Journal of Symbolic Computation 123 (2024) 102288

Contents lists available at ScienceDirect

Journal of Symbolic Computation

journal homepage: www.elsevier.com/locate/jsc

Levelwise construction of a single cylindrical algebraic cell

Jasper Nalbach[a], Erika Ábrahám[a], Philippe Specht[a], Christopher W. Brown[b], James H. Davenport[c], Matthew England[d]

[a] RWTH Aachen University, 52056 Aachen, Germany
[b] United States Naval Academy, 597 McNair Road, Annapolis, MD 21402-5002, United States
[c] University of Bath, Claverton Down, Bath BA2 7AY, United Kingdom
[d] Coventry University, Coventry CV1 2TL, United Kingdom

ARTICLE INFO

Article history:
Received 20 July 2023
Accepted 30 November 2023
Available online 5 December 2023

Keywords:
Satisfiability modulo theories
Cylindrical algebraic decomposition
Non-linear real arithmetic
Model-constructing satisfiability calculus
Formal proofs

ABSTRACT

Satisfiability modulo theories (SMT) solvers check the satisfiability of quantifier-free first-order logic formulae over different theories. We consider the theory of *non-linear real arithmetic* where the formulae are logical combinations of polynomial constraints. Here a commonly used tool is the *cylindrical algebraic decomposition (CAD)* to decompose the real space into cells where the constraints are truth-invariant through the use of *projection polynomials*.

A CAD encodes more information than necessary for checking satisfiability. One approach to address this is to repackage the CAD theory into a search-based algorithm: one that guesses sample points to satisfy the formula, and generalizes guesses that conflict constraints to cylindrical cells around samples which are avoided in the continuing search. This can lead to a satisfying assignment more quickly, or conclude unsatisfiability with far fewer cells. A notable example of this approach is Jovanović and de Moura's *NLSAT* algorithm. Since these cells are being produced locally to a sample there is scope to use fewer projection polynomials than the traditional CAD projection. The original NLSAT algorithm reduced the set a little; while Brown's *single cell construction* reduced it much further still. However, it refines a cell polynomial-by-polynomial, meaning the shape and size of the cell produced depends on the order in which the polynomials are considered.

E-mail addresses: nalbach@cs.rwth-aachen.de (J. Nalbach), abraham@cs.rwth-aachen.de (E. Ábrahám), philippe.specht@rwth-aachen.de (P. Specht), wcbrown@usna.edu (C.W. Brown), j.h.davenport@bath.ac.uk (J.H. Davenport), matthew.england@coventry.ac.uk (M. England).

https://doi.org/10.1016/j.jsc.2023.102288

The present paper proposes a method to construct such cells *levelwise*, i.e. built level-by-level according to a variable ordering instead of polynomial-by-polynomial for all levels. We still use a reduced number of projection polynomials, but can now consider a variety of different reductions and use *heuristics* to select the projection polynomials in order to optimize the shape of the cell under construction. The new method can thus improve the performance of the NLSAT algorithm. We formulate all the necessary theory that underpins the algorithm as a *proof system*: while not a common presentation for work in this field, it is valuable in allowing an elegant decoupling of heuristic decisions from the main algorithm and its proof of correctness. We expect the symbolic computation community may find uses for it in other areas too. In particular, the proof system could be a step towards formal proofs for non-linear real arithmetic.

This work has been implemented in the SMT-RAT solver and the benefits of the levelwise construction are validated experimentally on the SMT-LIB benchmark library. We also compare several heuristics for the construction and observe that each heuristic has strengths offering potential for further exploitation of the new approach.

1. Introduction

In this paper we present a new method to construct around a sample point a single cylindrical cell that is truth-invariant for a set of polynomial constraints. We demonstrate how the new method allows for improved decision procedures to determine the satisfiability of formulae involving such constraints. We use a proof system presentation for our method, which we consider an important contribution for such algebraic decision procedures. We take this opportunity to explain the benefit of such a presentation to the symbolic computation community. This introduction continues with a broad overview of context for the contribution, followed by the plan of the paper.

1.1. Non-linear real arithmetic and CAD

We are concerned with *non-linear real arithmetic* whose formulae are Boolean combinations of polynomial constraints with rational coefficients. This is a powerful logic that can express a wide variety of problems. This logic admits quantifier elimination (Tarski, 1948), i.e. any quantified formula in the logic may be replaced by an equivalent quantifier-free one. In this paper we restrict our attention to the problem of determining the satisfiability of quantifier-free formulae, or equivalently, determining the truth of purely existentially quantified formulae.

The most commonly used complete methods here are based on the idea of the *cylindrical algebraic decomposition (CAD)* introduced by Collins (1975). A CAD is a finite decomposition of $\mathbb{R}^n$ into cells, traditionally produced relative to a set of polynomials in n variables such that each polynomial has constant sign on each cell. It thus allows us to use a *finite set* of sample points (one for each cell) to study sign-constraints on those polynomials over the *infinite* space $\mathbb{R}^n$. The CAD method offered the first tractable approach to real quantifier elimination and found numerous applications in the years that followed. However, its practical use is restricted by a doubly exponential worst case complexity in the number of variables (Davenport and Heintz, 1988), that is felt often in practice: the algorithm makes use of iterated resultants (Collins, 1975) leading to polynomials of doubly-exponential degree.

It was soon realized that a CAD encoded far more information than needed even for quantifier elimination: a CAD for a set of polynomials can be used to study *any* logical formula built from those polynomials, not just the one of interest. Some progress was since made in adapting the core CAD algorithm to the logical formula, e.g. Collins and Hong (1991); England et al. (2020), but these were only partial solutions.

1.2. NLSAT, MCSAT and single cylindrical cells

A novel framework for *satisfiability modulo theories (SMT)* solving was introduced with the *NLSAT algorithm* of Jovanović and de Moura (2012) in 2012. This was since generalized into the *model constructing satisfiability calculus (MCSAT)* framework (de Moura and Jovanović, 2013) and has been applied to other logics such as non-linear integer arithmetic (Jovanović, 2017). In MCSAT the search at the Boolean and theory levels are carried out concurrently, mutually guided by each other away from unsatisfiable regions. Partial solution candidates for the Boolean structure and for the corresponding theory constraints are constructed incrementally in parallel, with Boolean conflicts generalized using propositional resolution and, for real algebra, theory conflicts generalized by CAD technology.

For the latter, when a theory model (sample point) is determined not to satisfy all those constraints which should hold according to the current Boolean search, then we seek to guide the future search by an *explanation* which generalizes the sample point to a region containing the point on which the same constraints fail for the same reasons. This can be achieved by having the polynomials involved in the combination of constraints which cause the failure all have invariant sign upon this region. Such regions are constructed as cylindrical algebraic cells, but they are not necessarily cells from the CAD that would be built for the problem. Instead, they are usually larger since not all constraints are involved in every conflict. The exclusion of the cell is learned by adding a new clause: the negation of the semi-algebraic description of the cell.

This motivates the optimization of sub-algorithms to produce single cells from a point and a set of polynomial constraints. Savings can be made not just by building cells with only a subset of the constraints, but also by restricting the combinations of those constraints that we do consider in reference to the current model. Until now, the state-of-the-art approach is that of Brown and Košta (2015). We continue this research in the present paper by *developing a new method for single cell construction*.

1.3. First contribution: proof system presentation

In contrast to Brown and Košta (2015), our new method allows for different choices of how to construct the cell. We will describe and evaluate some of these choices, however, it is important to distinguish that area of work from the broader method to build the cell introduced in the next subsection. The choices do not affect the correctness of the cell produced: in all cases the cell meets the essential criteria of containing the sample and being invariant for the truth of the conflicting constraints. Nor do the choices have an effect on high-level measures of complexity: they achieve similar reductions in algebraic work. However, it can be observed that the choices do greatly effect the cells produced and thus the performance of the algorithm and so it is worth to try to make an optimal choice. We do these choices *heuristically*, i.e. using methods not guaranteed to give an optimal answer but hopefully giving a reasonable answer quickly. To expedite and simplify future research on heuristics it is helpful to clearly separate out these heuristic choices from the broader algorithm and its proof of correctness.

To achieve that, we present our work as a *proof system*. Such a presentation clearly achieves the separation of heuristic decisions from correctness proof of the method. Essentially, we must find a chain of proof rules to prove our desired property, and if there is freedom in how the chain can be built then we can employ a heuristic method. The system is flexible, extensible, allows for detailed optimizations without changing the fundamental algorithm, and allows for correctness proofs to be portioned nicely. We plan to build on that in future work. We note that this is not just a presentation for the purpose of the paper, but also present in the underlying implementation we report on.

We acknowledge that a proof system presentation is uncommon in symbolic computation. It is more prevalent in the SAT and SMT communities where there is more intense work on the optimization of such heuristic choices and proof systems are an established presentation method. However, such a system has not been used before for CAD theory, even when deployed in the SMT context. We view our proof system presentation as a contribution in its own right, which allowed for greater exploration of heuristic choices in our work. We also hope the wider symbolic computation community may find it interesting as a potential new tool to use elsewhere. In particular, there is increasing interest in formal proofs. While SMT solvers are now able to generate these proofs for a variety of theories (Barbosa et al., 2022), the case of non-linear arithmetic is challenging. Our proof system might pave the way for mechanically-verifiable proofs for non-linear real arithmetic.

1.4. Second contribution: levelwise single cells

The current state-of-the-art for single cylindrical cell construction is that of Brown and Košta (2015). This constructs the single cell gradually, processing one polynomial at a time, initializing the cell as the entirety of $\mathbb{R}^n$ and then gradually refining it according to the sign of each polynomial considered. For each refinement the method needs to consider only the interaction of the next polynomial with the ones currently defining the cell, rather than all those that went before, which allows for savings compared to the original approach used by Jovanović and de Moura (2012). However, this approach introduces a sensitivity to the order in which polynomials are considered. A machine learning approach to select the order was considered by Brown and Daves (2020).

The alternative method contributed in this paper allows to produce the single cell incrementally *by level* (i.e. dimension / variable). This removed the direct sensitivity to the polynomial ordering, instead introducing at each level decisions about which polynomials to use first. This approach uncovers a greater range of decisions than the polynomial ordering, and allows for more reasoned heuristics than black-box machine learning. By allowing for these better heuristic decisions we can produce more optimal cells, in turn improving the performance of algorithms which use them.

1.5. Plan of the paper

We continue in Section 2 by introducing the necessary preliminaries and notations used, followed by background material on CAD. Then in Section 3 we present the existing state-of-the-art in single cell construction and an informal motivation for our new levelwise approach.

In Section 4 we establish the proof system, and in Section 5 we present our new algorithm and some heuristics that may be used with it. In Section 6 we give some qualitative analysis on our new method and the heuristics. Then an experimental evaluation on the use of the new method for explanation generation in MCSAT is given in Section 7. Finally, we conclude in Section 8 with an outlook on further research and open questions.

2. Preliminaries

Let $\mathbb{N}$ denote the set of all natural numbers including 0, $\mathbb{N}_{>0} = \mathbb{N} \setminus \{0\}$, $\mathbb{Q}$ be the rational numbers, and $\mathbb{R}$ be the real numbers. For $i, j \in \mathbb{N}$ with $i < j$, we define the sets of integers $[i..j] = \{i, \dots, j\}$ and $[i] = [0..i]$. For $i, j \in \mathbb{N}_{>0}$, $j \leq i$ and $r \in \mathbb{R}^i$, we denote by r_j the j-th component of r and by $r_{[j]}$ the vector $(r_1, \dots, r_j)$. For a tuple $t = (a, b, c, \dots)$ we denote by $t.a, t.b, t.c, \dots$ the corresponding tuple entries.

Let $f : D \to E$ be a function, then the *domain* D of f is denoted by $\mathrm{dom}(f)$, and the *restriction of f to $A \subseteq D$* is denoted by $f|_A$ (i.e. $f|_A : A \to E$ with $f|_A(a) = f(a)$ for all $a \in A$). Let $f, g : D \to E$ and let $<$ be a total order on E. We write $f < g$ if $f(d) < g(d)$ for all $d \in D$ and $f \leq g$ if $f(d) \leq g(d)$ for all $d \in D$.

2.1. Variables and polynomials

We assume that the reader is familiar with the common definitions and terminology related to polynomials. We introduce some notation in this section; for further reading we refer to Cox et al. (2006).

We work with the *variables* $x_1, \dots, x_n$ with $n \in \mathbb{N}_{>0}$ under a fixed *ordering* $x_1 \prec x_2 \prec \ldots \prec x_n$. A *polynomial* is built from a set of variables and numbers from $\mathbb{Q}$ using addition and multiplication.

We use $\mathbb{Q}[y]$ to denote *univariate* polynomials in some variable y and $\mathbb{Q}[x_1, \dots, x_i]$ for *multivariate* polynomials in those variables. We say that a polynomial p is of level j (denoted as $\mathrm{level}(p) = j$) if x_j is the largest variable appearing in p: i.e. either $j = 0$ and $p \in \mathbb{Q}$; or $j \in [1..n]$ and $p \in \mathbb{Q}[x_1, \dots, x_j] \setminus \mathbb{Q}[x_1, \dots, x_{j-1}]$.

Assume in the following some $i \in [n]$ and polynomials $p, q \in \mathbb{Q}[x_1, \dots, x_i]$.

We write $p(x_1, \dots, x_i)$ to indicate p's variable domain. For $j \in [1..i]$ and $r = (r_1, \dots, r_j) \in \mathbb{R}^j$ we write $p(r, x_{j+1}, \dots, x_i)$ for the polynomial p after substituting $r_1, \dots, r_j$ for $x_1, \dots, x_j$ in p and indicating the remaining free variables in p.

We use $\mathrm{realRoots}(p) \subseteq \mathbb{R}^i$ to denote the set of *real roots* of p, $\deg_{x_j}(p)$ to denote the *degree* of p in x_j, $\mathrm{ldcf}_{x_j}(p)$ the *leading coefficient* of p in x_j, $\mathrm{factors}(p)$ to denote the *irreducible factors* of p, $\mathrm{disc}_{x_j}(p)$ to denote the *discriminant* of p with respect to x_j, and $\mathrm{res}_{x_j}(p, q)$ to denote the *resultant* of p and q with respect to x_j.

2.2. Real algebraic numbers, constraints and cells

Real algebraic numbers are real roots of univariate polynomials with rational coefficients. Although we will not distinguish between real and real algebraic numbers in the following for simplicity, the algorithms are complete when restricting all choices of constants to real algebraic numbers.

A *constraint* $p \sim 0$ compares a polynomial $p \in \mathbb{Q}[x_1, \dots, x_i]$, $\mathrm{level}(p) = i$ to zero using a relation symbol, $\sim \in \{=, \neq, <, >, \leq, \geq\}$, and has the *solution set* $\{r \in \mathbb{R}^i \mid p(r) \sim 0\}$.

A subset of $\mathbb{R}^i$ for some $i \in [n]$ is called *semi-algebraic* if it is the solution set of a Boolean combination of polynomial constraints. A *cell* is a non-empty connected subset of $\mathbb{R}^i$ for some $i \in [n]$. A cell is called *algebraic* if it is a semi-algebraic set.

For simplifying the notation throughout this paper, we define $\mathbb{R}^0 = \{()\}$. Given $i, j \in \mathbb{N}_{>0}$ with $j < i$, we call $\mathbb{R}^i$ an *extension* of $\mathbb{R}^j$ and define the *projection* of a set $R \subseteq \mathbb{R}^i$ onto $\mathbb{R}^j$ by $R{\downarrow_{[j]}} = \{(r_1, \dots, r_j) \mid (r_1, \dots, r_i) \in R\}$.

Given a cell $R \subseteq \mathbb{R}^i$, $i \in [1..n]$ and continuous functions $f, g : R \to \mathbb{R}$, we define the sets $R \times (f, g) = \{(r, r_{i+1}) \mid r \in R, r_{i+1} \in (f(r), g(r))\}$, analogously $R \times (-\infty, g) = \{(r, r_{i+1}) \mid r \in R, r_{i+1} < g(r)\}$, $R \times (f, \infty) = \{(r, r_{i+1}) \mid r \in R, f(r) < r_{i+1}\}$, and $R \times f = \{(r, r_{i+1}) \mid r \in R, r_{i+1} = f(r)\}$. Note that if $f \leq g$ (on R), then these sets are cells (as a continuous image of a connected set is connected).

The *sign* of $r \in \mathbb{R}$, denoted $\mathrm{sgn}(r)$, is defined to be 1 if $r > 0$, -1 if $r < 0$, and 0 otherwise. A polynomial $p \in \mathbb{Q}[x_1, \dots, x_i]$ is *sign-invariant* on a set $R \subseteq \mathbb{R}^i$ if $\mathrm{sgn}(p(r)) = \mathrm{sgn}(p(r'))$ for all $r, r' \in R$. A set of polynomials $P \subseteq \mathbb{Q}[x_1, \dots, x_i]$ is *sign-invariant* on $R \subseteq \mathbb{R}^i$ if all $p \in P$ are sign-invariant on R.

2.3. CAD definition

A set $D = \{R_1, \dots, R_k\}$ of cells of $\mathbb{R}^n$ such that $\cup_{i=1,\dots,k} R_i = \mathbb{R}^n$ and $R_i \cap R_j = \emptyset$ is called a *decomposition* of $\mathbb{R}^n$. A decomposition is called *algebraic* if its cells are algebraic. A decomposition D of $\mathbb{R}^n$ is called *cylindrical* over a decomposition D' of $\mathbb{R}^m$, $m < n$ if all projections of cells $R \in D$ onto $\mathbb{R}^m$ are themselves cells in D'. I.e. the cells in $\mathbb{R}^n$ stack up in cylinders over the cells in $\mathbb{R}^m$. A *cylindrical algebraic decomposition (CAD)* is produced relative to a variable ordering: it is an algebraic decomposition D such that there exists a sequence of algebraic decompositions $(D_1, \dots, D_n)$, $D = D_n$ with each D_i a cylindrical decomposition of $\mathbb{R}^i$ over D_{i-1} for $i \in [2..n]$.

A decomposition inherits the invariance properties for polynomials and constraints defined above if they apply to all its cells. A CAD will usually be computed relative to an input polynomial set to ensure such an invariance property. Collins (1975) first introduced the notion of a CAD and an algorithm for computing a sign-invariant CAD. A central notion for this algorithm is delineability.

Definition 2.1 *(Delineability (Collins, 1975)).* Let $i \in \mathbb{N}$, $R \subseteq \mathbb{R}^i$ be a cell, and $p \in \mathbb{Q}[x_1, \dots, x_{i+1}] \setminus \{0\}$. The polynomial p is called *delineable* on R if and only if there exist finitely many continuous functions $\theta_1, \dots, \theta_k : R \to \mathbb{R}$ (for $k \geq 0$) such that

- $\theta_1 < \ldots < \theta_k$;
- the set of real roots of the univariate polynomial $p(r, x_{i+1})$ is $\{\theta_1(r), \dots, \theta_k(r)\}$ for all $r \in R$; and

- there exist constants $m_1, \ldots, m_k \in \mathbb{N}_{>0}$ such that for all $r \in R$ and all $j \in [1..k]$, the multiplicity of the root $\theta_j(r)$ of $p(r, x_{i+1})$ is m_j.

The θ_j are called *real root functions* of p on R. The cells $R \times \theta_j,\ j \in [1..k]$ are called *p-sections* over R. The cells $R \times (-\infty, \theta_1), R \times (\theta_k, \infty)$, and $R \times (\theta_j, \theta_{j+1})$ for $j \in [1..k-1]$, and in case $k = 0$ also $R \times (-\infty, \infty)$, are called *$p$-sectors* over R.

These notions are extended to finite sets of polynomials $P \subseteq \mathbb{Q}[x_1, \ldots, x_{i+1}] \setminus \{0\}$ such that P is delineable on R if the product of the polynomials in P is delineable on R. Accordingly, we define the real root functions of P, the P-sections and P-sectors; for the empty polynomial set there are no sections and a single sector $\mathbb{R}^{i+1}$.

A *CAD projection operator* is a function proj that maps a set of polynomials P to a set of lower-level polynomials such that the sign-invariance of proj(P) on R implies the delineability of P on R. The operator proj induces a *sign-invariant CAD of P*, recursively defined as follows. In the case where all polynomials in P are of level 1, then the CAD is the set of all sections and sectors of P. If the polynomials are of higher level, then the CAD contains all sections and sectors of P over each cell of the CAD of proj(P).

2.4. McCallum's projection operator

Although Collin's original projection operator is complete, the projection set is large and thus relatively inefficient. McCallum (1998) presented an improved operator by making the projection set smaller. To do so, his proof of correctness relies not on sign-invariance but on the stronger property of *order-invariance*. Although a stronger property, the induced cells in McCallum's CAD are actually bigger than in Collin's CAD. In the following, we present a simplified version of the McCallum CAD projection (simplified as we do not describe the optimization using *delineating polynomials*).

