_1$ and $O_2 \subset Q_2$, then we say that Q_1 and Q_2 are *maximal* cobounding subclusters for C_1 and C_2.

Let C_1 and C_2 be adjacent sign-invariant clusters of D' such that $R_1 \subset \partial R_2$, where $R_1 = \cup C_1$ and $R_2 = \cup C_2$. Let $S(R_1)$ be a stack over R_1 and $S(R_2)$ a stack over R_2. We say that $S(R_1)$ and $S(R_2)$ are *adjacent*. If for any section s of $S(R_2)$, $\partial s \cap Z^*(R_1)$ is a section t of $S^*(R_1)$, then we say that $S^*(R_2)$ has the *unique section boundary property (USBP)* in $S^*(R_1)$.

THEOREM **4.1** *Let D be a basis-determined cad of E^r, such that the induced cad D' of E^{r-1} has the boundary property. Let C_1 and C_2 be adjacent sign-invariant clusters of D', and suppose that $Q_1 \subset C_1$ and $Q_2 \subset C_2$ are cobounding subclusters for C_1 and C_2. Let R_1 and R_2 be the respective unions of Q_1 and Q_2, and suppose that for any cells c, d of $Q_1 \cup Q_2$, if $d \subset \partial c$, then $S^*(c)$ has the unique section boundary property in $S^*(d)$. Let $S(R_1)$ and $S(R_2)$ denote the unique stacks over R_1 and R_2 with which D is compatible, in the sense that each element of one of these stacks is a union of elements of D. Then $S^*(R_2)$ has the unique section boundary property in $S^*(R_1)$.*

PROOF. Suppose the assertion to be false, and let s be some section of $S^*(R_2)$ whose boundary points in $Z^*(R_1)$ are not a section of $S^*(R_1)$. Then there exist cells t_k and t_l

of D such that: t_k and t_l are contained in s, $t_k \in S(c_k)$ and $t_l \in S(c_l)$ for cells c_k and c_l of Q_2, there are cells d_k and d_l of Q_1 such that $d_k \subset \partial c_k$ and $d_l \subset \partial c_l$, and where $u_k \in S(d_k)$ and $u_l \in S(d_l)$ are the respective boundary sections of t_k and t_l, u_k is section n_k of its stack, u_l is section n_l of its stack, and $n_k \neq n_l$. There is a sequence (chain) of adjacent cells in s joining t_k and t_l. Each t_i in this chain is a section of $S(c_i)$ for some $c_i \in Q_2$, and for each such c_i, there is a $d_i \in Q_1$ such that $d_i \subset \partial c_i$, and either $d_i = d_{i+1}$ or d_i and d_{i+1} are adjacent. Since $S^*(c_i)$ has the USBP in $S^*(d_i)$, t_i has a boundary section u_i in $S^*(d_i)$. Then there exists a j for which u_j is section n_j of its stack, u_{j+1} is section n_{j+1} of its stack, and $n_j \neq n_{j+1}$. Suppose without loss of generality that $c_j \subset \partial c_{j+1}$, hence $t_j \subset \partial t_{j+1}$, hence $u_j \subset \partial t_{j+1}$. Since d_i and d_{i+1} are identical or adjacent, clearly section n_{j+1} of $S^*(d_j)$ is also contained in ∂t_{j+1}, hence both sections n_j and n_{j+1} of $S^*(d_j)$ are contained in ∂t_{j+1}, contradicting the USBP of $S^*(c_{j+1})$ in $S^*(c_j)$ □

By the results of Arnon *et. al.* (1988), the hypotheses of Theorem 4.1 are satisfied for each pair of cobounding subclusters for each pair of adjacent maximal sign-invariant clusters of the induced cad of E^2. Hence, for each such pair of clusters in the plane, it is sufficient to build adjacencies in E^3 over just one of the intercluster adjacencies between each of their pairs of maximal cobounding subclusters, and this is what we will do.

Let us now consider the task of finding the pairs of maximal cobounding subclusters for a pair of adjacent clusters of the induced cad D' of E^2. Life is made easier with the following concept. Suppose for adjacent clusters C_1 and C_2 of D', that whenever cells $c_1 \in C_1$ and $c_2 \in C_2$ are adjacent, $c_1 \subset \partial c_2 \subset \partial C_2$. Then we say that C_1 and C_2 have *one-way boundary inclusions*. The next theorem tells us that clusters in E^2 have one-way boundary inclusions.

THEOREM **4.2** *Suppose given a maximal sign-invariant clustering of a cad with the boundary property, such that some cell of a cluster C_1 is contained in the boundary of (one or more cells of) a cluster C_2. Then for any cells $c_1 \in C_1$ and $c_2 \in C_2$, if c_1 and c_2 are adjacent, then $c_1 \subset \partial c_2 \subset \partial C_2$.*

PROOF. Suppose cell d_1 of cluster C_1 is contained in boundary of C_2; then clearly d_1 is in the boundary of some cell d_2 of C_2. Since C_1 and C_2 are different maximal sign-invariant clusters, there is some input polynomial F which vanishes on one but not the other. Since real varieties are closed, F must vanish on d_1, i.e. on C_1, but not on d_2, i.e. not on C_2. Suppose now that cells $c_1 \in C_1$ and $c_2 \in C_2$ are adjacent. Then one contains a limit (boundary) point of other, hence by the boundary property, one is contained in boundary of the other. But then $c_1 \subset \partial C_2$, since $c_2 \subset \partial C_1$ would imply that F vanishes on c_2, a contradiction □

Fig. 10 gives an algorithm to find all pairs of maximal cobounding subclusters for a given pair of adjacent clusters. In Fig. 11 we give our 3-space adjacency algorithm *AdjacenciesThreeSpace*. It assumes that the particular value of the procedure parameter *ExtendCellToStack* of algorithm CLCAD that is called for in Section 5, i.e. algorithm *ExtendCellToStack* of Arnon *et al.* (1988), has been used to determine the stacks of the cad D of E^3. The various adjacency subalgorithms (e.g. *AdjacenciesOver*01) that *AdjacenciesThreeSpace* calls are from Arnon *et al.* (1988). Each such subalgorithm takes the two cells of an adjacency as inputs, and we are required to have primitive sample points for both. We assume that extended-to-primitive conversion is done as needed for these calls. A *nullifying* 0-cluster is a 0-cluster on whose unique constituent 0-cell some element of B is nullified.

$K \leftarrow$ **MaximalCoboundingSubclusters** (C_1, C_2)

Inputs: C_1 and C_2 are disjoint clusters of a cad of E^r which has the boundary property, such that C_1 and C_2 have one-way boundary inclusions if they are adjacent

Output If C_1 and C_2 are not adjacent, then K is the empty list Otherwise K is a list ($((Q_{1,1}, Q_{1,2}), L_1), ((Q_{2,1}, Q_{2,2}), L_2),$ $, ((Q_{n,1}, Q_{n,2}), L_n)$), such that $(Q_{1,1}, Q_{1,2})$, $(Q_{2,1}, Q_{2,2})$, $\ldots$, $(Q_{k,1}, Q_{k,2})$ are the maximal cobounding subclusters for C_1 and C_2, and for each $(Q_{i,1}, Q_{i,2})$, L_i is a list of all intercluster adjacencies between $Q_{i,1}$ and $Q_{i,2}$

(1) [Do it] For each intercluster adjacency $\{c_1, c_2\}$ between C_1 and C_2, create an initial element ($(\{c_1\}, \{c_2\}, \{\{c_1, c_2\}\})$ of K Then until no more coalescing is possible, attempt to "paste together" pairs $(Q_{i,1}, Q_{i,2}), L_i)$ and $(Q_{j,1}, Q_{j,2}), L_j)$ of elements of K We attempt to paste such a pair by first checking whether $Q_{k,1} \leftarrow Q_{i,1} \cup Q_{j,1}$ is a subcluster of C_1 and whether $Q_{k\,2} \leftarrow Q_{i,2} \cup Q_{j,2}$ is a subcluster of C_2. If so, then we set L_k to be $L_i \cup L_j$ plus any other intercluster adjacencies of $Q_{k,1}$ and $Q_{k,2}$, and check whether $Q_{k,1}$ and $Q_{k\,2}$ are cobounding subclusters of C_1 and C_2 □

Figure 10. Algorithm MaximalCoboundingSubclusters.