McCallum's theory relies on some notions which we will mention here only on the level of intuition: for more details, we refer to McCallum (1985, 1998). An i-dimensional *(analytic) submanifold* of $\mathbb{R}^n$ is a non-empty subset $R \subseteq \mathbb{R}^n$ that "looks locally like $\mathbb{R}^i$". Given an open subset $U \subseteq \mathbb{R}^i$, a function $f : U \to \mathbb{R}$ is called *analytic* if it has a multiple power series representation (McCallum, 1985) around each point of U. Given an i-dimensional submanifold R of $\mathbb{R}^n$, a function $f : R \to \mathbb{R}$ is called *analytic* if for all $r \in R$, R looks locally like $\mathbb{R}^i$ with respect to a coordinate system about r and f looks locally like an analytic function $\mathbb{R}^i \to \mathbb{R}$. Every open subset of $\mathbb{R}^i, i \in [1..n]$ is an analytic submanifold. To simplify notation, we say $\mathbb{R}^0$ is an analytic submanifold. Given an analytic submanifold $R \subseteq \mathbb{R}^i$ and analytic functions $f, g : R \to \mathbb{R}$ with $f < g$, the sets $R \times (f, g)$ and $R \times f$ are analytic submanifolds as well (McCallum, 1985, Theorem 2.2.3 and Theorem 2.2.4). Note that analytic submanifolds are cells. Additionally, the notion of delineability is extended to *analytic delineability*, which is only defined on connected analytic submanifolds and the real root functions are required to be analytic.

Let $p \in \mathbb{Q}[x_1, \ldots, x_n]$ be a polynomial and $r \in \mathbb{R}^n$ be a point. Then the *order of p at r* is defined as

$$\operatorname{ord}_r(p) = \min(\{k \in \mathbb{N} \mid \text{some partial derivative of total order } k \text{ of } p \text{ does not vanish at } r\} \\ \cup \{\infty\}).$$

We call p *order-invariant* on $R \subseteq \mathbb{R}^n$ if $\operatorname{ord}_r(p) = \operatorname{ord}_{r'}(p)$ for all $r, r' \in R$. Note that if p has no root in R, then $\operatorname{ord}_r(p) = 0$ for all $r \in R$ and p is trivially order-invariant on R. Note also that order-invariance implies sign-invariance.

McCallum's operator requires a smaller projection set than Collins' operator. It does so by maintaining the stronger property of order-invariance instead of sign-invariance; however, order-invariance can only be concluded if no polynomial is *nullified* on a point in the underlying cell.

Definition 2.2 *(Nullification).* Let $i \in \mathbb{N}$, $r \in \mathbb{R}^i$, and $p \in \mathbb{Q}[x_1, \ldots, x_{i+1}]$, level$(p) = i + 1$. The polynomial p *is nullified* on r if $p(r, x_{i+1}) = 0$.

The McCallum projection operator is thus incomplete. We note the recent validation of *Lazard projection* by McCallum et al. (2019) as an alternative to McCallum projection which is complete and is no bigger than McCallum except for those cases where McCallum cannot be applied (Brown and McCallum, 2020). We choose to formalize here with McCallum projection as it is extended to optimizations such as *equational constraints* (McCallum, 1999; England et al., 2020) which are very powerful in practice (see Section 4.6) and the existing single cell approach (Brown and Košta, 2015) is also based on McCallum projection. We expect that our work could be reformulated in Lazard projection if so desired.

2.5. Computing a CAD

In many applications, a decomposition of a set of polynomials P is not computed explicitly but instead the projection of P is used to generate sample points for every sign-invariant cell that would be formed in such a decomposition. The generation of cells / samples from the projection is called *lifting*. Polynomials can be evaluated at these sample points to check the satisfiability of constraints and formulae which involve them.

For representing cells explicitly, we need to give witnesses for the real root functions $\theta_j : R \to \mathbb{R}$ from Definition 2.1 defining the sectors and sections over a cell R.

Definition 2.3 *(Indexed root expression).* Let $i \in \mathbb{N}$, $p \in \mathbb{Q}[x_1, \ldots, x_{i+1}]$, $\mathrm{level}(p) = i+1$, and $j \in \mathbb{N}_{>0}$. The *indexed root expression* $\mathrm{root}_{x_{i+1}}[p, j] : \mathbb{R}^i \to \mathbb{R} \cup \{\mathit{undef}\}$ is the j-th real root of p in x_{i+1} at the given sample if it exists, and *undef* otherwise. That is for each $s \in \mathbb{R}^i$:

$$\mathrm{root}_{x_{i+1}}[p, j](s) = \begin{cases} \mathit{undef} & \text{if } j > |\mathrm{realRoots}(p(s, x_{i+1}))| \textit{ or } p(s, x_{i+1}) = 0, \textit{ and otherwise} \\ \xi_j & \text{where } \mathrm{realRoots}(p(s, x_{i+1})) = \{\xi_1, \ldots, \xi_k\} \textit{ and } \xi_1 < \ldots < \xi_k. \end{cases}$$

Indexed root expressions of this form are also called *indexed root expression of level* $i+1$. Assuming ξ denotes the above indexed root expression, we use $\xi.p$ to refer to p and $\xi.j$ to refer to j.

In the algorithms presented in this paper, we only need to evaluate the indexed root expressions for real algebraic numbers. Note that the existence and index of a real root function depends on the given sample. Thus, the same indexed root expression may refer to different real root functions at different sample points.

Definition 2.4 *(Symbolic intervals, single cell and CAD data structures).* A *symbolic interval* $\mathtt{I}$ *of level* i is either (i) of the form $\mathtt{I} = (\mathrm{sector}, l, u)$ where l is either an indexed root expression of level i or $-\infty$, and u is either an indexed root expression of level i or ∞; or (ii) of the form $\mathtt{I}_i = (\mathrm{section}, b)$ where b is an indexed root expression of level i. The polynomials $\mathtt{I}.l$, $\mathtt{I}.u$ respectively $\mathtt{I}.b$ are the *defining polynomials of* $\mathtt{I}$ (if they exist).

A *cell data structure* is a sequence $\mathtt{R} = (\mathtt{I}_1, \ldots, \mathtt{I}_n)$ of symbolic intervals $\mathtt{I}_i$ of level i. For an empty cell data structure, we define $\mathit{setOf}(()) = \{()\}$. For a cell data structure $(\mathtt{I}_1, \ldots, \mathtt{I}_i)$, $i \geq 1$, we define the corresponding subset of $\mathbb{R}^i$ as $\mathit{setOf}(\mathtt{I}_1, \ldots, \mathtt{I}_i) = \{(r, r') \mid r \in \mathit{setOf}(\mathtt{I}_1, \ldots, \mathtt{I}_{i-1}),\ r' \in (\mathtt{I}_i.l(r), \mathtt{I}_i.u(r))\}$, respectively $\mathit{setOf}(\mathtt{I}_1, \ldots, \mathtt{I}_i) = \{(r, r') \mid r \in \mathit{setOf}(\mathtt{I}_1, \ldots, \mathtt{I}_{i-1}),\ r' = \mathtt{I}_i.b(r)\}$ if the respective indexed root expressions are not *undef*, and $\mathit{setOf}(\mathtt{I}_1, \ldots, \mathtt{I}_i) = \mathit{undef}$ otherwise. Similarly, we define $\mathit{setOf}(R, \mathtt{I}_i)$ for arbitrary subsets $R \subseteq \mathbb{R}^{i-1}$ and $\mathit{setOf}(r, \mathtt{I}_i)$ for points $r \in \mathbb{R}^{i-1}$.

A *CAD data structure* $\mathtt{D}$ is a set of cell data structures. We define $\mathit{setOf}(\mathtt{D}) = \{\mathit{setOf}(\mathtt{R}) \mid \mathtt{R} \in \mathtt{D}\}$.

In our algorithms, indexed root expressions occurring in a cell data structure will always be defined; the above definition covers the *undef* case just for the sake of completeness. Furthermore, note that given a cell data structure $\mathtt{R} = (\mathtt{I}_1, \ldots, \mathtt{I}_i)$, the restrictions of members $\mathtt{I}_i.l|_{\mathit{setOf}(\mathtt{I}_1,\ldots,\mathtt{I}_{i-1})}$, $\mathtt{I}_i.u|_{\mathit{setOf}(\mathtt{I}_1,\ldots,\mathtt{I}_{i-1})}$, $\mathtt{I}_i.b|_{\mathit{setOf}(\mathtt{I}_1,\ldots,\mathtt{I}_{i-1})}$ are real root functions of their defining polynomials as θ_j in Definition 2.1.

Definition 2.5 *(Indexed root expressions of polynomials).* Let $i \in \mathbb{N}$, $s \in \mathbb{R}^i$, and $p \in \mathbb{Q}[x_1, \dots, x_{i+1}]$ of level $i+1$. The *set of indexed root expressions of p at s* is defined as

$$\mathrm{irExpr}(p, s) = \begin{cases} \mathit{undef} & p(s, x_{i+1}) = 0, \\ \{\mathrm{root}_{x_{i+1}}[p, j] \mid j \in [1..|\,\mathrm{realRoots}(p(s, x_{i+1}))|]\} & \text{otherwise.} \end{cases}$$

Let $P \subseteq \mathbb{Q}[x_1, \dots, x_{i+1}]$ be a set of polynomials of level $i+1$. The *set of indexed root expressions of P at s* is defined as

$$\mathrm{irExpr}(P, s) = \bigcup_{p \in P} \mathrm{irExpr}(p, s).$$

Let $\xi \in \mathbb{R}$. The *set of indexed root expressions of P for s and* ξ is defined as

$$\mathrm{irExpr}(P, s, \xi) = \{\mathrm{root}_{x_{i+1}}[p, j] \mid p \in P,\ j \in \mathbb{N}_{>0},\ \xi = \mathrm{root}_{x_{i+1}}[p, j](s)\}.$$

A description of a sign-invariant CAD defined by a projection operator with respect to a set of polynomials can be computed as follows. First the projection operator is applied from level n to 1, called the *projection phase*. This is followed by the *lifting phase* where, starting from level 1, all cells from level $i-1$ are extended to the level i such that all sections and sectors in the cylinder above every cell of level $i-1$ are identified as cells. To achieve this, for each cell $\mathtt{R}$ of dimension $i-1$, the polynomials on level i are partially evaluated up to their last dimension using a sample s from $\mathtt{R}$. This results in univariate polynomials whose roots can be isolated and sorted to give intervals: point intervals at the roots of the polynomials, the open intervals between them, and the two open intervals below and above all roots (or the whole real line if there are no roots). Delineability allows the conclusions drawn at the sample to generalize over $\mathtt{R}$. For each interval a symbolic description is determined that together extend $\mathtt{R}$ to level i.

3. Single cell computation

For our problem of study, we are not interested in computing a full sign-invariant CAD, but only a single sign-invariant algebraic cell. That is, given a set of polynomials $P \subseteq \mathbb{Q}[x_1, \dots, x_n]$ and a sample $s \in \mathbb{R}^n$, we need to compute a cell $R \subseteq \mathbb{R}^n$ such that $s \in R$ and P is sign-invariant on R.

In this paper, we focus on the computation of algebraic cells that adhere to Definition 2.4. These cells have a triangular algebraic description with respect to the variable ordering (i.e. a condition on x_1 alone; then one on (x_1, x_2), and so on). Such cells are called *(locally) cylindrical* (Brown and Košta, 2015; Ábrahám et al., 2021) as this shape is implied by the cylindricity property of a full CAD. Although we do not construct entire decompositions in this paper, we make use of CAD theory which results in cells having this property. Such cells have the advantage of being easy to visualize and compute with. For example, projection with respect to the variable ordering is trivial, as is checking whether a point is inside such a cell, which is particularly important for our use of such cells.

3.1. Previous work on optimizing single cell computations

As described in Section 1.2, Jovanović and de Moura use a single cell construction for model explanation in their algorithm to decide satisfiability for non-linear arithmetic (Jovanović and de Moura, 2012). This was based on Collins' CAD projection operator (Collins, 1975) but did not compute a full CAD projection: it left out coefficients based on the current sample (as we only need to consider subsequent coefficients when the leading coefficient vanishes).

This single cell construction was enhanced, first in the open case (i.e. building only cells with maximal dimension) by Brown (2013), and then for the general case by Brown and Košta (2015). This work was based on McCallum projection theory, and in addition to coefficients also avoided some computation of resultants and discriminants based on the current sample. The work led in turn to the consideration of entire decompositions built one cell at a time (Brown, 2015), and their use for quantifier elimination (Brown, 2017).

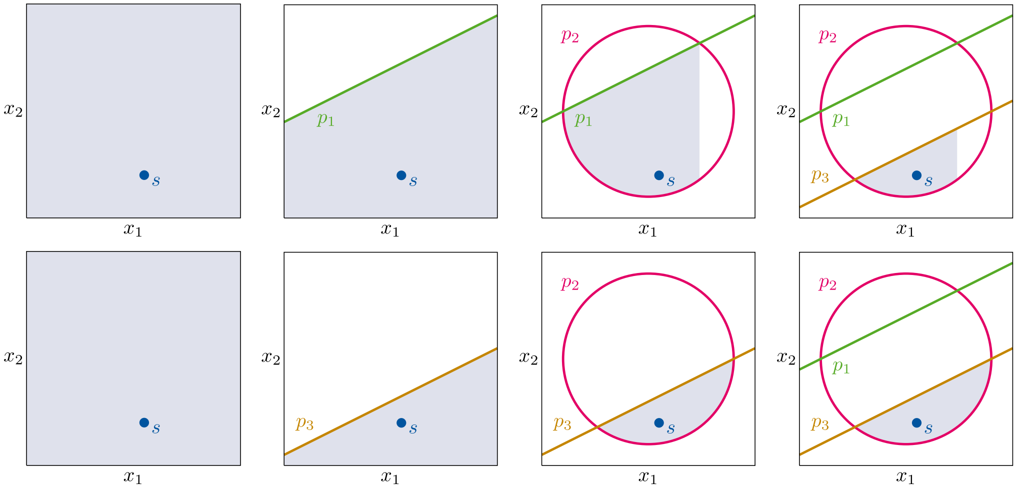


Fig. 1. Refinement-based construction of a single cell for Example 3.1. In the upper row the original cell (the entire plane) is refined first by polynomial p_1, then by p_2, then by p_3. In the lower row a different order of refinement is used: p_3, p_2, p_1 giving a larger cell. Note that in the figures of this paper we label the varieties of polynomials with their name, i.e. p_1 labels $p_1 = 0$.

The approach of Brown (2015) started with a cell data structure describing the whole of $\mathbb{R}^n$ which was continually refined by *merging* in new polynomials one at a time. The process maintains several invariants: (1) the cell contains the sample point s; (2) the cell is cylindrical; and (3) all polynomials merged so far have constant sign (more precisely, constant order) in the cell. The polynomials are merged one-by-one, and the order in which they are merged affects the cell that gets produced. Fig. 1 illustrates this process for Example 3.1 below, showing different results for two different orderings of the polynomials. From now on we will refer to this approach as *refinement-based single cell construction*. The merge operation, unless on the lowest level, projects the current polynomial p and then calls itself iteratively on the projection result. When the call returns to p, the polynomial is used to *refine the bounds* of the cell in the dimension that corresponds to the level of p. This is done by isolating the roots of the univariate polynomial resulting from partially evaluating p up to its last dimension using the sample s. If there is a root closer to s_i than the current bound then it becomes the new bound of the *sector*. In the special case that a root coincides with s_i, the sector collapses into a *section* described by said root. By this recursive refinement, the transitivity of the ordering on real root functions of polynomials induced by the order-invariance of resultants is exploited, so that at most two resultants are calculated per merge-operation and polynomial. Furthermore, as soon as a section is identified, some superfluous discriminants can be detected, and using the current sample irrelevant coefficients can be identified.

Example 3.1. We consider the point $s = (\frac{1}{8}, -\frac{3}{4})$ and polynomials $p_1 = x_1 - 2x_2 + 1$, $p_2 = x_1^2 + x_2^2 - 1$ and $p_3 = x_1 - 2x_2 - 1$. These are the polynomials studied in Fig. 1 which demonstrates the refinement-based single cell construction process for two different orderings. We see that one produces a larger cell than the other. It is not obvious how to choose good orders for adding polynomials in this approach (a machine learning heuristic was presented by Brown and Daves (2020)). Moreover, because the method works by refining the cell by the chosen polynomial, the order in which lower-level polynomials resulting from projections are added is not very flexible.

Our approach will build a single cell levelwise, i.e. constructing the cell level-by-level according to a variable ordering instead of polynomial-by-polynomial for all levels. We note that there exists another *levelwise* single cell construction in an unpublished work on the arXiv by Li and Xia (2020). In

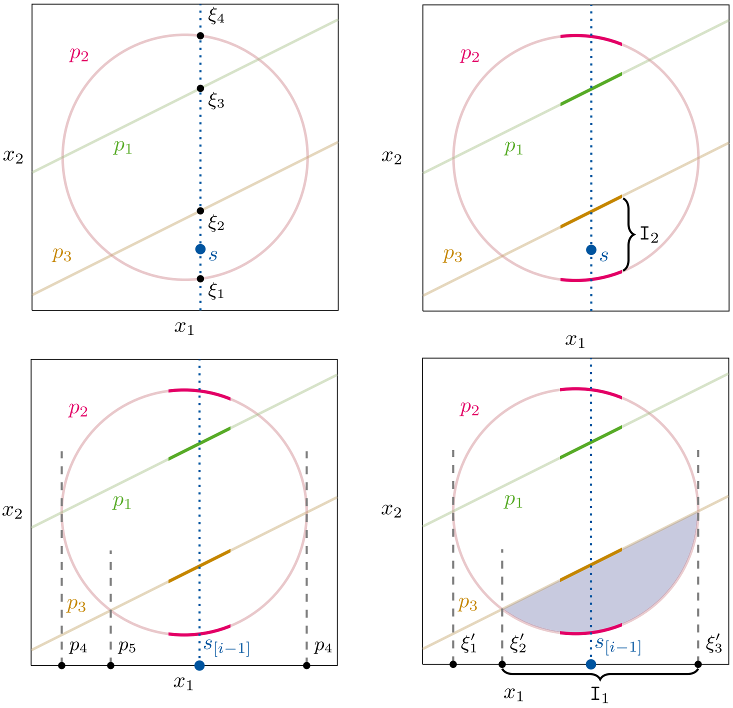


Fig. 2. Construction of a single cell for Example 3.1 following the new levelwise approach as described in Example 3.2.

comparison, our formulation using a proof system enables the use of more sophisticated optimizations in the later sections.

3.2. Our levelwise approach to single cell computation

Our new *levelwise* approach is based on using information about the ordering of the roots of polynomials relative to the sample point s and to one another to make decisions during the cell construction process that are likely to lead to larger cells.

We will exploit said information and the transitivity of the ordering on the roots induced by order-invariance of resultants. Thus, before projection, the roots of the polynomials on the current level are isolated and ordered to see which resultants are required for maintaining sign- or order-invariance of the given polynomials.

Example 3.2. Fig. 2 shows how our new levelwise approach would operate on the formula from Example 3.1. We start by considering $x_1 = s_1$, the first coordinate of the sample s. We evaluate the polynomials at this coordinate and isolate the real roots to find $\xi_1 = \mathrm{root}_{x_2}[p_2, 1]$, $\xi_2 = \mathrm{root}_{x_2}[p_3, 1]$, $\xi_3 = \mathrm{root}_{x_2}[p_1, 1]$, and $\xi_4 = \mathrm{root}_{x_2}[p_2, 2]$. We order these, along with the second coordinate of s, as in the top-left image.

Our sample lies in the interval for x_2 denoted $\mathtt{I}_2 = (\mathrm{sector}, \xi_1, \xi_2)$ in the top-right image. We thus know that, local to s_1, the cell we want will be bounded from below by p_2 and from above by p_3. We also know that the other section of p_2 and the section of p_1 are above the cell. In this figure the graphs of the polynomials are greyed out because the algorithm is not aware of them: it knows only their roots when evaluated at the sample. However, this is enough to allow the algorithm to infer behaviour around the sample, as illustrated by the thick partial lines.

As we generalize from the sample we must ensure than ξ_1 and ξ_2 remain well-defined and that no root function crosses the symbolic interval described by $\mathtt{I}_2$. To achieve this we compute a projection consisting of $p_4 = \mathrm{disc}_{x_2}(p_2)$ and $p_5 = \mathrm{res}_{x_2}(p_3, p_2)$ (also $\mathrm{disc}_{x_2}(p_1)$ and $\mathrm{disc}_{x_2}(p_3)$ but they do not have any real roots and so play no further role in this example). Crucially, because p_1 has only one section and that section lies above the cell of interest, the resultant of p_1 and the lower-boundary polynomial p_2 is not required or computed. The bottom-left figure shows how the zeros of the projection map onto the geometric features.

Analogously to x_2, we isolate the real roots $\xi_1' = \mathrm{root}_{x_1}[p_4, 1], \xi_2' = \mathrm{root}_{x_1}[p_5, 1], \xi_3' = \mathrm{root}_{x_1}[p_4, 2]$ in x_1. We determine the interval $\mathtt{I}_1 = (\mathrm{sector}, \xi_2', \xi_3')$ around s_1 for x_1 and thus generate the cell in the bottom-right image.

So we see that for Example 3.2 the new levelwise approach produces the second of the two possible outcomes from the refinement-based construction (Fig. 1). The example illustrates how the levelwise approach avoids “mistakes” that the refinement-based construction can make due to poor choices of orderings for the polynomials.

The levelwise approach also allows greater flexibility in how polynomials resulting from projection are handled, but that aspect of the algorithm requires more variables than in this simple example in order to be observed.

4. Proof system for single cells

In Section 5 we will present our new levelwise procedure to construct a single cell. We lay the basis for this by first formalizing our work as a proof system in this section. The system expresses how the properties we want to conclude rely on other “smaller” properties. From this, the procedure in the next section gains maximal freedom to make heuristic choices. Before we go into the formal presentation, we outline the idea for our running example.

4.1. Motivating example

Example 4.1. Recall Example 3.1 and Fig. 2 where we sought to construct a cell $R \subseteq \mathbb{R}^2$ around $s = (s_1, s_2)$ such that the sign-invariance of p_1, p_2, p_3 hold on R. We started by eliminating x_2: we isolated real roots of $p_j(s_1, x_2)$ to define $\xi_1, \dots \xi_4$, sorted them, and observed that $\xi_1(s_1) < s_2 < \xi_2(s_1)$, thus we determined that in the second level the cell was represented by $\mathtt{I}_2 = (\mathit{sector}, \xi_1, \xi_2)$.

This root ordering and the representation should now be generalized such that the underlying cell R_1 is restricted by specifying certain properties ensuring ξ_1 and ξ_2 remain as cell boundaries: they must remain well-defined over R_1; no further root of p_1, p_2, p_3 should appear that intersects $\mathtt{I}_2$ over R_1; and the other roots, ξ_3 and ξ_4, should remain outside the symbolic interval $\mathtt{I}_2$ over R_1. For the latter, we extend to a partial ordering $\preceq$ with $\xi_2 \preceq \xi_3$ and $\xi_2 \preceq \xi_4$.

To conclude that p_1, p_2, p_3 are sign-invariant in R, the proof system shows we need to maintain the properties that: the sample s is included in R; $\mathtt{I}_2$ describes the cell's boundaries on the current level; the underlying cell R_1 is a connected analytic submanifold; and p_1, p_2, p_3 are delineable. Furthermore, we maintain that the partial ordering $\preceq$ is maintained over R_1.

To maintain these properties the proof system concludes that we must also prove order-invariance and sign-invariance of some coefficients, resultants, and discriminants of the input polynomials. After simplification, we find that on level 1 the polynomial $p_4 = \mathrm{disc}_{x_2}(p_2)$ and $p_5 = \mathrm{res}_{x_2}(p_3, p_2)$ must be ensured sign-invariant. This leads to a representation $\mathtt{I}_1 = (\mathit{sector}, \xi_2', \xi_3')$ and then since all polynomials are univariate and their zeros are algebraic numbers, no further projection or proof steps are necessary. The constructed cell is described by $\mathtt{R} = (\mathtt{I}_1, \mathtt{I}_2)$.

The graph of the proof constructed is shown in Fig. 3, in which the sign-invariance properties $\mathit{sgn\_inv}(p_1)$, $\mathit{sgn\_inv}(p_2)$, $\mathit{sgn\_inv}(p_3)$ lead to $R = \mathit{setOf}(R\downarrow_{[1]}, \mathtt{I}_2)$ and $R\downarrow_{[1]} = \mathit{setOf}(\mathtt{I}_1)$. The exact proof rules used in that figure will be defined in the rest of this section.

Note how in the example there were multiple choices for $\preceq$. Choosing $\xi_2 \preceq \xi_3$ and $\xi_3 \preceq \xi_4$ would have been a valid choice but that leads to a smaller cell (the one depicted in the top right of Fig. 1). We will discuss ordering heuristics to make this choice in Section 6.1.

$$\begin{aligned}
\mathit{sgn\_inv}(p_1)(R) &\dashv \mathit{sample}(s)(R), \mathit{repr}(\mathtt{I}_2, s_1)(R), \mathit{ir\_ord}(\preceq, s_1)(R), \mathit{an\_del}(p_1)(R),\\
&\quad \mathit{an\_sub}(1)(R), \mathit{connected}(1)(R)\\
\mathit{sgn\_inv}(p_2)(R) &\dashv \mathit{sample}(s)(R), \mathit{repr}(\mathtt{I}_2, s_1)(R), \mathit{ir\_ord}(\preceq, s_1)(R), \mathit{an\_del}(p_2)(R),\\
&\quad \mathit{an\_sub}(1)(R), \mathit{connected}(1)(R)\\
\mathit{sgn\_inv}(p_3)(R) &\dashv \mathit{sample}(s)(R), \mathit{repr}(\mathtt{I}_2, s_1)(R), \mathit{ir\_ord}(\preceq, s_1)(R), \mathit{an\_del}(p_3)(R),\\
&\quad \mathit{an\_sub}(1)(R), \mathit{connected}(1)(R)\\
\mathit{sample}(s)(R) &\dashv \mathit{repr}(\mathtt{I}_2, s_1)(R), \mathit{sample}(s_1)(R)\\
\mathit{repr}(\mathtt{I}_2, s_1)(R) &\dashv R = \mathit{setOf}(R\downarrow_{[1]}, \mathtt{I}_2), \mathit{an\_del}(p_2)(R), \mathit{an\_del}(p_3)(R), \mathit{sample}(s_1)(R)\\
\mathit{ir\_ord}(\preceq, s_1)(R) &\dashv \mathit{an\_del}(p_1)(R), \mathit{an\_del}(p_2)(R), \mathit{an\_del}(p_3)(R), \mathit{ord\_inv}(\mathrm{res}_{x_2}(p_3, p_1))(R),\\
&\quad \mathit{ord\_inv}(\mathrm{res}_{x_2}(p_3, p_2))(R), \mathit{an\_sub}(1)(R), \mathit{connected}(1)(R), \mathit{sample}(s_1)(R)\\
\mathit{an\_del}(p_1)(R) &\dashv \mathit{non\_null}(p_1)(R), \mathit{ord\_inv}(\mathrm{disc}_{x_2}(p_1))(R), \mathit{sgn\_inv}(\mathrm{ldcf}_{x_2}(p_1))(R),\\
&\quad \mathit{an\_sub}(1)(R), \mathit{connected}(1)(R)\\
\mathit{an\_del}(p_2)(R) &\dashv \mathit{non\_null}(p_2)(R), \mathit{ord\_inv}(\mathrm{disc}_{x_2}(p_2))(R), \mathit{sgn\_inv}(\mathrm{ldcf}_{x_2}(p_2))(R),\\
&\quad \mathit{an\_sub}(1)(R), \mathit{connected}(1)(R)\\
\mathit{an\_del}(p_3)(R) &\dashv \mathit{non\_null}(p_3)(R), \mathit{ord\_inv}(\mathrm{disc}_{x_2}(p_3))(R), \mathit{sgn\_inv}(\mathrm{ldcf}_{x_2}(p_3))(R),\\
&\quad \mathit{an\_sub}(1)(R), \mathit{connected}(1)(R)\\
\mathit{non\_null}(p_1)(R) &\dashv \mathit{true}\\
\mathit{non\_null}(p_2)(R) &\dashv \mathit{true}\\
\mathit{non\_null}(p_3)(R) &\dashv \mathit{true}\\
\mathit{ord\_inv}(\mathrm{disc}_{x_2}(p_1))(R) &\dashv \mathit{true}\\
\mathit{ord\_inv}(\mathrm{disc}_{x_2}(p_2))(R) &\dashv \mathit{sgn\_inv}(p_4)(R), \mathit{sample}(s_1)(R)\\
\mathit{ord\_inv}(\mathrm{disc}_{x_2}(p_3))(R) &\dashv \mathit{true}\\
\mathit{sgn\_inv}(p_4)(R) &\dashv \mathit{repr}(\mathtt{I}_1)(R)\\
\mathit{ord\_inv}(\mathrm{res}_{x_2}(p_3, p_1))(R) &\dashv \mathit{true}\\
\mathit{ord\_inv}(\mathrm{res}_{x_2}(p_3, p_2))(R) &\dashv \mathit{sgn\_inv}(p_5)(R), \mathit{sample}(s_1)(R)
\end{aligned}$$

$$\begin{aligned}
\mathit{sgn\_inv}(p_5)(R) &\dashv \mathit{repr}(\mathtt{I}_1)(R)\\
\mathit{sgn\_inv}(\mathrm{ldcf}_{x_2}(p_1))(R) &\dashv \mathit{true}\\
\mathit{sgn\_inv}(\mathrm{ldcf}_{x_2}(p_2))(R) &\dashv \mathit{true}\\
\mathit{sgn\_inv}(\mathrm{ldcf}_{x_2}(p_3))(R) &\dashv \mathit{true}\\
\mathit{an\_sub}(1)(R) &\dashv \mathit{repr}(\mathtt{I}_1)(R)\\
\mathit{connected}(1)(R) &\dashv \mathit{true}\\
\mathit{sample}(s_1)(R) &\dashv \mathit{repr}(\mathtt{I}_1)(R)\\
\mathit{repr}(\mathtt{I}_1)(R) &\dashv R\downarrow_{[1]} = \mathit{setOf}(\mathtt{I}_1)
\end{aligned}$$

Fig. 3. Proof for Example 4.1 (slightly truncated, read in columns). $Q \dashv P_1, \ldots, P_k$ means that Q is derived from $P_1, \ldots, P_k$.

4.2. *Proof system: properties and rules*

Our system is for constructing cells. We will define *properties* of the cell that is being constructed and *rules of inference* to infer that a property holds for a cell given some other properties and conditions. As indicated in the previous example, there may be different rules that can be used to prove a property.

Definition 4.1 *(Property).* Let $i \in \mathbb{N}$. A *property of level* i is a function $q : \{R \mid R \subseteq \mathbb{R}^i\} \to \mathbb{B}$. A property q of level i *holds* on a set $R \subseteq \mathbb{R}^i$ if $q(R) = 1$. The set Prop denotes the set of all properties of any level. For a set $Q \subseteq$ Prop, we denote by $Q|_i$ the set of properties of level i and by $Q|_{[i]}$ the set of properties of level at most i.

Note that the concept of levels of polynomials has been extended to properties in the sense that the satisfaction of a property of level i cannot be influenced by values of the variables $x_{i+1}, \dots, x_n$.

We denote a *rule of inference* in the form $P_1, \dots, P_k \vdash Q$, where $P_1, \dots, P_k$ are the antecedents and Q is the consequent.

For convenience, we allow lifting properties from lower levels to higher levels as follows.

Definition 4.2. We extend every property q of level i to $q : \{R \mid R \subseteq \mathbb{R}^j,\ j \geq i\} \to \mathbb{B}$ such that $q(R) = q(R\downarrow_{[i]})$ for any $R \subseteq \mathbb{R}^j,\ j \geq i$. Accordingly, we introduce the following rule of inference:

$$R \subseteq \mathbb{R}^{i+1},\ q \in \mathrm{Prop}|_i,\ q(R\downarrow_{[i]}) \qquad \vdash q(R)$$

So, for example, this rule allows us to say that because the property of sign-invariance holds for $x^2 - 2$ in the one-level cell $-1 < x < 1$, sign-invariance holds for $x^2 - 2$ in the two-level cell which is the interior of the unit circle, since this projects down to $-1 < x < 1$.

The set of inference rules will be defined in such a way that the property of sign-invariance of a polynomial is reduced to the property of sign-invariance of polynomials at a lower level: thus the chain of rules is finite. On each level, we will additionally determine a *representation* describing the symbolic boundaries of the cell on this level as well as an ordering of the indexed roots of some polynomials assuring their sign-invariance which will be used later for heuristic decisions.

4.3. Basic cell properties

We will first define the basic properties on polynomials and cells that follow from the original work on McCallum CAD projection. Throughout, i will refer to the level of the given property and s refers to *algebraic* points. It is important to recall that an i-level property is a function that maps subsets of $\mathbb{R}^i$ to Boolean values. Consider for example the first element of Property 4.1: "*sample*(s)" for $s \in \mathbb{R}^i$ is a property, meaning that *sample* is a function that maps a point (and a level i that is given implicitly by the point) to a function mapping subsets of $\mathbb{R}^i$ to Booleans. Thus, *sample*(s) is not a true/false value. Rather, it is the function *sample*(s) applied to a given subset of $\mathbb{R}^i$ that yields a true/false value. Note that for the fifth and sixth items in Property 4.1 the level i must be given explicitly, while with the others the level is implicit in other arguments.

Property 4.1. *Let $i \in \mathbb{N}$, and $R \subseteq \mathbb{R}^i$.*

1. *Let $s \in \mathbb{R}^i$ be a sample point. The property sample(s) holds on R if and only if $s \in R$.*
2. *Let $p \in \mathbb{Q}[x_1, \dots, x_i]$ be a polynomial of level i. The property ord_inv(p) holds on R if and only if p is order-invariant on R.*
3. *Let $p \in \mathbb{Q}[x_1, \dots, x_i]$ be a polynomial of level i. The property sgn_inv(p) holds on R if and only if p is sign-invariant on R.*
4. *Let $p \in \mathbb{Q}[x_1, \dots, x_{i+1}]$ be a polynomial of level $i+1$. The property non_null(p) holds on R if and only if p is not nullified on any point in R.*
5. *The property an_sub(i) holds on R if and only if R is an analytic submanifold.*
6. *The property connected(i) holds on R if and only if R is connected.*
7. *Let $p \in \mathbb{Q}[x_1, \dots, x_{i+1}]$ be a polynomial of level $i+1$. The property an_del(p) holds on R if and only if p is analytically delineable on some connected superset $R' \supseteq R$ and p is order-invariant in all p-sections over R'.*

Note that *an_del*(p) is defined such that it holds on a subset $R \subseteq \mathbb{R}^i$ if and only if there is a connected superset of R on which p is analytically delineable. This is due to the fact that in general, we cannot assume that R is a connected set, but delineability is only defined on connected sets. We will use this trick in the following for other properties as well.

Further, note that for some properties such as *sgn_inv*(p), if they hold on $R \subseteq \mathbb{R}^i$, then they also hold on all subsets of R. For others such as *an_sub*(i) or *connected*(i), this is not the case: a subset of

R is not necessarily an analytic submanifold nor connected! This is one of the reasons why the proof rules below cannot cover all subsets where a given property holds, but they do cover sufficiently many subsets for our purposes.

For the remainder of this subsection we will present proof rules for these properties (with proofs of correctness available in Appendix A). For the ease of navigation we give references to the rules which apply to prove each property above (and provide similar references after introducing further properties in later subsections).

References.

***sample*(*s*)** Rule 4.12
***ord_inv*(*p*)** Rule 4.3, Rule 4.4, Rule 4.5
***sgn_inv*(*p*)** Rule 4.3, Rule 4.4, Rule 4.6, Rule 4.8, Rule 4.10
***non_null*(*p*)** Rule 4.2
***an_sub*(*i*)** Rule 4.7
***connected*(*i*)** Rule 4.11
***an_del*(*p*)** Rule 4.1

In the following rule definitions we list assumptions before the actual inference rule. These assumptions are formally part of the inference rule, but we keep them separated to improve readability.

We start by defining rules that relate to McCallum's projection.

Rule 4.1. Let $i \in \mathbb{N}$, $R \subseteq \mathbb{R}^i$, and $p \in \mathbb{Q}[x_1, \ldots, x_{i+1}]$, $\operatorname{level}(p) = i+1$. Assume that p is irreducible.

$$\mathit{an\_sub}(i)(R),\ \mathit{connected}(i)(R),\ \mathit{non\_null}(p)(R),\ \mathit{ord\_inv}(\operatorname{disc}_{x_{i+1}}(p))(R),$$
$$\mathit{sgn\_inv}(\operatorname{ldcf}_{x_{i+1}}(p))(R) \qquad \vdash \mathit{an\_del}(p)(R)$$

Rule 4.2. Let $i \in \mathbb{N}$, $R \subseteq \mathbb{R}^i$, $s \in \mathbb{R}^i$, and $p \in \mathbb{Q}[x_1, \ldots, x_{i+1}]$, $\operatorname{level}(p) = i+1$. Assume that p is irreducible, and $p = c_m \cdot x_{i+1}^m + \ldots + c_1 \cdot x_{i+1} + c_0$ such that $c_m, \ldots, c_0 \in \mathbb{Q}[x_1, \ldots, x_i]$.

$$\mathit{sample}(s)(R),\ \deg_{x_{i+1}}(p) > 1,\ \operatorname{disc}_{x_{i+1}}(p)(s) \neq 0,\ \mathit{sgn\_inv}(\operatorname{disc}_{x_{i+1}}(p))(R) \qquad \vdash \mathit{non\_null}(p)(R)$$
$$\mathit{sample}(s)(R),\ \exists j \in [m].\ (c_j(s) \neq 0 \land \mathit{sgn\_inv}(c_j)(R)) \qquad \vdash \mathit{non\_null}(p)(R)$$

Note that the condition $\deg_{x_{i+1}}(p) > 1 \land \operatorname{disc}_{x_{i+1}}(p)(s) \neq 0$ already implies that $p(s, x_{i+1}) \neq 0$, as does $c_j(s) \neq 0$. In practice, it is good to delay choosing which rule to use because we may observe that one of the non-zero c_j's is already required to be sign-invariant from some other part of the projection process (e.g. leading coefficients are often added to the projection). Moreover, the second case is trivial if the c_j is constant.

Rule 4.3. Let $p \in \mathbb{Q}$.

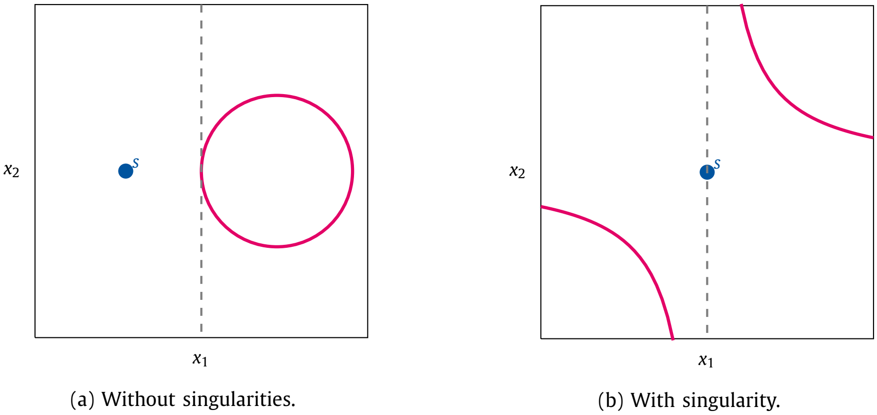
$$\vdash \mathit{ord\_inv}(p)(\mathbb{R}^0)$$
$$\vdash \mathit{sgn\_inv}(p)(\mathbb{R}^0)$$

Rule 4.4. Let $i \in \mathbb{N}_{>0}$, $R \subseteq \mathbb{R}^i$, and $p \in \mathbb{Q}[x_1, \ldots, x_i]$, $\operatorname{level}(p) = i$. Assume that p is reducible, and $\operatorname{factors}(p) = \{q_1, \ldots, q_k\}$.

$$\mathit{ord\_inv}(q_1)(R),\ \ldots,\ \mathit{ord\_inv}(q_k)(R) \qquad \vdash \mathit{ord\_inv}(p)(R)$$
$$\mathit{sgn\_inv}(q_1)(R),\ \ldots,\ \mathit{sgn\_inv}(q_k)(R) \qquad \vdash \mathit{sgn\_inv}(p)(R)$$

Rule 4.5. Let $i \in \mathbb{N}_{>0}$, $R \subseteq \mathbb{R}^i$, $s \in \mathbb{R}^i$, and $p \in \mathbb{Q}[x_1, \ldots, x_i]$, $\operatorname{level}(p) = i$. Assume that p is irreducible.

$$p(s) \neq 0,\ \mathit{sample}(s)(R),\ \mathit{sgn\_inv}(p)(R) \qquad \vdash \mathit{ord\_inv}(p)(R)$$
$$p(s) = 0,\ \mathit{sample}(s)(R),\ \mathit{an\_sub}(i-1)(R),\ \mathit{connected}(i)(R),\ \mathit{sgn\_inv}(p)(R),\ \mathit{an\_del}(p)(R) \qquad \vdash \mathit{ord\_inv}(p)(R)$$



(a) Without singularities.

(b) With singularity.

Fig. 4. Examples of the two cases of polynomials without roots at the projection s_1 of the sample $s = (s_1, s_2)$. Dashed lines denote cylinder boundaries over the projected cell.

4.4. *Sign invariance for polynomials without roots*

For polynomials without roots over the current sample, we must ensure that no further root appears over the underlying cell. In this case, it is sufficient to make the polynomial delineable. Fig. 4 illustrates some cases.

Rule 4.6. Let $i \in \mathbb{N}$, $R \subseteq \mathbb{R}^i$, $s \in \mathbb{R}^{i-1}$, and $p \in \mathbb{Q}[x_1, \dots, x_i]$, $\text{level}(p) = i$. Assume that p is irreducible, and $\text{realRoots}(p(s, x_i)) = \emptyset$.

$$\textit{sample}(s)(R),\ \textit{an\_del}(p)(R) \qquad \vdash \textit{sgn\_inv}(p)(R)$$

4.5. *Cell boundary representations*

We are aiming to generate the description of a sign-invariant cell for the input polynomials. In the language of our proof system: at every level we identify a symbolic interval upon which we prove sign-invariance of the polynomials of that level. To do this, we take their real roots over the current sample into account.