The reader should now be convinced of the following proposition

THEOREM **4.3** *Let D be a basis-determined sign-invariant cad of E^3, and let D' denote the induced sign-invariant cad of E^2 Suppose we have a graph representation for D which contains D's initial adjacencies, and the other adjacencies of it that are added by AdjacenciesThreeSpace Then the clusters of D that we obtain by a sign-invariant connected components computation in the graph for D are maximal (sign-invariant) clusters*

From algorithm *AdjacenciesThreeSpace* we see that we do not avoid all extended-to-primitive conversions of 0-cell sample points in the induced cad of the plane we are required to have a primitive sample point for each 0-dimensional maximal sign-invariant cluster in E^2, and possibly also for certain 0-cells in 1-clusters We now indicate how it is that the particular such conversions that actually are done are typically not as expensive as the ones that are not done Consider for example the sample points of 0-clusters Such 0-clusters are usually "topologically significant", e g they are typically the intersection points of two curves in E^2 It has been our empirical observation that the sample points of such "topologically significant" 0-cells often do not require field extension (i e nontrivial primitive element computation) in the conversion of their extended representations to primitive In other words, the algebraic polynomial which is part of their extended representation is typically linear Some explanation of this phenomenon is provided by Muller's observation (Muller, 1978), that probably at most one intersection of two (random) algebraic plane curves lies on any particular line in the plane, and so for any $F, G \in I_2$, the curve defined by F and the curve defined by G probably only have one intersection on a line $x = \alpha$, where α is the sample point of a 0-cell in the induced cad of E^1 If so, then $gcd(F(\alpha, y), G(\alpha, y))$ is linear, since each of its roots corresponds to an intersection point of the two curves Hence the y-coordinates β of intersection points are likely to have the property that $Q(\alpha, \beta) = Q(\alpha)$, i e the primitive element algorithm is trivial

The tacnode provides an illustrative example of how we are often able to avoid

AdjacenciesThreeSpace (G)

Inputs $G = (A, B, V, E, G')$ is a graph for a basis-determined A-invariant cad D of E^3 (with basis B), such that D has the boundary property, and such that if cell d of D' is contained in ∂c for a cell c of D' on which no element of B is nullified, then $S^*(c)$ has the unique section boundary property in $S^*(d)$, and such that G' contains all adjacencies of D', and has a primitive or extended sample point for each of its cells

Output G is modified so that it contains additional adjacencies among cells of D, in particular, if G is initial at input, then the sign-invariant connected components of G correspond to maximal sign-invariant clusters of D

(1) [Construct maximal sign-invariant clusters of induced cad] Do a sign-invariant connected components computation in the G' graph, to get (maximal) sign-invariant clusters of D'

(2) [Process (1-cluster, 1-cluster) adjacencies] For each pair C_1, C_2 of adjacent 1-clusters of D', and for each of their (0,1) intercluster adjacencies $\{c^0, c^1\}$, set $L \leftarrow$ *AdjacenciesOver*01(c^0, c^1, B', B) and add the adjacencies of L to E

(3) [Process (1-cluster, 2-cluster) adjacencies] For each pair C_1, C_2 of adjacent 1-cluster and 2-cluster of D', do $K \leftarrow$ *MaximalCoboundingSubclusters*(C_1, C_2). For each L_i of K, do the following loop. If L_i contains (1,2) adjacencies, then let $\{c_i^1, c_i^2\}$ be one of them, set $L \leftarrow$ *AdjacenciesOver*12(c_i^1, c_i^2, B), add the adjacencies of L to E, and exit this loop iteration Otherwise, if L_i contains (0,1) adjacencies, then let $\{c_i^0, c_i^1\}$ be one of them, set $L \leftarrow$ *AdjacenciesOver*01(c_i^0, c_i^1, B', B), add the adjacencies of L to E, and exit this loop iteration Otherwise, let $\{c_i^0, c_i^2\}$ be a (0,2) adjacency of L_i, set $L \leftarrow$ *AdjacenciesOverNonNullifying*02(c_i^0, c_i^2, B', B), and add the adjacencies of L to E