We determine whether we are in a section or sector (i.e. whether a polynomial has a root exactly at the current sample point or not) and pick indexed root expressions (as given by Definition 2.5) to be symbolic descriptions of the cell boundaries that generalize a concrete interval around the sample point: either the root at the current sample or the largest root below and the lowest root above. Note that the choice of such boundaries might not be unique in the cases where two root functions cross over the underlying sample. We will discuss more general choices of the symbolic intervals in Section 5.2.

Our next property is designed to state that a designated representation describes the cell boundaries on the current level.

To do so, we define how we use indexed root expressions to refer to real root functions of a polynomial. An indexed root expression has incontinuities where the number of the roots of its defining polynomial changes. Therefore a root function of a polynomial is only uniquely determined by an indexed root expression in combination with a given sample. In the following, we define the unique real root function determined this way:

Definition 4.3 *(Root functions).* Let $i \in \mathbb{N}_{>0}$, $s \in \mathbb{R}^{i-1}$, and $\xi \in \text{irExpr}(\xi.p, s)$ be an indexed root expression of level i.

Let $R \subseteq \mathbb{R}^i$ maximal such that $\xi.p$ is delineable on R, R is connected, and $s \in R$. Then $\theta_{\xi,s}$ denotes the real root function of p on R that witnesses analytic delineability of p and equals $\xi|_R$.

Property 4.2. *Let $i \in \mathbb{N}_{>0}$, $R \subseteq \mathbb{R}^i$, $s \in \mathbb{R}^{i-1}$, and $\mathtt{I}$ be a symbolic interval of level i.*

The property repr$(\mathtt{I}, s)$ holds on R if and only if $\mathtt{I}.l \in \mathrm{irExpr}(\mathtt{I}.l.p, s)$ (if $\mathtt{I}.l \neq -\infty$), $\mathtt{I}.u \in \mathrm{irExpr}(\mathtt{I}.u.p, s)$ (if $\mathtt{I}.u \neq \infty$) respectively $\mathtt{I}.b \in \mathrm{irExpr}(\mathtt{I}.b.p, s)$ and one of the following holds:

- $\mathtt{I} = (\mathit{sector}, l, u)$, $\mathrm{dom}(\theta_{l,s}) \cap \mathrm{dom}(\theta_{u,s}) \supseteq R{\downarrow_{[i-1]}}$ *and* $R = \{(r, r') \mid r \in R{\downarrow_{[i-1]}}, r' \in (\theta_{l,s}(r), \theta_{u,s}(r))\}$; *or*
- $\mathtt{I} = (\mathit{sector}, -\infty, u)$, $\mathrm{dom}(\theta_{u,s}) \supseteq R{\downarrow_{[i-1]}}$ *and* $R = \{(r, r') \mid r \in R{\downarrow_{[i-1]}}, r' \in (-\infty, \theta_{u,s}(r))\}$; *or*
- $\mathtt{I} = (\mathit{sector}, l, \infty)$, $\mathrm{dom}(\theta_{l,s}) \supseteq R{\downarrow_{[i-1]}}$ *and* $R = \{(r, r') \mid r \in R{\downarrow_{[i-1]}}, r' \in (\theta_{l,s}(r), \infty)\}$; *or*
- $\mathtt{I} = (\mathit{sector}, -\infty, \infty)$ *and* $R = \{(r, r') \mid r \in R{\downarrow_{[i-1]}}, r' \in \mathbb{R}\}$; *or*
- $\mathtt{I} = (\mathit{section}, b)$, $\mathrm{dom}(\theta_{b,s}) \supseteq R{\downarrow_{[i-1]}}$ *and* $R = \{(r, r') \mid r \in R{\downarrow_{[i-1]}}, r' = \theta_{b,s}(r)\}$,

(for the real root functions $\theta_{l,s}$, $\theta_{u,s}$ respectively $\theta_{u,b}$ according to Definition 4.3).

References.

***repr*$(\mathtt{I}, s)$** Rule 4.13

Since the cell's boundaries are described by real root functions, we can prove that it is an analytic submanifold.

Rule 4.7. Let $i \in \mathbb{N}_{>0}$, $R \subseteq \mathbb{R}^i$, $s \in \mathbb{R}^{i-1}$, and $\mathtt{I}$ be a symbolic interval of level i.

$$\vdash \mathit{an\_sub}(0)(\mathbb{R}^0)$$

$$\mathit{repr}(\mathtt{I}, s)(R),\ \mathit{an\_sub}(i-1)(R) \qquad \vdash \mathit{an\_sub}(i)(R)$$

4.6. Equational constraint projection

An optimization to CAD which tailors it to the logical structure of the problem is the theory of equational constraints. A polynomial equation is an *equational constraint* if it is logically implied by the truth of the input formula. If the input to CAD has an equational constraint then we may perform fewer projection and lifting operations to achieve truth-invariance. This optimization dates back to McCallum (1999) with the recent paper by England et al. (2020) summarizing the state of the art. The examples there demonstrate the drastic savings that are achievable in these cases (the analysis by England et al. (2020) shows the double exponent in the complexity bound decreases for each equational constraint).

In the context of building a single cell we have even greater scope to use equational constraint savings. Here, any time we are generalizing in the section case, we can apply the reduced projection and lifting. I.e., if a polynomial is zero at our sample then it must also be zero throughout the cell we are generalizing the sample to, and so for the purposes of this cell, it is an equational constraint. In this case we need only make the section defining polynomial delineable. All other constraints can be made sign-invariant simply by including resultants with the section defining polynomial (but not other projection polynomials).

Rule 4.8. Let $i \in \mathbb{N}_{>0}$, $R \subseteq \mathbb{R}^i$, $s \in \mathbb{R}^{i-1}$, $p \in \mathbb{Q}[x_1, \ldots, x_i]$, $\mathrm{level}(p) = i$, and $\mathtt{I}$ be a symbolic interval of level i. Assume that p is irreducible, and $\mathtt{I} = (\mathrm{section}, b)$.

Let $Q := \mathit{an\_sub}(i-1)(R) \wedge \mathit{connected}(i-1)(R) \wedge \mathit{repr}(\mathtt{I}, s)(R) \wedge \mathit{an\_del}(b.p)(R)$.

$$Q,\ b.p = p \qquad \vdash \mathit{sgn\_inv}(p)(R)$$

$$Q,\ b.p \neq p,\ \mathit{ord\_inv}(\mathrm{res}_{x_i}(b.p, p))(R) \qquad \vdash \mathit{sgn\_inv}(p)(R)$$

4.7. Root orderings

We define a partial ordering on the indexed root expressions of the given set of polynomials, which we will use to ensure that none of their roots crosses the cell's boundary on the current level.

We first give a general definition of indexed root orderings.

Definition 4.4 *(Indexed root ordering).* Let $i \in \mathbb{N}$, and Ξ be a set of indexed root expressions of level $i+1$. An *indexed root ordering on* Ξ is a relation $\preceq \subseteq \Xi \times \Xi$ such that its reflexive and transitive closure $\preceq^t$ is a partial order on Ξ. Indexed root orderings of this form are also called *indexed root ordering of level* $i+1$, and we define $\mathrm{dom}(\preceq) = \{\xi, \xi' \mid (\xi, \xi') \in \preceq\}$.

Let $s \in \mathbb{R}^i$. An indexed root ordering $\preceq$ of level $i+1$ *matches* s if and only if $\xi \in \mathrm{irExpr}(\xi.p, s)$ for all $\xi \in \mathrm{dom}(\preceq)$ and $\xi(s) \leq \xi'(s)$ for all $(\xi, \xi') \in \preceq$.

In our algorithm we will pick an indexed root ordering such that, in the sector case, roots lower than the interval's lower bound remain lower, and roots greater than the interval's upper bound remain greater. In the section case, the lower and upper bounds in that condition both refer to a single bound. We do not give an explicit definition here yet, as it will be part of the rules defined below.

The motivation for introducing the general concept of indexed root orderings is to allow for choices between different root orderings. We will discuss them in Section 5.2.

Example 4.2. Given indexed root expressions $\Xi = \{\xi_1, \ldots, \xi_5\}$ of level $i+1$ and a sample $s \in \mathbb{R}^i$ such that $\xi_1(s) < \ldots < \xi_5(s)$ and $\mathtt{I} = (\mathrm{sector}, \xi_1, \xi_2)$; then $\preceq = \{(\xi_2, \xi_3), (\xi_2, \xi_4), (\xi_2, \xi_5)\}$, $\preceq' = \{(\xi_2, \xi_3), (\xi_3, \xi_4), (\xi_2, \xi_5)\}$, and $\preceq'' = \{(\xi_2, \xi_3), (\xi_3, \xi_4), (\xi_4, \xi_5)\}$ are all indexed root orderings on Ξ that allow us to derive sign-invariance of the corresponding polynomials above the sample s and the interval $\mathtt{I}$.

We need to maintain that an indexed root ordering holds over the underlying cell. First, we will define this property formally.

Property 4.3. *Let $i \in \mathbb{N}$, $R \subseteq \mathbb{R}^i$, $s \in \mathbb{R}^i$, $\preceq$ be an indexed root ordering of level $i+1$, and $\preceq^t$ be the reflexive and transitive closure of $\preceq$.*

The property $\mathrm{ir\_ord}(\preceq, s)$ *holds on R if and only if $\preceq$ matches s, for all $\xi \in \mathrm{dom}(\preceq)$ it holds $R \subseteq \mathrm{dom}(\theta_{\xi,s})$, and for all $\xi, \xi' \in \mathrm{dom}(\preceq)$ it holds that*

$$\xi \preceq^t \xi' \wedge \xi(s) < \xi'(s) \implies \theta_{\xi,s}(r) < \theta_{\xi',s}(r) \textit{ for all } r \in R$$

and

$$\xi \preceq^t \xi' \wedge \xi(s) = \xi'(s) \implies \theta_{\xi,s}(r) = \theta_{\xi',s}(r) \textit{ for all } r \in R$$

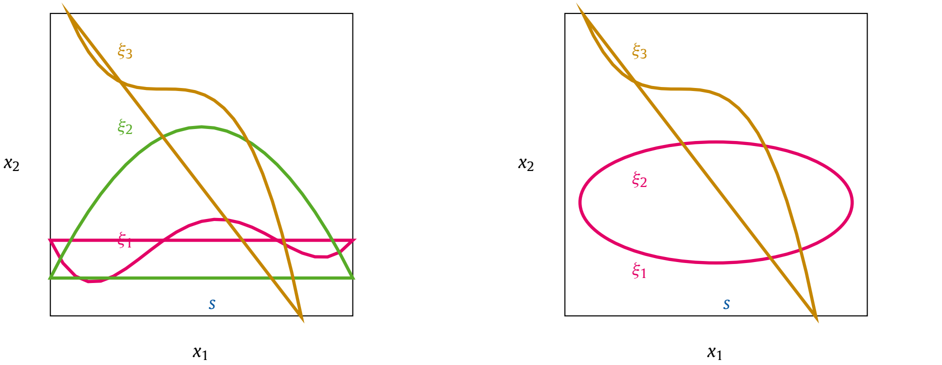
(for the real root functions $\theta_{\xi,s}$, $\theta_{\xi',s}$ according to Definition 4.3).

References.

***ir_ord*($\preceq$, *s*)** Rule 4.9

Note that the semantics of an indexed root expression ξ is only defined in combination with the sample s, as this uniquely determines the real root function $\theta_{\xi,s}$; thus, we add s to the signature of the property. To clarify this further: the sample s is only used to identify the referred real root functions and is not necessarily contained in the constructed cell by definition of the above property (although in the following proof rules, this will be required).

The property is maintained by delineability and adding resultants for the defining polynomials of a pair of indexed root expressions to the projection. We essentially prove that the ordering of root



(a) Assume $\preceq = \{(\xi_1, \xi_2), (\xi_2, \xi_3)\}$. All three real root functions are defined over the underlying cell. Only two resultants are necessary to maintain the ordering of the root functions, e.g. $\mathrm{res}_{x_2}(\xi_1.p, \xi_2.p)$ and $\mathrm{res}_{x_2}(\xi_2.p, \xi_3.p)$. We conclude the ordering between ξ_1 and ξ_3 by transitivity.

(b) Assume $\preceq = \{(\xi_1, \xi_2), (\xi_1, \xi_3)\}$. The lower bound on x_1 is due to the delineability of the common defining polynomial of ξ_1 and ξ_2. The ordering $\xi_1 \preceq \xi_2$ is maintained by $\mathrm{disc}_{x_2}(\xi_1.p)$ and the ordering $\xi_1 \preceq \xi_3$ by the resultant $\mathrm{res}_{x_2}(\xi_1.p, \xi_3.p)$. Note that latter causes the upper bound on x_1 to be smaller than necessary, as $\xi_3.p$ intersects $\theta_{\xi_2,s}$ “before” it intersects $\theta_{\xi_1,s}$. More sophisticated rules circumventing this are future work.

Fig. 5. Two examples for the property *ir_ord*$(\preceq, s)$.

functions ensured by the order-invariance of resultants is transitive. Fig. 5 shows an example where transitivity is maintained for three root functions.

Rule 4.9. Let $i \in \mathbb{N}$, $R \subseteq \mathbb{R}^i$, $s \in \mathbb{R}^i$, and $\preceq$ be an indexed root ordering of level $i+1$. Assume that $\xi.p$ is irreducible for all $\xi \in \mathrm{dom}(\preceq)$, and that $\preceq$ matches s.

$$\mathit{sample}(s)(R),\ \mathit{an\_sub}(i)(R),\ \mathit{connected}(i)(R),\ \forall \xi \in \mathrm{dom}(\preceq).\ \mathit{an\_del}(\xi.p)(R),$$
$$\forall (\xi, \xi') \in \preceq.\ \mathit{ord\_inv}(\mathrm{res}_{x_{i+1}}(\xi.p, \xi'.p))(R) \qquad \vdash \mathit{ir\_ord}(\preceq, s)(R)$$

Now we are ready to define the rule for ensuring the sign-invariance of a polynomial p, given a representation $\mathtt{I}$ and an indexed root ordering $\preceq$. The latter will be used to ensure that no real root function of p crosses the cell’s boundaries. The cases are depicted in Fig. 6.

Rule 4.10. Let $i \in \mathbb{N}_{>0}$, $R \subseteq \mathbb{R}^i$, $s \in \mathbb{R}^{i-1}$, $p \in \mathbb{Q}[x_1, \ldots, x_i]$, $\mathrm{level}(p) = i$, $\mathtt{I}$ be a symbolic interval of level i, $\preceq$ be an indexed root ordering of level i, and $\preceq^t$ be the reflexive and transitive closure of $\preceq$.

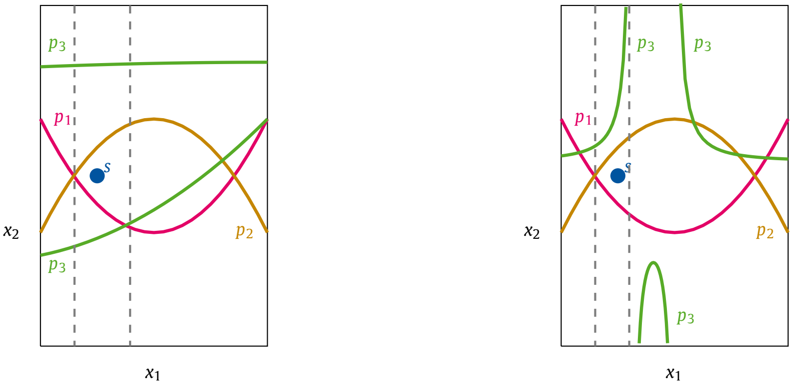
We choose l, u such that either $\mathtt{I} = (\mathrm{sector}, l, u)$ or ($\mathtt{I} = (\mathrm{section}, b)$ for $b = l = u$).

Assume that p is irreducible, $\mathrm{irExpr}(p, s) \neq \emptyset$, $\xi.p$ is irreducible for all $\xi \in \mathrm{dom}(\preceq)$, $\preceq$ matches s, and for all $\xi \in \mathrm{irExpr}(p, s)$ it holds either $\xi \preceq^t l$ or $u \preceq^t \xi$.

$$\mathit{sample}(s)(R),\ \mathit{repr}(\mathtt{I}, s)(R),\ \mathit{ir\_ord}(\preceq, s)(R),\ \mathit{an\_del}(p)(R) \qquad \vdash \mathit{sgn\_inv}(p)(R)$$

4.8. Connectedness

For maintaining properties (in particular the order-invariance of some polynomials) on higher levels, we need to maintain connectedness. Besides some preconditions, i.e. that the cell’s boundaries are described by two well-defined real root functions, we need to ensure that the lower and upper bound of a sector do not cross as illustrated in Fig. 7.



(a) The indexed root ordering ensures that real root functions remain outside the cell. As p_3 is delineable, this is sufficient to prove sign invariance.

(b) We note that if p_3 has singularities, the cell might be made smaller than required for ensuring sign-invariance due to the strong notion of delineability. Relaxing this notion is part of future research.

Fig. 6. Two examples for maintaining the sign-invariance of a polynomial p_3 in a cell (here defined by the enclosed region between the graphs of p_1 and p_2).

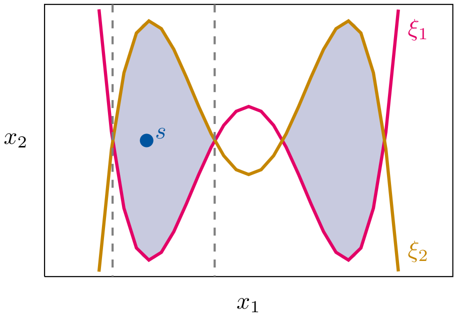


Fig. 7. Let $\mathtt{I} = (sector, \xi_1, \xi_2)$. Then $repr(\mathtt{I}s_1)$ holds in the whole blue shaded area, thus forming a non-connected subset of $\mathbb{R}^2$. By adding $\mathrm{res}_{x_2}(\xi_1.p, \xi_2.p)$ to the projection, the subset is made connected by restricting the underlying cell as indicated by the dashed vertical lines. (For interpretation of the colours in the figure(s), the reader is referred to the web version of this article.)

Rule 4.11. Let $i \in \mathbb{N}_{>0}$, $R \subseteq \mathbb{R}^i$, $s \in \mathbb{R}^{i-1}$, $\mathtt{I}$ be a symbolic interval of level i, $\preceq$ be an indexed root ordering of level i, and $\preceq^t$ be the reflexive and transitive closure of $\preceq$. Assume that $\preceq$ matches s.

Let $Q := connected(i-1)(R) \wedge repr(\mathtt{I}, s)(R)$.

$$\begin{array}{ll}
 & \vdash connected(0)(\mathbb{R}^0) \\
Q\,,\ \mathtt{I} = (sector, l, u),\ l \neq -\infty,\ u \neq \infty,\ l \preceq^t u,\ ir\_ord(\preceq, s) & \vdash connected(i)(R) \\
Q\,,\ \mathtt{I} = (sector, l, u),\ l = -\infty \vee u = \infty & \vdash connected(i)(R) \\
Q\,,\ \mathtt{I} = (section, b) & \vdash connected(i)(R)
\end{array}$$

4.9. Generalization of the current sample

We define a mapping for generalizing the sample to the cell on the current level.

Rule 4.12. Let $i \in \mathbb{N}_{>0}$, $R \subseteq \mathbb{R}^i$, $s \in \mathbb{R}^i$, and $\mathtt{I}$ be a symbolic interval of level i. Assume that $s_i \in setOf(s_{[i-1]}, \mathtt{I})$.

$$\vdash \mathit{sample}(())(\mathbb{R}^0)$$

$$\mathit{sample}(s_{[i-1]})(R),\ \mathit{repr}(\mathtt{I}, s_{[i-1]})(R) \qquad \vdash \mathit{sample}(s)(R)$$

4.10. Cell descriptions

Now, we define a property that states that the cell is described by its representation.

Property 4.4. *Let $i \in \mathbb{N}_{>0}$, $R \subseteq \mathbb{R}^i$, and $\mathtt{I}$ be a symbolic interval of level i.*
The property $\mathit{holds}(\mathtt{I})$ holds on R if and only if $R = \mathit{setOf}(R\downarrow_{[i-1]}, \mathtt{I})$.

This property is the only one that cannot be mapped to a set of other properties, i.e. properties of this kind are the assumptions in our proof system.

Given $\mathit{holds}(\mathtt{I})$, we can maintain $\mathit{repr}(\mathtt{I}, s)$. To do so, we need to ensure that the indexed root expressions describing the cell's boundaries always refer to the same roots on the underlying cell. This can be achieved by making their defining polynomials delineable (such that the referred real root function is defined and the number of roots below the referred root function is constant over the underlying cell).

Rule 4.13. Let $i \in \mathbb{N}_{>0}$, $R \subseteq \mathbb{R}^{i-1}$, $s \in \mathbb{R}^{i-1}$, and $\mathtt{I}$ be a symbolic interval of level i. Assume that $\mathtt{I}.l \in \mathrm{irExpr}(\mathtt{I}.l.p, s)$ (if $\mathtt{I}.l \neq -\infty$), $\mathtt{I}.u \in \mathrm{irExpr}(\mathtt{I}.u.p, s)$ (if $\mathtt{I}.u \neq \infty$) respectively $\mathtt{I}.b \in \mathrm{irExpr}(\mathtt{I}.b.p, s)$.

$$\mathit{sample}(s)(R),\ \mathit{holds}(\mathtt{I})(R),\ \mathtt{I} = (\mathrm{section}, b),\ \mathit{an\_del}(b.p)(R) \qquad \vdash \mathit{repr}(\mathtt{I}, s)(R)$$

$$\mathit{sample}(s)(R),\ \mathit{holds}(\mathtt{I})(R),\ \mathtt{I} = (\mathrm{sector}, l, u),\ l = -\infty \vee \mathit{an\_del}(l.p)(R),\ u = \infty \vee \mathit{an\_del}(u.p)(R) \qquad \vdash \mathit{repr}(\mathtt{I}, s)(R)$$

4.11. Ordering of properties

We observe that the rules of inference defined in Rules 4.1 to 4.13 are cycle-free in the sense that all properties in the antecedents are smaller than the consequent property according to some ordering $\lhd$:

Definition 4.5. The ordering $\lhd$ is defined such that properties of level i are greater than any property of level $i-1$, and such that it satisfies the following partial order of properties of each level i (starting with the greatest element):

1. $\mathit{ir\_ord}(\preceq, s)$ for all $\preceq$ for roots of level i and $s \in \mathbb{R}^{i-1}$
2. $\mathit{an\_del}(p)$ for all p of level i
3. $\mathit{non\_null}(p)$ for all p of level i
4. $\mathit{ord\_inv}(p)$ for all reducible p of level i
5. $\mathit{ord\_inv}(p)$ for all irreducible p of level i
6. $\mathit{sgn\_inv}(p)$ for all reducible p of level i
7. $\mathit{sgn\_inv}(p)$ for all irreducible p of level i
8. $\mathit{connected}(i)$
9. $\mathit{an\_sub}(i)$
10. $\mathit{sample}(s)$ for all $s \in \mathbb{R}^i$ of level i
11. $\mathit{repr}(\mathtt{I}, s)$ for all $\mathtt{I}$ of level i and $s \in \mathbb{R}^{i-1}$
12. $\mathit{holds}(\mathtt{I})$ for all $\mathtt{I}$ of level i

The proof rules introduced in this section are visualized in Fig. 8 which shows the relationships between rules of different levels.

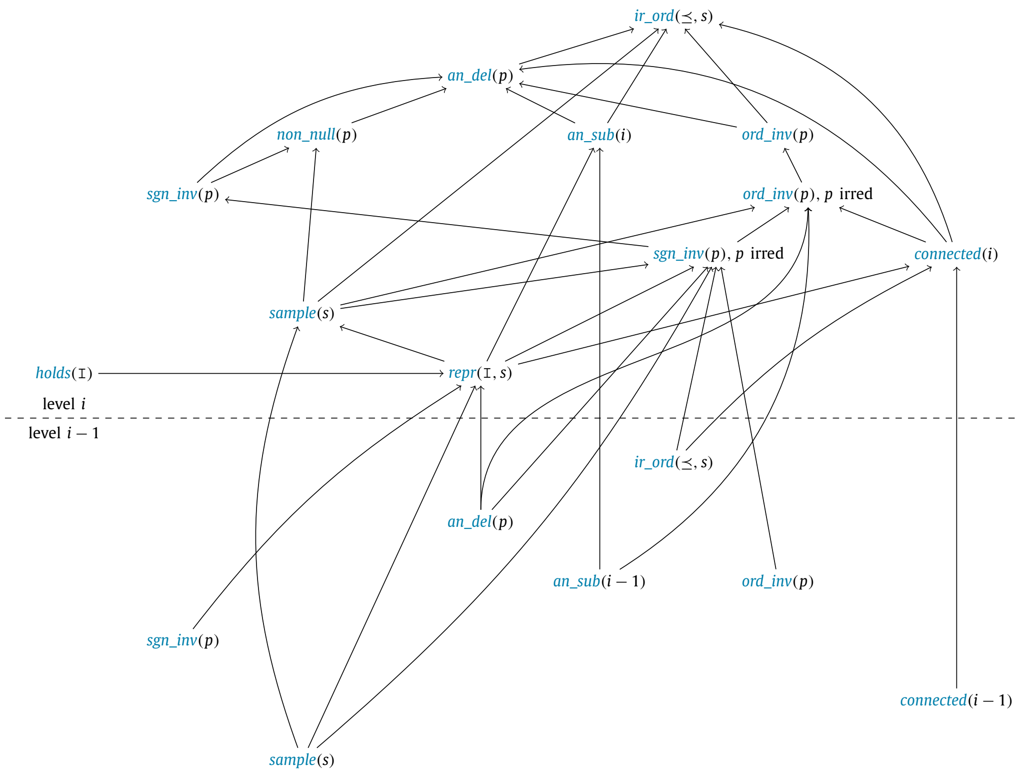


Fig. 8. Proof rule relationships; *irred* stands for *irreducible*.

5. Levelwise construction of a single cell

So far, we presented an abstract proof system, i.e. a set of rules which allow us to construct a proof for sign-invariance of some polynomials in a cell. In the following, we will give an algorithm for constructing a single cell in a levelwise manner by specifying how the proof rules are applied. We first give the algorithmic framework and then define heuristics needed to instantiate the algorithm.

5.1. Algorithm to construct a single cell

Algorithm 1 constructs a single cell. The input to the algorithm is a set of polynomials for which a cell is to be computed on which these polynomials are sign-invariant. This initializes the set of properties Q on Line 1 which is maintained to contain properties that need to be fulfilled by the yet to be chosen representations on the lower levels. I.e. at the beginning of iteration i, the data structure $(\mathtt{I}_1, \ldots, \mathtt{I}_i)$ needs to maintain all properties in Q. To do so, Algorithm 1 iteratively calls Algorithm 2 to process the properties and construct the interval on level i, taking note of any failure cases due to nullifications that might occur.

Algorithm 3 is a sub-algorithm that replaces a property in Q by a set of properties Q' induced by a proof rule if no such set is already contained in Q, and otherwise simply removes the property from Q. *FAIL* is returned here if and only if no proof rule is applicable; which is the case only when a polynomial is nullified (i.e. Rule 4.2) and equational constraint projection does not allow us to ignore that fact. The sets Q' induced by a proof rule are determined in Line 1 as follows. Assume we want to derive the property *non_null*(p) of $R \subseteq \mathbb{R}^i$ for some polynomial p of level $i+1$ with respect to the current sample $s \in \mathbb{R}^i$. The algorithm would check if the conditions of the proof rules in Rule 4.2 are satisfied, and then determine the corresponding properties of R which form the set Q'. In the first case, we check whether p is irreducible, $\deg_{x_{i+1}}(p) > 1$, and $\mathrm{disc}_{x_{i+1}}(p)(s) \neq 0$ hold; if yes, we add the set $Q' = \{sample(s), sgn\_inv(\mathrm{disc}_{x_{i+1}}(p))\}$ to the set of choices. In the second case, we check whether p is irreducible and there exists a coefficient c_j of p such that $c_j(s) \neq 0$ holds; if yes, we add the set $Q' = \{sample(s), sgn\_inv(c_j)\}$ to the set of choices.

Algorithm 2 is used to extend the cell construction by a level. It applies the proof rules according to the ordering $\lhd$ on the properties from Definition 4.5 until the only properties of level i remaining are the sign-invariance of some irreducible polynomials (note that all smaller properties w.r.t. $\lhd$ are provable with only the sample s present except if a nullification occurs). Then, a *representation* consisting of a symbolic interval $\mathtt{I}$, a set of polynomials E and an indexed root ordering $\preceq$ is chosen. The roles of $\mathtt{I}$ and $\preceq$ are already discussed above; the polynomials in E are meant to be excluded from the indexed root ordering, and thus the application of the equational constraint projection rule (Rule 4.8) is enforced. The formal requirements on the representation will be defined in Definition 5.1. Finally, the remaining properties on level i are eliminated by application of the proof rules, until only a property of the form $holds(\mathtt{I})$ is left on the level (which is the greatest property of the current level w.r.t. $\lhd$).

Algorithm 1: `single_cell`(i, P, s).

```
Input  : finite $P \subseteq \mathbb{Q}[x_1, \ldots, x_n]$, $s \in \mathbb{R}^n$
Output: cell data structure $\mathtt{R}$ such that $s \in setOf(\mathtt{R})$ and all polynomials in $P$ are sign-invariant on $setOf(\mathtt{R})$; or $FAIL$
$Q := \{sgn\_inv(p) \mid p \in P\}$
for $i = n, \ldots, 1$ do
    $result :=$ construct_interval($i$, $Q$, $s$) // Algorithm 2
    if $result \neq FAIL$ then
        $\mathtt{I}_i, Q := result$
    else
        return $FAIL$
return $(\mathtt{I}_1, \ldots, \mathtt{I}_n)$
```

Algorithm 2: `construct_interval`(i, Q, s).

```
Input  : $i \in \mathbb{N}_{>0}$, finite $Q \subseteq \mathrm{Prop}|_{[i]}$, $s \in \mathbb{R}^i$
Output: an interval data structure $\mathtt{I}$ such that $s \in setOf(s_{[i-1]}, \mathtt{I})$, and a set $Q' \subseteq \mathrm{Prop}|_{[i-1]}$ such that for any $R \subseteq \mathbb{R}^{i-1}$
         it holds $(\forall q' \in Q'.\, q'(R)) \implies (\forall q \in Q.\, q(setOf(R, \mathtt{I})))$; or $FAIL$
foreach $q \in Q|_i$ where $q$ is the greatest element with respect to $\lhd$ (from Definition 4.5) and $q \neq sgn\_inv(p)$ for an irreducible $p$
  do
    $Q :=$ apply_pre($i$,$Q$,$q$,$(s)$) // Algorithm 3
    if $Q = FAIL$ then
        return $FAIL$
$P_{nonnull} := \{p \mid sgn\_inv(p) \in Q|_i$ s.t. $p(s_{[i-1]}, x_i) \neq 0\}$
$\Xi :=$ irExpr$(P_{nonnull}, s_{[i-1]})$
choose representation $(\mathtt{I}, E, \preceq)$ of $\Xi$ with respect to $s$
if $i > 1$ then
    foreach $q \in Q|_i$ where $q$ is the greatest element with respect to $\lhd$ (from Definition 4.5) and $q \neq holds(\mathtt{I})$ do
        $Q :=$ apply_pre($i$,$Q$,$q$,$(s, \mathtt{I}, \preceq)$)                                // Algorithm 3
        if $Q = FAIL$ then
            return $FAIL$
/* $Q|_i$ contains now only $holds(\mathtt{I})$                                                */
return $\mathtt{I}$,$(Q \setminus Q|_i)$
```

Theorem 5.1. *Let $P \subseteq \mathbb{Q}[x_1, \ldots, x_n]$, and $s \in \mathbb{R}^n$. The method* `single_cell` *(Algorithm 1) either returns a cell data structure $\mathtt{R}$ such that all polynomials in P are sign-invariant on setOf$(\mathtt{R})$ and $s \in$ setOf$(\mathtt{R})$, or returns FAIL.*

Algorithm 3: `apply_pre`$(i, Q, q, parameters)$.

```
Input : $i \in \mathbb{N}_{>0}$, finite $Q \subseteq \mathrm{Prop}|_{[i]}$, $q \in \mathrm{Prop}|_i \cap Q$ and either $parameters = (s)$ or $parameters = (s, \mathtt{I}, \preceq)$ with $s \in \mathbb{R}^i$,
        symbolic interval $\mathtt{I}$ on level $i$, indexed root ordering $\preceq$ on level $i$
Output: a set $(Q \setminus \{q\}) \cup Q' \subseteq \mathrm{Prop}|_{[i]} \setminus \{q\}$ such that for all $R \subseteq \mathbb{R}^i$ it holds $(\forall q' \in Q'.\, q'(R)) \implies q(R)$; or FAIL
1 let $choices \subseteq 2^{\mathrm{Prop}}$ such that each $Q' \in choices$ is a set of properties from a proof rule which can be applied according to
    $parameters$ to derive $q$ (see explanatory text above for details)
2 if $choices = \emptyset$ then
3     return FAIL
4 if $\nexists Q' \in choices.\ Q' \subseteq Q$ then
5     choose $Q' \in choices$
6     return $(Q \setminus \{q\}) \cup Q'$
7 else
8     return $Q \setminus \{q\}$
```

Proof. Note that the algorithm will terminate as Algorithm 3 replaces properties by smaller properties according to the ordering as defined in Definition 4.5, there does exist a smallest property, and the CAD projection of a set of polynomials is finite. The correctness follows from the correctness proofs of the rules of inference. □

5.2. Heuristic choices

Our algorithm has two points where a choice must be made: (1) when choosing a *rule of inference* to apply in Algorithm 3 Line 5; and (2) when choosing the *representation* in Algorithm 2 Line 7.

For the first choice, the rules of inference may define multiple possible property sets for a property that each imply that property and so we have the freedom to choose which to use. We aim to pick the "best" such set. One could compute each of these sets individually and compare them, or, to avoid superfluous heavy computations of resultants or discriminants, prefer sets which seem easier to compute. The latter approach, making the choice heuristically (with respect to e.g. estimated computational effort, or degrees of polynomials, or number of polynomials / real roots) is the approach taken by our implementation. For instance, in Rule 4.2, we always pick a coefficient rather than computing a discriminant. Note that even if sets involving resultants or discriminants are avoided where possible, the algorithm should first check whether one such set is already maintained before computing any new set.

For the second choice, we formalize what we mean by a representation. As already discussed, there might be multiple choices for the cell description $\mathtt{I}$ on the current level and the indexed root ordering $\preceq$ for fulfilling the requirements for Rule 4.10. Additionally, if $\mathtt{I}$ is a section, we can make use of the equational constraints projection in Rule 4.8. A representation determines all these parameters.

Definition 5.1 *(Representation for Ξ).* Let $i \in \mathbb{N}_{>0}$, $s \in \mathbb{R}^i$, and Ξ be a set of indexed root expressions of level i such that $\xi \in \mathrm{irExpr}(\xi.p, s_{[i-1]})$ for every $\xi \in \Xi$.

A *representation* for Ξ with respect to s is a tuple $(\mathtt{I}, E, \preceq)$ where $\mathtt{I}$ is a symbolic interval of level i, E is a set of polynomials of level i, and $\preceq$ is an indexed root ordering of level i such that

- $s \in \mathit{setOf}(s_{[i-1]}, \mathtt{I})$,
- either $p \in E$ or $\mathrm{irExpr}(p, s_{[i-1]}) \subseteq \mathrm{dom}(\preceq)$ for all $p \in \{\xi.p \mid \xi \in \Xi\}$,
- if $E \neq \emptyset$ then $\mathtt{I} = (\mathrm{section}, b)$ for some indexed root expression b,
- $\preceq$ matches $s_{[i-1]}$, and
- $\xi \preceq^t \mathtt{I}.l$ or $\mathtt{I}.u \preceq^t \xi$ for all $\xi \in \Xi$ if $\mathtt{I} = (\mathrm{sector}, l, u)$ respectively $\xi \preceq^t \mathtt{I}.b$ or $\mathtt{I}.b \preceq^t \xi$ for all $\xi \in \Xi$ if $\mathtt{I} = (\mathrm{section}, b)$.

This definition is quite general by allowing *additional* indexed root expressions in Ξ' compared to the set Ξ describing the roots of the present polynomials. This for example enables heuristics to under-approximate the constructed cell to reduce heavy computations in future work (see Section 8.1).

In the following however, we will assume that $\Xi' = \Xi$, that is, $\mathtt{I}$ and $\preceq$ will only consider roots from Ξ corresponding to the polynomials for which we need to derive sign-invariance.

Note that we omit the requirements of Rule 4.11 here yet for readability reasons and as the adaption is trivial. If required, we only need to add the pair $(\mathtt{I}.l, \mathtt{I}.u)$ to the indexed root ordering.

For the symbolic interval $\mathtt{I}$, we minimize the degrees in the main variable of the defining polynomials.

Definition 5.2 *(Choice of the symbolic interval).* Let $i \in \mathbb{N}_{>0}$, $P \subseteq \mathbb{Q}[x_1, \dots, x_i]$ be a set of irreducible polynomials of level i, $s \in \mathbb{R}^i$ be a sample such that no $p \in P$ is nullified on $s_{[i-1]}$, and $\Xi \subseteq \mathrm{irExpr}(P, s_{[i-1]})$.

We define the sets Ξ_{lo} and Ξ_{up} of the closest lower respectively upper bounds of s_i as follows:

$$\Xi_{lo} = \{\xi \in \Xi \mid \xi(s_{[i-1]}) \leq s_i \wedge \forall \xi' \in \Xi. \left(\xi'(s_{[i-1]}) \leq s_i \implies \xi'(s_{[i-1]}) \leq \xi(s_{[i-1]})\right)\},$$
$$\Xi_{up} = \{\xi \in \Xi \mid s_i \leq \xi(s_{[i-1]}) \wedge \forall \xi' \in \Xi. \left(s_i \leq \xi'(s_{[i-1]}) \implies \xi(s_{[i-1]}) \leq \xi'(s_{[i-1]})\right)\}.$$

We pick $\xi_{lo} \in \Xi_{lo}$ such that $\deg_{x_i}(\xi_{lo}.p)$ is minimal and $\xi_{up} \in \Xi_{up}$ such that $\deg_{x_i}(\xi_{up}.p)$ is minimal. Additionally, if $\Xi_{lo} = \Xi_{up}$, then we require $\xi_{lo} = \xi_{up}$.

We define the *lowest degree interval* $\mathtt{I}_{\mathrm{ldeg}}$ of s with respect to Ξ as

$$\mathtt{I}_{\mathrm{ldeg}} = (section, \xi_{lo}) \qquad \text{if } \xi_{lo} = \xi_{up},$$
$$\mathtt{I}_{\mathrm{ldeg}} = (sector, \xi_{lo}, \xi_{up}) \qquad \text{otherwise.}$$

For the indexed root ordering $\preceq$, there are two possibilities: aim to make the underlying cell as big as possible; or aim to avoid heavy resultant computations. In theory, we could compute the results for all (or all promising) possible indexed root orderings and pick the best one. However, as this is infeasible in practice, we define below several alternative heuristics with different rationales.

To achieve this we employ a somewhat idealistic view on the problem. Recall that an indexed root ordering is ensured by making resultants order-invariant, which is often a stronger ordering on the roots than required by the picked indexed root ordering. We ignore this fact and base our heuristics on the set of indexed roots without considering common defining polynomials between them (for now). This is further discussed in Section 6.

We start with the following observation: as the given derivation rules always require delineability for all polynomials whose sign-invariance is proven using an indexed root ordering, a fixed ordering of all real root functions defined at $s_{[i-1]}$ of such polynomials is guaranteed anyway. Thus, we can restrict the heuristics for the choice of $\preceq$ by computing the ordering on the set $\tilde{\Xi}$ containing for each polynomial only the closest lower and upper roots to s_i, and extending an ordering $\tilde{\preceq}$ of $\tilde{\Xi}$ to an ordering $\preceq$ on Ξ. This is formalized in the following definition.

Definition 5.3 *(Choice of the indexed root ordering: reduction).* Let i, P, s, Ξ be as in Definition 5.2. Then

$$\begin{aligned}\tilde{\Xi} = \quad & \{\xi \in \Xi \mid \xi(s_{[i-1]}) \leq s_i \wedge \forall \xi' \in \Xi \setminus \{\xi\}. \, (\xi'.p = \xi.p \implies \neg(\xi(s_{[i-1]}) < \xi'(s_{[i-1]}) \leq s_i))\} \\ & \cup \{\xi \in \Xi \mid s_i \leq \xi(s_{[i-1]}) \wedge \forall \xi' \in \Xi \setminus \{\xi\}. \, (\xi'.p = \xi.p \implies \neg(s_i \leq \xi'(s_{[i-1]}) < \xi(s_{[i-1]})))\}.\end{aligned}$$

For any indexed root ordering $\tilde{\preceq}_X$ on $\tilde{\Xi}$ matching $s_{[i-1]}$, we define the ordering $\preceq_X$ on Ξ matching $s_{[i-1]}$ as

$$\preceq_X = \tilde{\preceq}_X \cup \{(\xi, \xi') \mid \xi, \xi' \in \Xi \text{ such that } \xi.p = \xi'.p \wedge \xi(s_{[i-1]}) < \xi'(s_{[i-1]}))\}.$$

In the following definitions we give different heuristics to choose an indeed root ordering $\tilde{\preceq}_X$ which in each case can be extended to a corresponding ordering $\preceq_X$.

First, in the section case, the below EQUATIONAL CONSTRAINT heuristic enforces simply the application of the equational constraint rule to all polynomials.

Definition 5.4 *(*EQUATIONAL CONSTRAINT *representation).* Let i, P, s, Ξ, ξ_{lo}, ξ_{up}, $\mathtt{I}_{\mathrm{ldeg}}$ be as in Definition 5.2. If $\xi_{lo} = \xi_{up}$, we define the EQUATIONAL CONSTRAINT *representation* as the tuple $(\mathtt{I}_{\mathrm{ldeg}}, P, \emptyset)$.

Next, the BIGGEST CELL heuristic defines the weakest ordering on the indexed roots according to Definition 5.1 and thus defines the biggest possible underlying cell (under the assumption that this ordering can be realized perfectly, i.e. the resultants have roots only below the crossing of a polynomial's root with a cell boundary) as visualized in Fig. 9a. Note that for the section case, the BIGGEST CELL heuristic is the EQUATIONAL CONSTRAINT heuristic plus some discriminants and coefficients; thus, the application of BIGGEST CELL only makes sense in the sector case.