(4) [Process adjacencies of non-nullifying 0-clusters] For each non-nullifying 0-cluster C, with unique constituent cell c^0, do the following two steps First, for each 1-cluster C_1 which is adjacent to C, and for each of their (0,1) intercluser adjacencies $\{c^0, c^1\}$, set $L \leftarrow$ *AdjacenciesOver*01(c^0, c^1, B', B) and add the adjacencies of L to E Second, for each 2-cluster C_2 which is adjacent to C, do $K \leftarrow$ *MaximalCoboundingSubclusters*(C, C_2), and for each L_i of K, do the following loop If L_i contains (0,1) adjacencies, then let $\{c_i^0, c_i^1\}$ be one of them, set $L \leftarrow$ *AdjacenciesOver*01(c_i^0, c_i^1, B', B), add the adjacencies of L to E, and exit this loop iteration Otherwise, let $\{c_i^0, c_i^2\}$ be a (0,2) adjacency of L_i, set $L \leftarrow$ *AdjacenciesOverNonNullifying*02(c_i^0, c_i^2, B', B), and add the adjacencies of L to E

(5) [Process adjacencies of nullifying 0-clusters] For each nullifying 0-cluster C, with unique constituent cell c^0, do the following steps For each (0,1) adjacency $\{c^0, c^1\}$ of D', set $L \leftarrow$ *AdjacenciesOver*01(c^0, c^1, B', B) and add the adjacencies of L to E For each (0,2) adjacency $\{c^0, c^2\}$ of D', set $L \leftarrow$ *AdjacenciesOverNullifying*02(c^0, c^2, B', B), and add the adjacencies of L to E □

Figure 11 Algorithm AdjacenciesThreeSpace

$G \leftarrow$ **CLCAD** $(A, Basis, Projection, ExtendCellToStack, Adjacencies)$

Inputs: A is a finite subset of I_r, for some $r \geq 1$. *Basis* is a procedure which, for any $i \geq 1$, given a subset U of I_i, computes a basis for $prim(U)$. *Projection* is a procedure which, for any $i \geq 2$, maps a subset of I_i to a subset of I_{i-1} having the expected properties (cf. Theorem 2.4 of Arnon *et al.*, 1988). $ExtendCellToStack(c, B', B; g, J, I, L)$ is a procedure with the same specifications as the input parameter of the same name to algorithm *ExtendCadClusters* of Section 3. *Adjacencies* is a procedure which, for any $i \geq 2$, given a graph for a cad of E^i, finds certain of its interstack adjacencies and adds them (i.e. adds the corresponding edges) to the graph.

Output: $G = (A, B, V, E, G')$ is a graph representation for an A-invariant cad D of E^r.

(1) [$r = 1$ (base case).] Set $B \leftarrow Basis(A)$. If $r > 1$, then go to step (2). Construct a list J of open isolating intervals for the real roots of the elements of B, thus determining the cells of a cad D of E^1. Set $T \leftarrow InputSignaturesOverCell(A, B, \emptyset, J)$. Construct an index and a primitive sample point for each cell. From these and from T, create a triple for each cell, and set V to a list of all these triples. The adjacencies of D are obvious; collect them as the set E. Set G' to $\emptyset$, to complete the construction of a graph G for D. Return.

(2) [$r > 1$. Initial graph.] Set $P \leftarrow Projection(A)$, and call CLCAD with inputs P, *Basis*, *Projection*, *ExtendCellToStack*, and *Adjacencies*, to obtain output G'. Call algorithm *ExtendCadClusters* of Section 3 with inputs A, B, G', and *ExtendCellToStack* to obtain an initial graph G for an A-invariant cad of E^r.