Definition 5.5 *(*BIGGEST CELL *representation).* Let i, P, s, Ξ, ξ_{lo}, ξ_{up}, $\mathtt{I}_{\mathrm{ldeg}}$ be as in Definition 5.2.

We define the indexed root ordering $\preceq_{\mathrm{biggest}}$ on Ξ according to

$$\begin{aligned}\tilde{\preceq}_{biggest} = \;& \{(\xi, \xi_{lo}) \mid \xi \in \tilde{\Xi} \setminus \{\xi_{lo}\} \text{ s.t. } \xi(s_{[i-1]}) \le \xi_{lo}(s_{[i-1]})\} \\ & \cup \{(\xi_{up}, \xi) \mid \xi \in \tilde{\Xi} \setminus \{\xi_{up}\} \text{ s.t. } \xi_{up}(s_{[i-1]}) \le \xi(s_{[i-1]})\}\end{aligned}$$

and the BIGGEST CELL *representation* as the tuple $(\mathtt{I}_{\mathrm{ldeg}}, \emptyset, \preceq_{\mathrm{biggest}})$.

The below LOWEST DEGREE BARRIERS heuristic minimizes the degrees of the defining polynomials (locally per level), and thus also the degree of the computed resultants (under the above assumption) as visualized in Fig. 9b. Furthermore, it enforces the equational constraints rule whenever possible.

This heuristic has two motivations. First, that the polynomial degrees grow doubly exponentially during the CAD projection, see e.g. Bradford et al. (2016, Table 1). Second, that the running time of the resultant computation depends quadratically on the degree in the main variable of the input polynomials, see Ducos (2000).

In the following definition, we use the *lexicographical ordering* on tuples, that is $t = (t_1, \ldots, t_k) < (t'_1, \ldots, t'_k) = t'$ holds if $t_1 < t'_1 \vee (t_1 = t'_1 \wedge t_2 < t'_2) \vee (t_1 = t'_1 \wedge t_2 = t'_2 \wedge t_3 < t'_3) \vee \ldots$.

Definition 5.6 *(*LOWEST DEGREE BARRIERS *representation).* Let i, P, s, Ξ, ξ_{lo}, ξ_{up}, $\mathtt{I}_{\mathrm{ldeg}}$ be as in Definition 5.2.

We assume an injection $o : \Xi \to \mathbb{N}$ that orders the elements of Ξ such that $o(\mathtt{I}_{\mathrm{ldeg}}.b) = 0$ respectively $o(\mathtt{I}_{\mathrm{ldeg}}.l) = 0$, $o(\mathtt{I}_{\mathrm{ldeg}}.u) = 1$ (if both are indexed root expressions), or $o(\mathtt{I}_{\mathrm{ldeg}}.l) = 0$ (if $\mathtt{I}_{\mathrm{ldeg}}.u = \infty$), or $o(\mathtt{I}_{\mathrm{ldeg}}.u) = 0$ (if $\mathtt{I}_{\mathrm{ldeg}}.l = -\infty$).

Let $\Xi' \subseteq \Xi$. For $\xi \in \Xi'$, we define the *barrier* of ξ w.r.t. Ξ', that is a root in Ξ' between $\xi(s_{[i-1]})$ and s_i with minimal degree in the main variable:

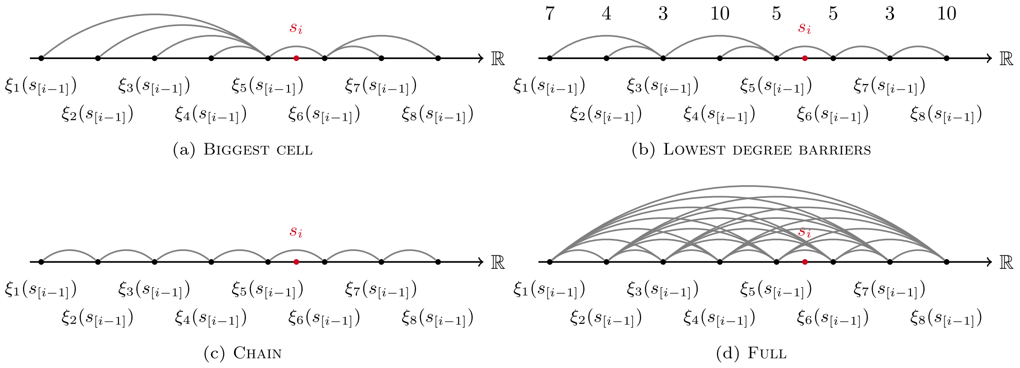
$$\begin{aligned}&\text{if } \xi(s_{[i-1]}) \le s_i && \text{then } \mathrm{barrier}_{\Xi'}(\xi) = \operatorname*{arg\,min}_{\{\xi' \in \Xi' \mid \xi(s_{[i-1]}) \le \xi'(s_{[i-1]}) \le s_i\}} (\deg_{x_i}(\xi'.p), s_i - \xi'(s_{[i-1]}), o(\xi'));\\ &\text{if } s_i \le \xi(s_{[i-1]}) && \text{then } \mathrm{barrier}_{\Xi'}(\xi) = \operatorname*{arg\,min}_{\{\xi' \in \Xi' \mid s_i \le \xi'(s_{[i-1]}) \le \xi(s_{[i-1]})\}} (\deg_{x_i}(\xi'.p), \xi'(s_{[i-1]}) - s_i, o(\xi')).\end{aligned}$$

If $\xi_{lo} \ne \xi_{up}$, we define the indexed root ordering $\preceq_{\mathrm{barriers}}$ on Ξ according to

$$\begin{aligned}\tilde{\preceq}_{barriers} = \;& \{(\xi, \mathrm{barrier}_{\tilde{\Xi}}(\xi)) \mid \xi \in \tilde{\Xi} \text{ s.t. } \xi(s_{[i-1]}) < s_i \text{ and } \xi \ne \mathrm{barrier}_{\tilde{\Xi}}(\xi)\}, \\ & \cup \{(\mathrm{barrier}_{\tilde{\Xi}}(\xi), \xi) \mid \xi \in \tilde{\Xi} \text{ s.t. } s_i < \xi(s_{[i-1]}) \text{ and } \xi \ne \mathrm{barrier}_{\tilde{\Xi}}(\xi)\}\end{aligned}$$

and the LOWEST DEGREE BARRIERS *representation* as the tuple $(\mathtt{I}_{\mathrm{ldeg}}, \emptyset, \preceq_{\mathrm{barriers}})$.

If $\xi_{lo} = \xi_{up}$, we exclude the polynomials with roots around s_i which do not qualify as a barrier for some other roots from the indexed root ordering (thus enforcing the application of the equational constraint rule). For a set of polynomials $P' \subseteq P$, we define $\tilde{\Xi}|_{P'} = \{\xi \mid \xi \in \tilde{\Xi} \text{ s.t. } \xi.p \in P'\}$ and let $P_{\mathrm{eq}} \subseteq P$ be the result of the following fixed point computation (i.e. $P_{\mathrm{eq}} = P_j = P_{j+1}$ for some j):



(a) Biggest cell

(b) Lowest degree barriers

(c) Chain

(d) Full

Fig. 9. Visualization of indexed root ordering heuristics.

$$P_0 = \emptyset$$

$$\begin{aligned} P_{j+1} = P_j \cup \{p \in P \mid p \neq \mathtt{I}_{\text{ldeg}}.b.p \\ \land \exists \xi \in \tilde{\Xi} \setminus \tilde{\Xi}|_{P_j}.\, (\xi.p = p \land \text{barrier}_{\tilde{\Xi}\setminus\tilde{\Xi}|_{P_j}}(\xi) = \mathtt{I}_{\text{ldeg}}.b) \\ \land \nexists \xi' \in \tilde{\Xi} \setminus \tilde{\Xi}|_{P_j}.\, \text{barrier}_{\tilde{\Xi}\setminus\tilde{\Xi}|_{P_j}}(\xi') = \xi)\}. \end{aligned}$$

We define the indexed root ordering $\preceq_{\text{barriers}}$ on $\Xi \setminus \Xi|_{P_{\text{eq}}}$ according the definition of $\tilde{\preceq}_{\text{barriers}}$ as above but only considering the roots $\tilde{\Xi} \setminus \tilde{\Xi}|_{P_{\text{eq}}}$ and the Lowest degree barriers *representation* as the tuple $(\mathtt{I}_{\text{ldeg}}, P_{\text{eq}}, \preceq_{\text{barriers}})$.

The below chain heuristic fixes the total ordering on the roots as visualized in Fig. 9c.

Definition 5.7 *(*Chain *representation).* Let i, P, s, Ξ, ξ_{lo}, ξ_{up}, $\mathtt{I}_{\text{ldeg}}$ be as in Definition 5.2.
Let $\{\xi_1, \ldots, \xi_k\} = \tilde{\Xi}$ s.t. $\xi_j(s_{[i-1]}) \leq \xi_{j+1}(s_{[i-1]})$ for all $j \in [1..k-1]$.
We define the indexed root ordering $\preceq_{\text{chain}}$ on Ξ according to

$$\tilde{\preceq}_{chain} = \{(\xi_j, \xi_{j+1}) \mid j \in [1..k-1]\}$$

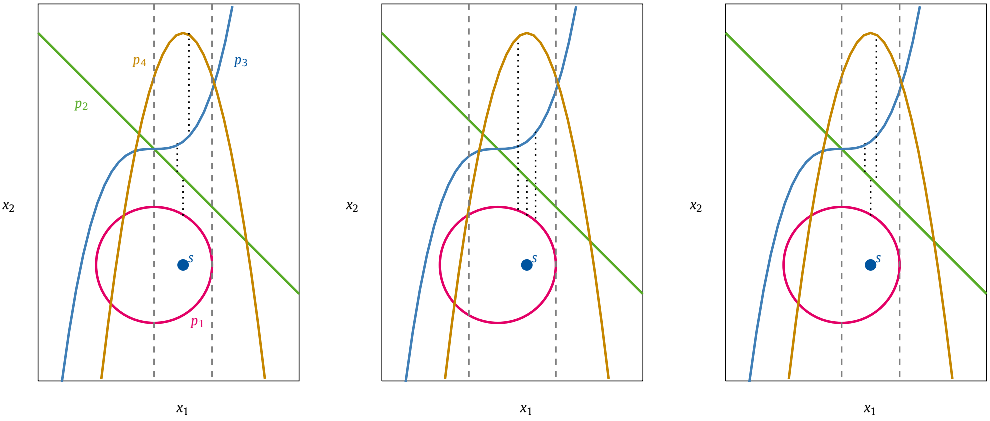
and the Chain *representation* as the tuple $(\mathtt{I}_{\text{ldeg}}, \emptyset, \preceq_{\text{chain}})$.

Finally, the full heuristic fixes the same total ordering as it is the unique transitive closure of the chain heuristic as visualized in Fig. 9d. Note that we include this heuristic only for illustrative purposes: it resembles the cells constructed naively by making a full projection without adapting to the behaviour at the sample.

Definition 5.8 *(*Full *representation).* Let i, P, s, Ξ, ξ_{lo}, ξ_{up}, $\mathtt{I}_{\text{ldeg}}$ as in Definition 5.2.
Let $\{\xi_1, \ldots, \xi_k\} = \tilde{\Xi}$ s.t. $\xi_j(s_{[i-1]}) \leq \xi_{j+1}(s_{[i-1]})$ for all $j \in [1..k-1]$.
We define the indexed root ordering $\preceq_{\text{full}}$ on Ξ according to

$$\tilde{\preceq}_{full} = \{(\xi_j, \xi_{j'}) \mid j, j' \in [1..k-1],\ j < j'\}$$

and the Full *representation* as the tuple $(\mathtt{I}_{\text{ldeg}}, \emptyset, \preceq_{\text{full}})$.



(a) CHAIN heuristic (b) BIGGEST CELL heuristic (c) LOWEST DEGREE BARRIERS heuristic

Fig. 10. Root ordering heuristics; the dotted lines indicate for which pairs of polynomials the resultant is added to the projection.

6. Qualitative observations

6.1. Comparing ordering heuristics

In this section we will explain the intuition behind the various ordering heuristics. They are designed to optimize desirable characteristics. We acknowledge their heuristic nature, and in particular that the intuition is made under the idealistic view that any indexed root ordering $\preceq$ can be perfectly realized: that is, the resultants calculated only have roots which indicate a crossing of two root functions considered in $\preceq$ when in reality they also have "spurious" roots which do not have relevance for the problem at hand.

From the FULL *to the* CHAIN *heuristic* As mentioned above, both the FULL and the CHAIN heuristic fix the same ordering on the real root functions over any cell containing the current sample as depicted in Fig. 10a.

Obviously, the CHAIN heuristic is more efficient here as it uses a strict subset of the work done by the FULL heuristic. Unlike the FULL heuristic, the CHAIN heuristic takes the underlying sample into account, and thus the resulting projection is only valid for a single cell, i.e. *locally delineable*. The FULL heuristic is independent from the sample and makes the set of polynomials fully delineable.

From the CHAIN *to the* BIGGEST CELL *heuristic* The CHAIN heuristic still fixes a stronger ordering on the root functions than necessary. As defined above, the BIGGEST CELL heuristic is the minimal requirement on the ordering to maintain sign-invariance on the constructed cell. This way, we hope that the size of the underlying cell is maximized, as in Fig. 10b. Note that while the CHAIN heuristic only takes the sample in one dimension less into account, the BIGGEST CELL heuristic also considers the highest dimension of the sample.

From the BIGGEST CELL *heuristic to the* LOWEST DEGREE BARRIERS *heuristic* The LOWEST DEGREE BARRIERS heuristic minimizes the degrees of the resultants, as illustrated in Fig. 10c. The rationale here is that resultant computations are heavy and their complexity depends on the degrees of the input polynomials, and that polynomials with lower degrees have fewer roots. Thus reducing degrees may reduce the case of resultants having real roots that do not actually correspond to relevant points. Note that despite the naming, the LOWEST DEGREE BARRIERS heuristic could in theory lead to bigger cells than the BIGGEST CELL heuristic in certain cases.

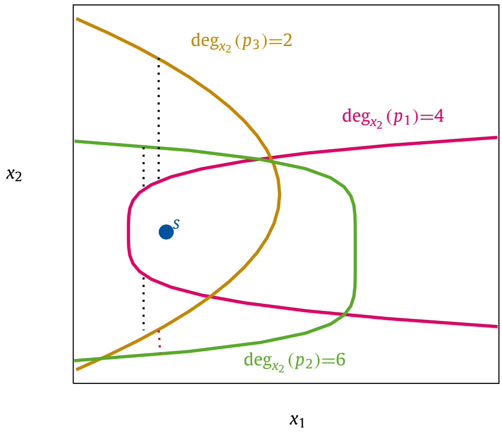


Fig. 11. Redundancies in the Lowest degree barriers heuristic.

Equational constraint projection The equational constraint projection allows us to leave out discriminants and coefficients of all polynomials except the section-defining polynomial by only adding resultants of all polynomials with the defining polynomial. Thus, we expect this rule to be more efficient than the presented heuristics in most cases. However, we added the possibility for the application of heuristics for cases where the section-defining polynomial has high degree and thus computing resultants with that polynomial is desirable to be avoided.

On the gap between the idealized view and reality For the lowest degree barriers heuristic, this might lead to the computation of a redundant set of resultants regarding the minimal requirement on the indexed root ordering, as shown in Fig. 11, where the relation between the first roots of p_2 and p_3 (depicted in red) is superfluous but its corresponding resultant is added because all roots are considered individually and not their connection via the defining polynomials.

6.2. Comparison with the refinement-based approach

The refinement-based approach to single cell construction of Brown and Košta (2015) saves resultants compared to full CAD projection in the same way as our levelwise variant, by exploiting the transitivity of the induced ordering on real root functions. Complexity-wise, the approaches are the same, as both add up to two resultants per polynomial on a level.

However, the influence on the ordering of constraints on the refinement-based variant means the quality of the constructed cell varies. In the worst case, the resulting ordering corresponds to the chain heuristic: in Fig. 10a, this is achieved for the example when the polynomials are merged in the ordering p_4, p_3, p_2, p_1. Then, the upper cell boundary is updated in every step to a lower boundary. In the best case, the biggest cell heuristic is achieved: for our example in Fig. 10b, this is when the polynomial p_1 defining the sector's boundary is merged first.

For the section case, it should also be noted that the levelwise approach can always apply the equational constraints rule as the cell description is known before the projection. The refinement-based approach only starts applying the equational constraints rule when the cell collapses to a section, until then it adds discriminants of all polynomials, which may not actually be needed. The illustrating examples may be used to explain this observation. Consider Fig. 10a but with the sample s moved to the upper root of p_1. When merging polynomials in the ordering p_4, p_3, p_2, p_1, the polynomials p_4, p_3, p_2 are merged as in the sector case until the cell collapses to a section when merging p_1. When merging p_1 first, then the section case is identified directly and the reduced projection applied when merging p_4, p_3, p_2, meaning the discriminants and coefficients for those polynomials need not be added.

6.3. Potential for non-connected "cell" descriptions

Recall that we aim to compute a single cell on which the input polynomials are sign-invariant, but McCallum's CAD projection theory uses the stronger property of order-invariance. It has been observed before that this can allow for small optimizations at the top layer. For example, when using equational constraint projection we must take discriminants if the projection is in a middle layer (to ensure order-invariance is provided which is a hypothesis of the next lifting) but can avoid these at the top layer as we need never lift over that (England et al., 2020).

The visualization of our proof rules in Fig. 8 alerted us to another such optimization, which if enacted has some strange consequences. Note that a cell being connected at level i is required for it to be order-invariant, but not sign-invariant. Thus, we need not ensure connectivity at the top level, i.e. we could avoid taking resultants of the upper and lower bound polynomials to satisfy Rule 4.11. Without connectedness, it would not be accurate to describe what we are constructing as a cell, rather it is a semi-algebraic set (see also Fig. 7). However, it is still describing a portion of space on whose points the respective polynomials are all sign-invariant. Thus, in the context of the MCSAT search which we discuss in the next section, the set is still describing a portion of space on whose points the constraints are all unsatisfiable for the same reason and so its negation is still a valid explanation clause to further that search.

We note that the resultants saved by this optimization may still have to be computed later, if the cell is used in further propagation. However, this will not always be the case and so often this optimization may save computation. Discovery of this optimization illustrates the advantages of the proof system presentation used in this paper.

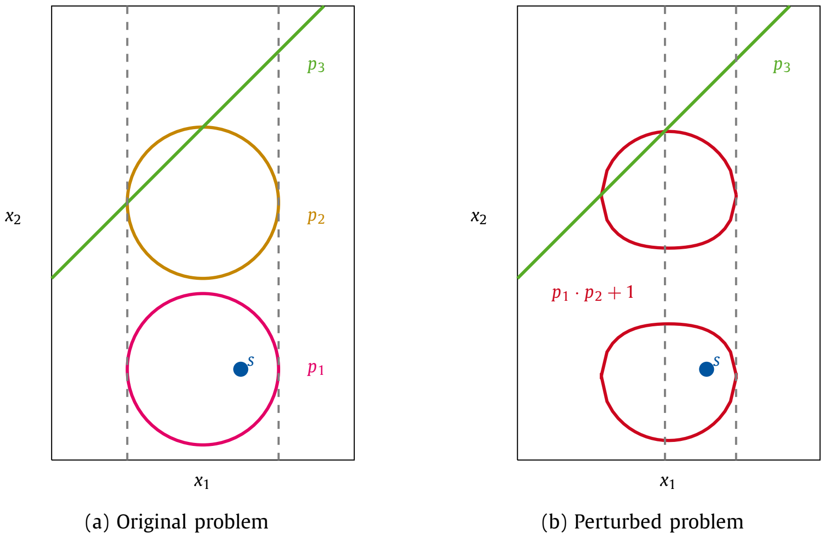
6.4. Factorization of polynomials and cell size

To ensure the output of CAD is correct we must compute a *square-free basis* of the current set of polynomials P (i.e. a set of square-free polynomials without common factors which define the same varieties as P) before the application of a projection operator.

The approach of the rules presented above is to fully factorize each polynomial, resulting in what is called the *finest* square-free basis. This is a fairly standard choice made in CAD implementations as the effort of computing a full factorization pays off compared to the heavy resultant, discriminant, and real root isolation computations, which are all simpler for smaller polynomials.

When building a full CAD the choice of a square-free basis does not affect the decomposition computed, just the time taken to compute it. However, for the single-cell construction, this makes a difference also in the size of the resulting cell, specifically the cell can be larger if we factor. This can easily be observed by considering Example 3.1. Here polynomial $p_1 \cdot p_2 \cdot p_3$ is already square-free. Using this directly in the one-cell construction algorithm without factorization as a whole would simply result in computing the discriminant of the polynomial (and some coefficients): the discriminant must have as factors all the cross resultants of p_1, p_2, p_3 by definition. So in this case, no improvement over a full projection is achieved, and we would find the smaller cell from Fig. 1. However, if we factor to consider the set $\{p_1, p_2, p_3\}$ instead then we can obtain the larger cell from Fig. 1.

The limits of factorization We note that this described gain in size of the computed cell from factorization does not mean we build optimal cells for all problems where there is a geometric separation. Fig. 12a shows an example with two circles and a linear polynomial for which the constructed cell we ideally build is the inside of the circle defined by p_1. If we were originally presented with $p_1 \cdot p_2$, then factorization would allow a one-cell algorithm to ignore the intersections of p_2 and p_3 to construct the entire inner circle. But consider the similar example from Fig. 12b, in which we have perturbed the problem to consider the irreducible polynomial $p_1 \cdot p_2 + 1$. Graphically, the two problems seem to be similar, and the delineation of the roots are identical. But in the second case the single cell construction cannot treat the two ovals separately: we must consider the irrelevant intersection and thus build a smaller than ideal cell.



(a) Original problem

(b) Perturbed problem

Fig. 12. By perturbing a problem making a reducible polynomial irreducible, the problem gets harder.

This shows some limits of the single cell construction by way of a well observed truth in computer algebra: problems which appear similar to human beings (i.e. when viewed geometrically) are not always equally hard algebraically.

7. Experimental evaluation

The presented proof rules are, due to their generality, potentially applicable with only small additions to a variety of problems and algorithms related to non-linear arithmetic, such as quantifier elimination by CAD (Collins, 1975), various CAD optimizations e.g. (McCallum, 1999; Collins and Hong, 1991), non-uniformly cylindrical decompositions for quantifier elimination (Brown, 2015, 2017), the generation of explanations when using cylindrical algebraic coverings for a traditional SMT solver (Ábrahám et al., 2021), and the generation of explanations in MCSAT (Jovanović and de Moura, 2012; Jovanovic et al., 2013; Brown and Košta, 2015). As the latter was the main motivation for our work, the evaluation of this paper will focus on generating theory explanations in MCSAT for non-linear arithmetic.

7.1. Generating explanations for MCSAT

Recall the description of MCSAT in Section 1.2. We are interested in when MCSAT resolves theory conflicts. I.e. when there is a set of constraints C in real variables $x_1, \ldots, x_n, x_{n+1}$ that should be satisfied according to the Boolean model and an assignment $s : \{x_1, \ldots, x_n\} \to \mathbb{R}$ such that s cannot be extended to a value for x_{n+1} satisfying C. The task then is to exclude a cell around s that generalizes this conflict, i.e. a region cell where the reason for unsatisfiability of C is invariant.

This reason of unsatisfiability is maintained when all input polynomials are sign-invariant on the generalized cell. To achieve this, we could do a full McCallum projection step, obtaining a set of properties of one level below allowing to construct a cell around s.

However, this is already too strong, as we need the set of input polynomials $P = \{p \mid (p \sim 0) \in C\} \subset \mathbb{Q}[x_1, \ldots, x_n, x_{n+1}]$ to be delineable over a cell *containing the current sample* $s \in \mathbb{R}^n$ for maintaining the desired property. We achieve this by determining the indexed root expression of the real roots of the set $\{p \in \text{factors}(P) | \, \text{level}(p) = n+1\}$ over s in x_{n+1} and ordering them such that $\xi_1(s) \leq \xi_2(s) \leq \ldots \leq \xi_k(s)$. Finally, we ensure that all lower level factors $\{p \in \text{factors}(P) | \, \text{level}(p) < n+1\}$ are sign-invariant, check that each polynomial is not nullified (if not, we stop), make each polynomial individually delineable and add the resultants of the pair of polynomials $(\xi_j.p, \xi_{j+1}.p)$ for $j \in [1..k-1]$. Thus, the input of the presented one-cell algorithm is given as

$$\begin{aligned} Q = &\{sgn\_inv(p) \mid p \in \text{factors}(P),\ \text{level}(p) < n+1\} \\ &\cup \{an\_del(p) \mid p \in \text{factors}(P),\ \text{level}(p) = n+1\} \\ &\cup \{ord\_inv(\text{res}_{x_{n+1}}(\xi_j.p, \xi_{j+1}.p)) \mid j \in [k-1]\}. \end{aligned}$$

This approach is similar to the chain heuristic for indexed root orderings presented in Definition 5.7.

Note that this approach could be embedded nicely into our system as a proof rule, taking over some optimizations. Furthermore, there are alternative approaches for elimination in the first levels, i.e. by computing a covering of unsatisfying intervals of input constraints. These possibilities are part of our plans for future work.

Further, note that in MCSAT, conflicts might also depend on previously computed cells which are expressed by conjunctions of *extended constraints* where a variable is compared with an indexed root expression; these constraints can also occur in the input. To handle an extended constraint $x_{n+1} \sim \text{root}_{x_{i+1}}[p, j]$ with $p \in \mathbb{Q}[x_1, \ldots, x_{n+1}]$, we simply add p to the set P of input polynomials.

7.2. Implementation

For the evaluation of the presented algorithm, we employ the SMT-RAT (SMT-RAT, 2023; Corzilius et al., 2015) solver, which provides an *MCSAT* engine allowing the combination of multiple explanation backends. Several incomplete and complete methods are combined in the sense that these backends are called sequentially until one returns an explanation.

Currently available backends are the *Fourier-Motzkin variable elimination (FM)* (Jovanovic et al., 2013), *interval constraint propagation (ICP)* (Kremer, 2019), *virtual substitution (VS)* (Ábrahám et al., 2017), the complete model-based CAD cell construction algorithm from *NLSAT* (Jovanović and de Moura, 2012) using Collin's projection operator as well as the refinement-based single cell construction algorithm (Brown and Košta, 2015). Furthermore, SMT-RAT employs a fully dynamic activity-based variable ordering heuristic for scheduling theory variable assignments and Boolean decisions (Nalbach et al., 2019).

All variants of the presented levelwise one-cell construction algorithm are implemented as backends in SMT-RAT. This is a preliminary implementation not exploiting the full power of the proof system, in particular, it is not checked whether a property is already implied by some other properties in the projection.

For the evaluation, we compare the following solver variants:

NL The model-based projection using Collin's operator from *NLSAT* (Jovanović and de Moura, 2012) as a complete explanation backend. I.e. to use when one of the following variants which are all based on McCallum projection hits a nullification which they cannot handle (see Definition 2.2).

OC-* The refinement-based one-cell construction algorithm (Brown and Košta, 2015). We use the same MCSAT embedding as described above. Furthermore, the refinement-based method is able to return an explanation when a polynomial is nullified in the sector case in some special cases which our levelwise approach cannot handle yet; for better comparability, these special cases are excluded from the following tests. In case of failure, the complete explanation from *NLSAT* is called. To further specify the algorithm's behaviour and make it reproducible, we implemented heuristics for the order of merging of initial polynomials. These specify the * in the variant name as follows.

ASC The merge-operation is called on the initial polynomials in *ascending* order by their total degree.

DSC The merge-operation is called on the initial polynomials in *descending* order by their total degree.

LW-*-* The new levelwise one-cell construction algorithm with different heuristics applied in the section and sector case. In case of failure, the complete explanation from *NLSAT* is called. The first * in the variant name dictates the employed heuristic for the section case and the second for the sector case. Possible substitutions for the stars are as follows.

EQ	EQUATIONAL CONSTRAINT heuristic is applied (only for section case).
BC	BIGGEST CELL heuristic is applied (only for sector case).
CH	CHAIN heuristic is applied.
LDB	LOWEST DEGREE BARRIERS heuristic is applied.

[solver]+ with *[solver]* being one of the solver variants above, uses the FM-, ICP- and VS-based backends serially, in this order, before resorting to *[solver]*. Furthermore, we apply general preprocessing to the input before calling the main solver (Corzilius et al., 2015).

All variants are executed on the SMT-LIB benchmark library (Barrett et al., 2010) for quantifier-free non-linear real arithmetic, abbreviated as QF_NRA. This set contains 11552 problem instances.

The machine used for testing has four 2.1 GHz AMD Opteron CPUs with 12 cores each. In the created test series, each instance was solved with 15 minutes timeout and 6 GB of memory.

For reproducibility, the implementation which generated the following results is available at https://doi.org/10.5281/zenodo.5764569.

7.3. Results

Examined solvers First of all, we observed that `OC-ASC` solves as many instances as `OC-DSC` but needs slightly less time; thus, we omit `OC-DSC`. Furthermore, we observe that `LW-EQ-CH` and `LW-EQ-LDB` solve more instances than `LW-CH-CH` and `LW-LDB-LDB`. In our basic implementation, saving leading coefficients and discriminants pays off compared to the other heuristics. Thus, for further examination, we focus on the `LW-EQ-⋆` variants which always use the equational constraints projection in the section case. A brief summary of solved instances of all solvers can be seen in Table 1. Furthermore, from now on, `VB-LW` (respectively `VB-LW+`) is the virtual best of the `LW-EQ-⋆` (respectively `LW-EQ-⋆+`) solvers. `VB` and `VB+` are the corresponding virtual bests with respect to `LW-EQ-⋆` and `OC-ASC`.

General observations Before we compare the different approaches, we make some general comments on the results based on exemplary solvers in Table 2.

As already observed in Table 1 and confirmed by Table 2, large parts of the benchmark set are relatively easy. Around half of the benchmarks do not involve a single explanation call; and when enabling the additional incomplete backends, 79% of the benchmarks can be solved without a single call to the single cell construction. Considering the total number of explanation calls made, only 3.14% of them use the single cell construction for the `VB-LW+` solver, meaning that even for the problems where single cell is needed, it is only needed rarely.

The fail rate of the levelwise backend (the cases where a nullification occurs) is smaller on the `VB-LW+` solver (6%) than on the `VB-LW` solver (17.98%). That means, that nullifications are more probable on the simple parts of the problem.

It should also be noted that `VB-LW+` needs significantly less explanation calls than `VB-LW`; which means, that the incomplete backends have explanations of higher quality.

To summarize, we can only make meaningful statements on our heuristics based on the solver variants without the additional backends, as otherwise, there are too few calls to the single cell construction in solved instances. The possible reasons for this are twofold. On the one hand, our procedure and its implementation may not yet be suitable to solve the harder instances in the benchmark set. On the other hand, the benchmark set might not contain enough interesting or diverse benchmarks for an evaluation.

Overall results All solvers are depicted in the performance profile in Fig. 13.

Table 1
Details on the instances solved by each solver: the number solved (first column), the number of satisfiable and unsatisfiable solved instances (the second and third columns) and the number of solved instances where the solver made at least one call to the single-cell construction and the number that did not require any such call to solve (the final two columns). Note that the last column is, for the first block of solvers, the number of instances that may be solved with Boolean reasoning and construction of sample points alone, i.e. without any theory calls. For the second block of solvers this also includes the instances that are solved using the additional incomplete theory backends.

	solved	sat	unsat	with single cell	without single cell
NL	9057	4518	4539		
OC-ASC	9316	4605	4711	4194	5122
OC-DSC	9316	4605	4711	4194	5122
LW-EQ-BC	9308	4598	4710	4186	5122
LW-EQ-CH	9311	4602	4709	4189	5122
LW-EQ-LDB	9304	4595	4709	4182	5122
VB-LW	9403	4650	4753	4281	5122
VB	**9435**	**4671**	**4764**	**4313**	5122
LW-CH-CH	9305	4600	4705	4183	5122
LW-LDB-LDB	9296	4585	4711	4174	5122
NL+	9535	4699	4836		
OC-ASC+	9797	4788	5009	615	9182
LW-EQ-BC+	9795	4786	5009	613	9182
LW-EQ-CH+	9797	4787	5010	615	9182
LW-EQ-LDB+	9795	4786	5009	613	9182
VB-LW+	9797	4787	5010	615	9182
VB+	**9800**	**4790**	**5010**	**618**	9182
Total	11552*	5069*	5379*		

* For 1104 instances, it is not known whether they are satisfiable or unsatisfiable.

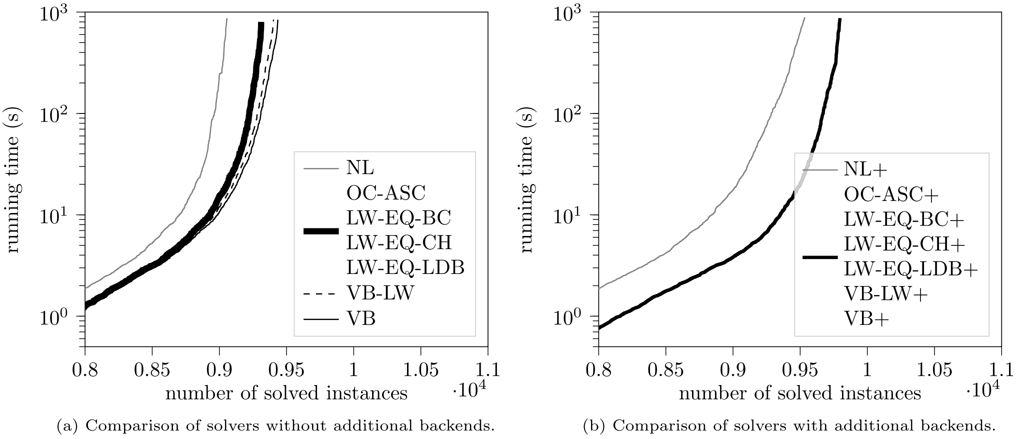
Table 2
Comparison of four exemplary solvers: firstly, the subset of all solved instances and secondly the subset of all solved instances where at least one call to the single cell construction was made are considered. For each, the statistics describing the total number of instances, the mean running time, the sum of constructed cells as well as the sum of attempts to generate a cell using single cell construction or NLSAT backend are given. For the last two, these sums are included in the previous one; and their share of the total number of constructed cells is given. Note that the explanation calls involving single cell are the ones not being solved by the additional backends and the calls involving NLSAT are the ones where the single cell construction failed.

	LW-EQ-BC	LW-EQ-BC+	VB-LW	VB-LW+
solved instances	9308	9795	9403	9797
mean running time (s)	4.36	5.22	5.56	5.26
sum calls to explanation	83628	58826	107964	58953
of them: sum calls to SC	83628 (100.0%)	1740 (2.96%)	107964 (100.0%)	1849 (3.14%)
of them: sum calls to NL	15394 (18.41%)	117 (0.2%)	19412 (17.98%)	119 (0.2%)
solved instances with single cell	4186	613	4281	615
mean running time (s)	9.13	20.34	11.7	21.81
sum calls to explanation	83628	10363	107964	10490
of them: sum calls to SC	83628 (100.0%)	1740 (16.79%)	107964 (100.0%)	1849 (17.63%)
of them: sum calls to NL	15394 (18.41%)	117 (1.13%)	19412 (17.98%)	119 (1.13%)

First of all, the NL and NL+ solvers perform significantly worse than the single cell variants, which justifies the investigation of this new levelwise cell construction approach.

The refinement-based approach OC-ASC as well as LW-EQ-⋆ perform similarly. Among the levelwise variants, the LW-EQ-CH solves the most; however, these differences are not significant and depend on the implementation and heuristics chosen in the MCSAT solver. Considering OC-ASC+ and LW-EQ-⋆+, these differences vanish even more when combining them with incomplete methods handling simple sub-problems. These results are summarized in Table 1.

Virtual best and orthogonality of heuristics VB-LW performs significantly better than the LW-EQ-⋆ solvers. This means that although the number of solved instances is similar for all heuristics, each



(a) Comparison of solvers without additional backends.

(b) Comparison of solvers with additional backends.

Fig. 13. Performance profile (running times).

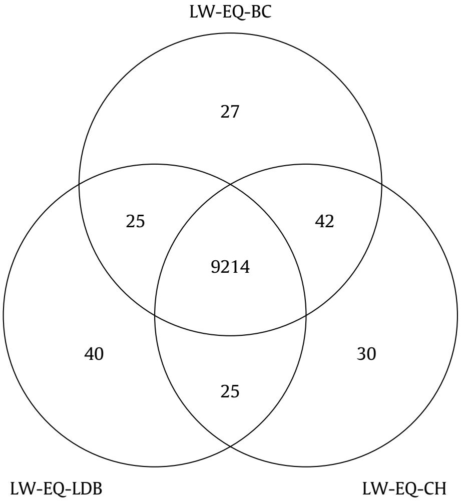


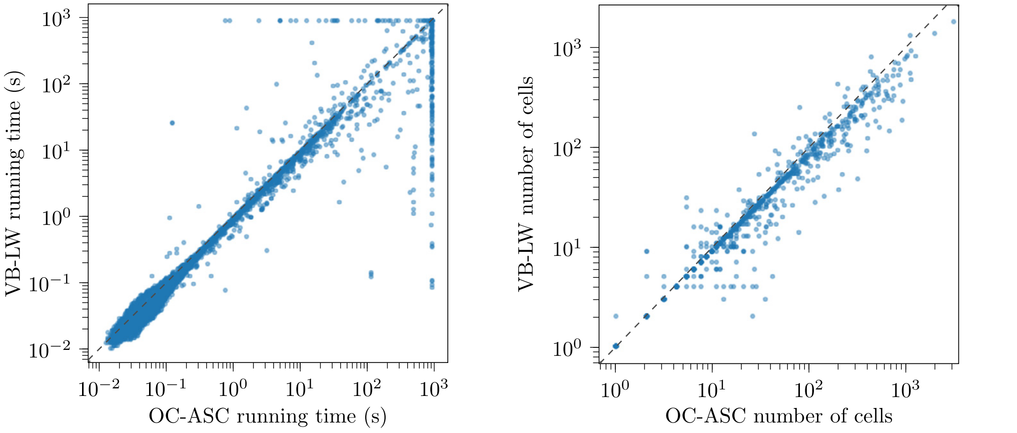
Fig. 14. Commonly solved instances.

heuristic solves instances that the others do not solve. Thus these heuristics are *orthogonal* to some degree. We depict the number of instances solved by differing combinations of solvers in Fig. 14. Note that the same holds for VB, meaning that OC-ASC is orthogonal to LW-EQ-⋆ as well.

Note that the VB-LW+ solver does not solve significantly more instances than any of the LW-EQ-⋆+ solvers. That means the differences of the heuristics only become noticeable in the simple parts of the instances. An explanation could be that harder parts of instances require heavy resultant computations which could quickly shift the instance to unsolvable within the timeout.

Now focusing on the “pure” solvers without additional backends, we observe that on simple instances the refinement based approach OC-ASC and the virtual best of the levelwise approaches VB-LW behave similar on simple instances while they are more orthogonal on harder instances, as indicated by Fig. 15a.

The SMT-LIB benchmark set is split up into families which each have a similar structure. We note that all variants LW-EQ-⋆ and OC-ASC solve roughly the same amount of benchmarks in each family individually, as seen in Table 3, while the virtual best solvers do solve more. That means that the orthogonality of the variants is split up over the various benchmark families.



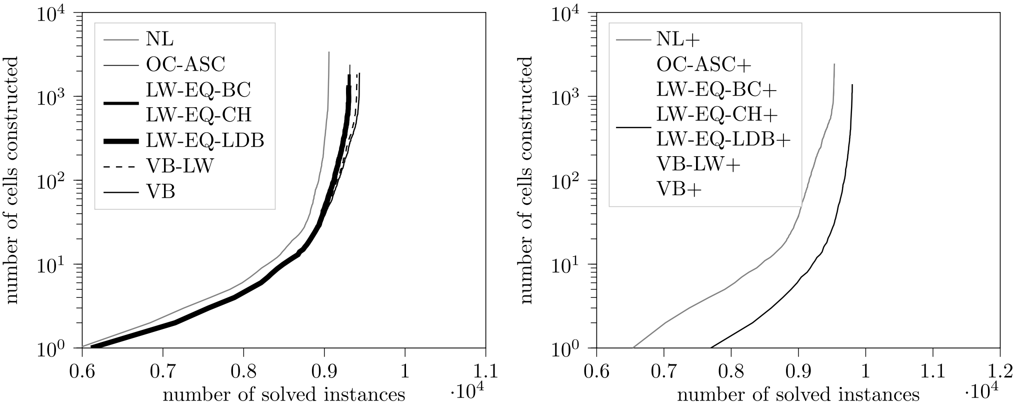
(a) Running times on all instances.

(b) Number of constructed cells on instances solved by both solvers.

Fig. 15. Scatter plots comparing `VB-LW` and `OC-ASC` on the individual instances.

Table 3
Solved instances by solver and family. SMT-LIB benchmark families: (1) 20161105-Sturm-MBO (2) 20161105-Sturm-MGC (3) 20170501-Heizmann-UltimateInvariantSynthesis (4) 20180501-Economics-Mulligan (5) 2019-ezsmt (6) LassoRanker (7) UltimateAutomizer (8) hong (9) hycomp (10) kissing (11) meti-tarski (12) zankl.

	1	2	3	4	5	6	7	8	9	20	11	12
OC-ASC	25	0	0	86	9	2	26	6	2180	8	6913	61
LW-EQ-BC	22	0	0	89	9	2	22	6	2170	8	6918	62
LW-EQ-CH	25	0	0	88	9	3	22	5	2174	8	6914	63
LW-EQ-LDB	22	0	0	88	9	2	22	7	2170	8	6913	63
VB-LW	26	0	0	90	9	4	25	7	2249	8	6918	67
VB	27	0	0	94	9	4	28	7	2267	8	6923	68
Total	405	9	69	135	63	821	61	20	2752	45	7006	166



(a) Comparison of solvers without additional backends.

(b) Comparison of solvers with additional backends.

Fig. 16. Performance profile (number of cells).