(3) [$r > 1$. Non-initial adjacencies among r-space cells.] Apply *Adjacencies* to G □

Figure 12: Algorithm CLCAD.

extended-to-primitive conversions of the sample points of 0-cells in 1-clusters. A sign-invariant cad of E^2 for the tacnode is shown in Fig. 9. Using an implementation of algorithm CAD (cf. Section 6), construction of this cad took 29 minutes, with 27 minutes of that spent in converting the extended representations of four 0-cell sample points to primitive: cells (4,2), (4,6), (8,2), and (8,6). Using CLCAD, we were able to construct the same cad of 2-space in 1 minutes; primitive sample points for cells (4,2), (4,6), (8,2), and (8,6) were not required, because they belong to a 1-dimensional sign-invariant cluster (the collection of all cells contained in the curve), which is adjacent only to 2-dimensional sign-invariant clusters, and for each such adjacent 2-cluster, each pair of maximal cobounding subclusters for the curve and the 2-cluster has an intercluster adjacency between a 2-cell of the 2-cluster and a 1-cell of the curve.

5 Main algorithm

In Fig. 12 we give our main algorithm CLCAD in a form which has various procedure parameters. Thus the exact version of the general cluster-based cad strategy that a user desires can be obtained by passing appropriate concrete procedures for these parameters, for example, one might use McCallum projection (McCallum, 1988) instead of the projection map assumed in Section 2, or one might use other adjacency algorithms than those we have given in Section 4.

Let us now list the particular concrete procedures that we pass for CLCAD's pro-

Adjacencies (G)

(1) If $r = 1$ or $r \geq 4$ then return. If $r = 2$ then $AdjacenciesTwoSpace(G)$. If $r = 3$ then $AdjacenciesThreeSpace(G)$ □

Figure 13: Algorithm Adjacencies.

cedure parameters in our current implementation; this information in effect summarizes Section 2-6. The default for calls to CLCAD is: we don't care what basis B for $prim(A)$ the procedure *Basis* computes. We set $Projection(A) = PROJ(B) \cup cont(A)$, $2 \leq i \leq r$. We pass procedure *ExtendCellToStack* of Arnon *et al.* (1988) as argument *ExtendCellToStack*. Finally, as one may expect from Sections 4, we pass the algorithm shown in Fig. 13 as argument *Adjacencies* of CLCAD.

Given the these concrete procedures as values for CLCAD's procedure parameters, and given input polynomials $A \subset I_r$ with $1 \leq r \leq 3$, the sign-invariant connected components of the undirected graph (V, E) built by algorithm CLCAD correspond to maximal sign-invariant clusters of the cad D of E^r, and the boundary of each cell of D is a (disjoint) union of lower-dimensional cells, i.e. D has the boundary property.

6 Examples

6.1 General remarks.

We have not so far performed a detailed study of our implemented cluster-based cad algorithm's behavior, but preliminary experiments indicate that its performance is sometimes better, sometimes worse, than the "original" cad algorithm (i.e. algorithm CAD of Arnon *et al.*, 1984a). Of course, the results of the comparisons we have carried out reflect the use of the particular adjacency algorithms given in Section 4. In any particular such comparison, the outcome seems to depend on the relative time of the extended-to-primitive sample point conversions that the original algorithm must do but which the cluster-based algorithm avoids, compared to the adjacency computations that the cluster-based algorithm must do but which do not occur in the original algorithm. Thus the Quartic and Ellipse examples below, for which the original algorithm was faster, most likely had easy sample point conversions relative to the cost of adjacency computations.

Fig. 14 contains a summary of the results of our comparisons. The times in it were obtained from algorithms CAD of Arnon *et al.* (1984a), and algorithm CLCAD of this paper, with both algorithms computing finest squarefree bases. Both algorithms were implemented in the SAC-2 computer algebra system (Collins, 1980), on a Vax 11/785 running Unix. The times given in the table are in minutes. $T_{original}$ is the time spent by the original cad algorithm, and $T_{clustered}$ the time spent by the cluster-based algorithm, for each example. A time of zero minutes means less than half a minute. The notation "$> n$ minutes" means that an algorithm ran for at least n minutes before either it was terminated or our computer went down. The column "Cells" gives the number of cells in the cad's built by both the original and cluster-based algorithms, and "Clusters" gives the number of maximal sign-invariant clusters in the cad built by the cluster-based

Name	$T_{original}$	$T_{clustered}$	*Cells*	*Clusters*
Tacnode	29	1	55	5
SIAM	1	1	41	15
Toptyp	> 120	4	37	9
Pair1	> 90	7	103	15
Pair2	> 83	9	127	27
Pair3	19	4	85	21
Pair4	1	1	63	15
Pair5	7	7	57	15
Quartic1	2	2	21	5
Quartic2	> 115	> 115	?	?
Quartic3	> 270	25	37	3
Quartic4	46	47	55	5
Quartic5	0	0	21	4
CADIII	0	1	51	3
Quartic	2	10	123	35
Implicit	> 300	89	855	9
SphereCatas	> 827	282	1393	9
Ellipse	9	74	2291	715

Figure 14: Sample comparisons of original and cluster-based cad algorithms.

algorithm. Sections 6.2 - 6.5 give the input polynomials for each example, and where applicable, cite a source for the example.

6.2 Miscellaneous bivariate examples.

6.2.1 Tacnode (Arnon *et al.*, 1984a)

$$y^4 - 2y^3 + y^2 - 3x^2y + 2x^4$$

.

6.2.2 SIAM papers pair of polynomials (Arnon *et al.*, 1984a, 1984b)

$$144y^2 + 96x^2y + 9x^4 + 105x^2 + 70x - 98$$
$$xy^2 + 6xy + x^3 + 9x$$

6.2.3 Toptyp algorithm example (Arnon & McCallum, 1988)

$$y^4 - 2xy^3 - x^2y^2 + y^2 + 2x^3y + x^2 - 1$$

.

6.3 Five randomly generated pairs of bivariate polynomials.

Each consisted of a quadratic and a cubic polynomial, with two-digit integer base coefficients.

6.3.1 first pair.

$$3y^2 - 2xy + 28x + 31$$
$$-8y^3 + 6x^2y - 15xy - 7y - 7x^3 + 11x + 6.$$

6.3.2 second pair.

$$-9y^2 + 30xy - 22x^2 + 21$$
$$2y^3 - 12x^2y - 12xy - 8y + 11x^2 - 2x - 2.$$

6.3.3 third pair.

$$-2y - 13x + 22$$
$$-13y^3 + 5xy^2 + 12y^2 + 14x^2y + 11y - 10x^2 + 11.$$

6.3.4 fourth pair.

$$-12xy - 15y - 30x^2 + 4x + 21$$
$$-xy^2 + 15y^2 + 8x^2y - 12y + 12x^3 + 9.$$

6.3.5 fifth pair.

$$27xy + 9x^2 - 31x + 4$$
$$5y^3 - 14xy^2 + 15y^2 + 13x^2y + 2xy + 14y - 7x^3 - 3x.$$

6.4 Five randomly generated bivariate quartics.

Each had two-digit integer base coefficients.

6.4.1 first quartic.

$$44xy^3 + 57xy^2 + 25y + 37x^3 - 31x.$$

6.4.2 second quartic.

$$-62y^4 - 29x^2y^2 - 45y^2 + 45x^3y - 5x^2y + 26x^4 + 27x - 58.$$

6.4.3 third quartic.

$$-50y^4 + 48y^3 - 8y^2 - 34x^2y - 11x^3 - 5x.$$

6.4.4 fourth quartic.

$$60xy^3 + 59y^3 - 41y^2 - 55x^3y + 47xy + 45y + 22x^4 - 38x^3 + 3x^2 - 24.$$

6.4.5 fifth quartic.

$$52x^2y^2 + 30xy^2 + 49y^2 - 4x^2y + 62xy + 9x^4 + 33x^3.$$

6.5 Trivariate examples.

6.5.1 CADIII example surface (Arnon *et al.*, 1988)

$$y^3z + xy^2 - x^3$$

6.5.2 Positive definite canonical form quartic (Arnon & Mignotte, 1988)

$$p$$
$$8pr - 9q^2 - 2p^3$$
$$256r^3 - 128p^2r^2 + 144pq^2r + 16p^4r - 27q^4 - 4p^3q^2$$

6.5.3 Curve Implicitization (Arnon, 1988)

$$505t^3 - 864t^2 + 570t + x - 343$$
$$211t^3 - 276t^2 - 90t - y + 345$$

6.5.4 Unit sphere and catastrophe surfaces (McCallum, 1988)

$$z^2 + y^2 + x^2 - 1$$
$$z^3 + xz + y$$

6.5.5 Ellipse example (Arnon & Mignotte, 1988)

$$a$$
$$a - 1$$
$$b$$
$$b - 1$$
$$b - a$$
$$c$$
$$c - 1$$
$$c + 1$$
$$c + a + 1$$
$$c + a - 1$$
$$c - a + 1$$
$$c - a - 1$$
$$b^2c^2 + b^4 - a^2b^2 - b^2 + a^2$$

7 References

Aho, A. V., Hopcroft, J., Ullman, J. (1974). *The Design and Analysis of Computer Algorithms.* Reading, Massachusetts: Addison-Wesley.

Arnon, D. S. (1979). A cellular decomposition algorithm for semi-algebraic sets. Proceedings of an International Symposium on Symbolic and Algebraic Manipulation (EUROSAM '79). *Springer Lec. Notes Comp. Sci.* **72**, 301-315.

Arnon, D. S. (1981). *Algorithms for the Geometry of Semi-Algebraic Sets.* PhD thesis, Tech. Rept. #436, Comp. Sci. Dept., Univ. Wisconsin-Madison.

Arnon, D. S., Collins, G. E., McCallum, S. (1984a). Cylindrical algebraic decomposition I: the basic algorithm, *SIAM J. Comp.* **13/4**, 865-877.

Arnon, D. S., Collins, G. E., McCallum, S. (1984b). Cylindrical algebraic decomposition II: an adjacency algorithm for the plane, *SIAM J. Comp.* **13/4**, 878-889.

Arnon, D. S. (1988). Geometric reasoning with logic and algebra. *Artificial Intelligence* (special issue on Geometric Reasoning and Artificial Intelligence; to appear).

Arnon, D. S., McCallum, S. (1988). A polynomial-time algorithm for the topological type of a real algebraic curve. *J. Symb. Comp.* **5**, (this issue).

Arnon, D. S., Mignotte, M. (1988). On mechanical quantifier elimination for elementary algebra and geometry. *J. Symb. Comp.* **5**, (this issue).

Arnon, D. S., Collins, G. E., McCallum, S. (1988). An adjacency algorithm for cylindrical algebraic decompositions of three-dimensional space. *J. Symb. Comp.* **5**, (this issue).

Collins, G. E. (1975). Quantifier elimination for real closed fields by cylindrical algebraic decomposition. Proceedings of the Second GI Conference on Automata Theory and Formal Languages. *Springer Lec. Notes Comp. Sci.* **33**, 515–532.

Collins, G. E. (1980). SAC-2 and ALDES now available. *ACM SIGSAM Bull.* **14**, 19.

Kaltofen, E. (1982). Polynomial factorization. In (Buchberger, B., Loos, R., Collins, G. E., eds.) *Computer Algebra - Symbolic and Algebraic Computation* (Computing Supplementum 4), pp. 83-94. Vienna and New York: Springer-Verlag.

Kozen, D., Yap., C. K. (1985). Algebraic cell decomposition in NC. *Proc. IEEE Conf. on Foundations of Comp. Sci. (FOCS)*, 515–521.

Loos, R. (1982). Computing in algebraic extensions. In (Buchberger, B., Loos, R., Collins, G. E., eds.) *Computer Algebra - Symbolic and Algebraic Computation* (Computing Supplementum 4), pp. 173-187. Vienna and New York: Springer-Verlag.

McCallum, S. (1988). An improved projection operation for cylindrical algebraic decomposition of three-dimensional space. *J. Symb. Comp.* **5**, (this issue).

Müller, F. (1978). *Ein exakter Algorithmus zur nichtlinearen Optimierung für beliebige Polynome mit mehreren Veranderlichen*, Meisenheim am Glan: Verlag Anton Hain.

Prill, D. (1986). On approximation and incidence in cylindrical algebraic decompositions. *SIAM J. Comp.* **15**, 972–993.

Schwartz, J. T., Sharir, M. (1983). On the 'piano movers' problem II. General techniques for computing topological properties of real algebraic manifolds. *Adv. Applied Math.* **4**, 298–351.