Number of constructed cells Fig. 16 depicts a performance profile where we consider the number of constructed cells (that is, the number of times the explanation function is called by MCSAT) instead of the running time. We can clearly see that the solvers that solve more instances tend to compute

Table 4
Statistics on the 3226 instances solved by all three solvers with at least one call to the single cell construction and without a call to the NLSAT explanation (single cell construction always succeeds): mean number of cells per instance, mean dimension of constructed cells, mean maximum degree of polynomials occurring in an instance, mean number of resultants/discriminants/coefficients in the projection per instance.

	mean #cells per inst.	mean dim. of cells	mean max deg. per inst.	mean #res. per inst.	mean #disc. per inst.	mean #coeff. per inst.
LW-EQ-BC	4.38	8.59	**6.51**	8.95	13.94	13.62
LW-EQ-CH	4.39	8.52	6.52	9.07	14.04	13.80
LW-EQ-LDB	4.39	**8.48**	6.57	9.07	14.05	13.88
VB-LW	**4.32**	8.49	6.52	**8.77**	**13.75**	**13.48**

fewer cells during a run. That is, the quality of the explanations is better. Again, considering the refinement based approach and the levelwise approaches, differences are only significant without the other backends. In particular, the virtual best solver clearly solves more instances with fewer cells. Fig. 15b makes this even more clear by comparing the number of cells constructed by `VB-LW` and `OC-ASC` for every instance.

Comparison of heuristics For further comparison of the three heuristics, we collected more statistics for each instance: the dimension of constructed cells (i.e. the number of levels considered during its construction), the maximum degree in the main variable of the polynomials occurring in the computation, and the size of the computed projection (i.e. number of resultants, discriminants and coefficients). A summary of these statistics is shown in Table 4.

First note that the virtual best has the lowest value or close to the lowest value in all categories, which means that they might indicate the performance of a solver to some degree. However, interpretation needs to be careful, as the differences are relatively small, confirming the previous observations. Further, note that we do consider fewer instances as in previous analysis. However, we do have two minor observations to make here.

Regarding the heuristic `LW-EQ-LDB`: its idea was to minimize the degrees of the polynomials in the projection, however, the average maximum degree is slightly higher than for the other two heuristics. `LW-EQ-BC` needs slightly less projection steps, but is similar to the other solvers in terms of number of cells created. This means that the sizes of the cells are likely similar, although again, the intention was to produce the biggest possible cells. To emphasize, in the analysis in Fig. 16 which is based on all solved instances (including the ones where the NLSAT backend was used as fallback), it needs more cells than `LW-EQ-LDB`.

To summarize, this simple analysis can not confirm that the ideas behind the different heuristics take effect on the benchmarks or prove different behaviour on all benchmarks. However, as stated above, on individual benchmarks, they do behave differently, as shown by the performance of the virtual best solver.

8. Conclusions and future work

8.1. Future work

The formulation of the presented proof system allows heuristic to influence the shape of the constructed cells. The experimental evaluation shows potential for further development of those. Furthermore, for some applications such as incremental linearization (Cimatti et al., 2017), under-approximations of the constructed cell might be beneficial if they can be computed more efficiently; more concretely, the resultant computations get trivial if the lower and upper bounds are replaced by one or more linear polynomials, resulting in boxes or polyhedra.

The proof system could be applied in the future in contexts other than MCSAT, such as the *cylindrical algebraic coverings method* (Ábrahám et al., 2021) or quantifier elimination algorithms such as *NuCAD* (Brown, 2015, 2017).

By relying on the theory of McCallum projection, the presented proof system is incomplete. Recently, there has been progress on the Lazard projection operator (Lazard, 1994; McCallum and Hong, 2016; McCallum et al., 2019; Brown and McCallum, 2020; Nair et al., 2019, 2020), which is complete while maintaining the advantages of the McCallum operator. Thus, it is promising to extend our framework to exploit the Lazard projection.

As indicated in Figs. 5b and 6b, the notion of delineability is stronger than we need for single cell construction. We will investigate the role of the leading coefficients and the zeros of a resultant.

Finally, the presented set of inference rules allows producing fine-grained proof graphs, which could enable the certification and external automated verification of results.

8.2. Conclusion

We introduced the new concept of levelwise single cell construction, motivated by maintaining the savings of the existing refinement based approach of Brown and Košta (2015) while allowing for more flexibility and new optimizations.

The theoretical part of this paper consists of a proof system in order to enable fine-grained projections based on a given sample. To demonstrate the possibilities of the proof system, we gave a simple algorithm which builds upon this proof system as well as several heuristics for the application of the given rules. Finally, we gave a qualitative evaluation as well as some notable observations.

We evaluated our algorithm by an implementation applied to explanation generation in MCSAT. We showed that our basic heuristics yield different performances for different instances indicating that there is room for further algorithmic development, as well as a more elaborated implementation of the proof rules.

Importantly, our proof system allows for a wide range of improvements through experimentation with heuristics, approximations and on theoretical matters as well as extensions to other algorithms than the single cell construction.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Data availability

Data will be made available on request.

Acknowledgements

Jasper Nalbach was supported by the DFG RTG 2236 *UnRAVeL*. James Davenport and Matthew England were supported by the EPSRC DEWCAD Project, *Pushing Back the Doubly-Exponential Wall of Cylindrical Algebraic Decomposition* (grant references EP/T015748/1 and EP/T015713/1).

Appendix A. Correctness of the proof system

Note that throughout this section, when proving the correctness of a mapping, we implicitly use the assumptions from the mapping's definition in the proofs.

Rule 4.1. Let $i \in \mathbb{N}$, $R \subseteq \mathbb{R}^i$, and $p \in \mathbb{Q}[x_1, \ldots, x_{i+1}]$, $\operatorname{level}(p) = i + 1$. Assume that p is irreducible.

$$\mathit{an\_sub}(i)(R),\ \mathit{connected}(i)(R),\ \mathit{non\_null}(p)(R),\ \mathit{ord\_inv}(\operatorname{disc}_{x_{i+1}}(p))(R),$$
$$\mathit{sgn\_inv}(\operatorname{ldcf}_{x_{i+1}}(p))(R) \qquad \vdash \mathit{an\_del}(p)(R)$$

Lemma A.1. *Rule 4.1 on page 14 is correct.*

Proof. Follows immediately from Brown (2001, Theorem 3.1) and McCallum (1998, Theorem 2). □

Rule 4.2. Let $i \in \mathbb{N}$, $R \subseteq \mathbb{R}^i$, $s \in \mathbb{R}^i$, and $p \in \mathbb{Q}[x_1, \ldots, x_{i+1}]$, $\text{level}(p) = i+1$. Assume that p is irreducible, and $p = c_m \cdot x_{i+1}^m + \ldots + c_1 \cdot x_{i+1} + c_0$ such that $c_m, \ldots, c_0 \in \mathbb{Q}[x_1, \ldots, x_i]$.

$$\mathit{sample}(s)(R),\ \deg_{x_{i+1}}(p) > 1,\ \text{disc}_{x_{i+1}}(p)(s) \neq 0,\ \mathit{sgn\_inv}(\text{disc}_{x_{i+1}}(p))(R) \quad \vdash \mathit{non\_null}(p)(R)$$
$$\mathit{sample}(s)(R),\ \exists j \in [m].\ (c_j(s) \neq 0 \wedge \mathit{sgn\_inv}(c_j)(R)) \quad \vdash \mathit{non\_null}(p)(R)$$

Lemma A.2. *Rule 4.2 on page 14 is correct.*

Proof. The first rule follows from the definition of the discriminant by the Sylvester matrix.

For the second rule, observe that p is nullified on any point $r \in R$ if and only if $c_j(r) = 0$ for all $j \in [m]$. Then, the statement follows immediately. See also (Brown and Košta, 2015, Lemma 5). □

Rule 4.3. Let $p \in \mathbb{Q}$.

$$\vdash \mathit{ord\_inv}(p)(\mathbb{R}^0)$$
$$\vdash \mathit{sgn\_inv}(p)(\mathbb{R}^0)$$

Lemma A.3. *Rule 4.3 on page 14 is correct.*

Proof. Trivial. □

Rule 4.4. Let $i \in \mathbb{N}_{>0}$, $R \subseteq \mathbb{R}^i$, and $p \in \mathbb{Q}[x_1, \ldots, x_i]$, $\text{level}(p) = i$. Assume that p is reducible, and $\text{factors}(p) = \{q_1, \ldots, q_k\}$.

$$\mathit{ord\_inv}(q_1)(R),\ \ldots,\ \mathit{ord\_inv}(q_k)(R) \quad \vdash \mathit{ord\_inv}(p)(R)$$
$$\mathit{sgn\_inv}(q_1)(R),\ \ldots,\ \mathit{sgn\_inv}(q_k)(R) \quad \vdash \mathit{sgn\_inv}(p)(R)$$

Lemma A.4. *Rule 4.4 on page 14 is correct.*

Proof. Note that if p is reducible, then $p \notin \text{factors}(p)$. Both statements follow by (McCallum, 1985, Lemma 3.2.2). □

Rule 4.5. Let $i \in \mathbb{N}_{>0}$, $R \subseteq \mathbb{R}^i$, $s \in \mathbb{R}^i$, and $p \in \mathbb{Q}[x_1, \ldots, x_i]$, $\text{level}(p) = i$. Assume that p is irreducible.

$$p(s) \neq 0,\ \mathit{sample}(s)(R),\ \mathit{sgn\_inv}(p)(R) \quad \vdash \mathit{ord\_inv}(p)(R)$$
$$p(s) = 0,\ \mathit{sample}(s)(R),\ \mathit{an\_sub}(i-1)(R),\ \mathit{connected}(i)(R),\ \mathit{sgn\_inv}(p)(R),\ \mathit{an\_del}(p)(R) \quad \vdash \mathit{ord\_inv}(p)(R)$$

Lemma A.5. *Rule 4.5 on page 14 is correct.*

Proof. In the first case, from $p(s) \neq 0$ and the sign-invariance of p on R follows $p(r) \neq 0$ for all $r \in R$. Thus, p is order-invariant on R as an immediate consequence of the definition of order-invariance.

In the second case, from $p(s) = 0$ and the sign-invariance of p on R follows $p(r) = 0$ for all $r \in R$. As p is analytically delineable on $R\downarrow_{[i-1]}$, p is order-invariant in each p-section on $R\downarrow_{[i-1]}$, R is a p-section on $R\downarrow_{[i-1]}$, thus the order-invariance of p on R follows immediately. □

Rule 4.6. Let $i \in \mathbb{N}$, $R \subseteq \mathbb{R}^i$, $s \in \mathbb{R}^{i-1}$, and $p \in \mathbb{Q}[x_1, \ldots, x_i]$, $\text{level}(p) = i$. Assume that p is irreducible, and $\text{realRoots}(p(s, x_i)) = \emptyset$.

$$\mathit{sample}(s)(R),\ \mathit{an\_del}(p)(R) \quad \vdash \mathit{sgn\_inv}(p)(R)$$

Lemma A.6. *Rule 4.6 on page 15 is correct.*

Proof. By assumption p is (analytically) delineable on a connected superset $R' \supseteq R{\downarrow_{[i-1]}}$, thus by Definition 2.1 the number of roots of p is constant over R'. As $\mathrm{realRoots}(p(s, x_{i+1})) = \emptyset$ and $s \in R{\downarrow_{[i-1]}} \subset R'$, p does not have any roots over R', thus p is sign-invariant on $R' \times \mathbb{R} \supset R$. □

Rule 4.7. Let $i \in \mathbb{N}_{>0}$, $R \subseteq \mathbb{R}^i$, $s \in \mathbb{R}^{i-1}$, and $\mathtt{I}$ be a symbolic interval of level i.

$$\begin{array}{ll} & \vdash \mathit{an\_sub}(0)(\mathbb{R}^0) \\ \mathit{repr}(\mathtt{I}, s)(R),\ \mathit{an\_sub}(i-1)(R) & \vdash \mathit{an\_sub}(i)(R) \end{array}$$

Lemma A.7. *Rule 4.7 on page 16 is correct.*

Proof. As the endpoints of $\mathtt{I}$ are described by analytic functions, the statement follows immediately from (McCallum, 1985, Theorem 2.2.3 and Theorem 2.2.4). □

Rule 4.8. Let $i \in \mathbb{N}_{>0}$, $R \subseteq \mathbb{R}^i$, $s \in \mathbb{R}^{i-1}$, $p \in \mathbb{Q}[x_1, \ldots, x_i]$, $\mathrm{level}(p) = i$, and $\mathtt{I}$ be a symbolic interval of level i. Assume that p is irreducible, and $\mathtt{I} = (\mathrm{section}, b)$.

Let $Q := \mathit{an\_sub}(i-1)(R) \wedge \mathit{connected}(i-1)(R) \wedge \mathit{repr}(\mathtt{I}, s)(R) \wedge \mathit{an\_del}(b.p)(R)$.

$$\begin{array}{ll} Q,\ b.p = p & \vdash \mathit{sgn\_inv}(p)(R) \\ Q,\ b.p \neq p,\ \mathit{ord\_inv}(\mathrm{res}_{x_i}(b.p, p))(R) & \vdash \mathit{sgn\_inv}(p)(R) \end{array}$$

Lemma A.8. *Rule 4.8 on page 16 is correct.*

Proof. As $\mathit{repr}(\mathtt{I}, s)$ holds on R, it holds $b.p(r) = 0$ for all $r \in R$.

In the first case, if $p = b.p$, p is trivially sign-invariant on R.

In the second case, if $p \neq b.p$, by the order-invariance of $\mathrm{res}_{x_i}(b.p, p)$ on $R{\downarrow_{[i-1]}}$, (McCallum, 1999, Theorem 2.2) yields that p is sign-invariant on R. □

Rule 4.9. Let $i \in \mathbb{N}$, $R \subseteq \mathbb{R}^i$, $s \in \mathbb{R}^i$, and $\preceq$ be an indexed root ordering of level $i+1$. Assume that $\xi.p$ is irreducible for all $\xi \in \mathrm{dom}(\preceq)$, and that $\preceq$ matches s.

$$\begin{array}{ll} \mathit{sample}(s)(R),\ \mathit{an\_sub}(i)(R),\ \mathit{connected}(i)(R),\ \forall \xi \in \mathrm{dom}(\preceq).\ \mathit{an\_del}(\xi.p)(R), & \\ \qquad \forall (\xi, \xi') \in \preceq.\ \mathit{ord\_inv}(\mathrm{res}_{x_{i+1}}(\xi.p, \xi'.p))(R) & \vdash \mathit{ir\_ord}(\preceq, s)(R) \end{array}$$

Lemma A.9. *Rule 4.9 on page 18 is correct.*

Proof. First consider $\xi \preceq \xi'$. Either $\xi.p = \xi'.p$, then $\theta_{\xi,s} = \theta_{\xi',s}$ by definition of delineability (Definition 2.1), or $\mathrm{res}_{x_{i+1}}(\xi.p, \xi'.p)$ is order-invariant on R, then either $\theta_{\xi,s} = \theta_{\xi',s}$ or $\theta_{\xi,s} < \theta_{\xi',s}$ on R holds by Theorem A.1.

Now consider $\xi \preceq^t \xi'$ (where $\preceq^t$ is the transitive closure of $\preceq$), then there exist $\xi_1, \ldots, \xi_k$ such that $\xi = \xi_1 \preceq \ldots \preceq \xi_k = \xi'$. We show that either $\theta_{\xi,s} = \theta_{\xi',s}$ or $\theta_{\xi,s} < \theta_{\xi',s}$ on R holds by induction. The base case for $k = 2$ is proven by the preceding paragraph. Assume that the statement holds for ξ_1 and ξ_j with $j < k$ (induction hypothesis), i.e. either $\theta_{\xi_1,s} = \theta_{\xi_j,s}$ or $\theta_{\xi_1,s} < \theta_{\xi_j,s}$ on R. Observe that $\theta_{\xi_j,s} = \theta_{\xi_{j+1},s}$ or $\theta_{\xi_j,s} < \theta_{\xi_{j+1},s}$ on R by the preceding paragraph. It follows by transitivity that $\theta_{\xi_1,s} = \theta_{\xi_{j+1},s}$ or $\theta_{\xi_1,s} < \theta_{\xi_{j+1},s}$ on R. □

Definition A.1 *(Degree invariance).* Let $i \in \mathbb{Z}$, $R \subseteq \mathbb{R}^i$, and $p \in \mathbb{Q}[x_1, \ldots, x_{i+1}]$ of level $i+1$. Then p is called *degree-invariant* on R if and only if $\deg_{x_{i+1}}(p(r, x_{i+1})) = \deg_{x_{i+1}}(p(r', x_{i+1}))$ for all $r, r' \in R$.

Theorem A.1. *Let* $i \in \mathbb{N}$, $R \subseteq \mathbb{R}^i$ *be a connected analytic submanifold and* $p_1, p_2 \in \mathbb{Q}[x_1, \ldots, x_{i+1}]$ *irreducible and coprime such that* $\mathrm{level}(p_1) = \mathrm{level}(p_2) = i+1$, $\theta_1, \theta_2 : R \to \mathbb{R}$ *be real root functions of* p_1 *and* p_2 *respectively, and* $\sim \in \{=, <, >\}$.

If p_1, p_2 are not nullified on any point in R and $\mathrm{res}_{x_{i+1}}(p_1, p_2)$ is order-invariant on R and p_1 and p_2 are analytically delineable on R, then $\theta_1(r) \sim \theta_2(r) \iff \theta_1(r') \sim \theta_2(r')$ for all $r, r' \in R$.

Proof. The hypotheses state that θ_1 and θ_2 are continuous and analytic real root functions over R. If the two functions are identical over R, then the theorem holds. So assume they are not, i.e. assume there is a point $s \in R$ at which the two functions have different values. We will show that the two functions differ everywhere in R which, because they are continuous, proves the theorem.

Let $s \in R$, $\alpha \in \mathbb{R}$ such that $\alpha < r'$ for every $r' \in \mathbb{R}$ such that $p_1(s, r') = 0$ or $p_2(s, r') = 0$ (we can choose such an α as p_1 and p_2 are delineable on R), and choose $S \subseteq R$ such that $s \in S$, and $\alpha > r'$ for every $r \in S$ and every $r' \in \mathbb{R}$ such that $p_1(r, r') = 0$ or $p_2(r, r') = 0$.

Let $p_1^* = p_1(x_1, \ldots, x_i, x_{i+1} - \alpha)$ and $p_2^* = p_2(x_1, \ldots, x_i, x_{i+1} - \alpha)$. The mapping that shifts x_{i+1} by α is a homeomorphism that maps the roots of p_1 and p_2 to the roots of p_1^* and p_2^*, respectively. So we can study the roots of p_1 and p_2 by understanding the roots of p_1^* and p_2^*. But for p_1^* and p_2^* we are guaranteed that $p_1^*(r, 0) \neq 0$ and $p_2^*(r, 0) \neq 0$ for all $r \in S$, i.e. the constant coefficients with respect to x_{i+1} of p_1^* and p_2^* do not vanish on S. Further, by Gelfand et al. (1994), the resultant is invariant under this transformation.

Let $\overline{p_1} = x_{i+1}^{d_1} p_1^*(x_1, \ldots, x_i, 1/x_{i+1})$ and $\overline{p_2} = x_{i+1}^{d_2} p_2^*(x_1, \ldots, x_i, 1/x_{i+1})$, where the $d_1 = \deg_{x_{i+1}}(p_1)$ and $d_2 = \deg_{x_{i+1}}(p_2)$.

Moreover, since neither p_1 nor p_2 is nullified in S, $\overline{p_1}$ and $\overline{p_2}$ are non-nullified in S. The transformation from p_1 and p_2 to $\overline{p_1}$ and $\overline{p_2}$ defines an analytic homeomorphism from $S \times (\mathbb{R} \setminus \{0\})$ to $S \times (\mathbb{R} \setminus \{\alpha\})$ that maps the zeros of $\overline{p_1}$ and $\overline{p_2}$ off the $x_{i+1} = 0$ hyperplane to the zeros of p_1 and p_2 (which do not cross the $x_{i+1} = \alpha$ hyperplane). Specifically, the homeomorphism is given by the mapping $(x, y) \mapsto (x, 1/y - \alpha)$. Moreover, the leading coefficients of $\overline{p_1}$ and $\overline{p_2}$, which are the same as the constant coefficients (in x_{i+1}) of p_1^* and p_2^*, are non-zero throughout S. This means that $\overline{p_1}$ and $\overline{p_2}$ are not nullified anywhere in S and their leading coefficients (in x_{i+1}) are non-vanishing on S, and thus $\overline{p_1}$ and $\overline{p_2}$ are degree-invariant on S. We note that $\overline{p_1}$ and $\overline{p_2}$ are delineable on S (as p_1 and p_2 are delineable on S and the homeomorphism); in particular, for every root function θ of p_1 or p_2, there exists a corresponding root function $\overline{\theta}$ of $\overline{p_1}$ or $\overline{p_2}$, respectively. Further, the resultant is unchanged by this transformation, once again by Gelfand et al. (1994), so $\mathrm{res}_{x_{i+1}}(\overline{p_1}, \overline{p_2})$ is order-invariant in S.

Now, let $\overline{\theta_1}$ and $\overline{\theta_2}$ be the root functions of $\overline{p_1}$ and $\overline{p_2}$ corresponding to θ_1 and θ_2. By Theorem A.2, $\overline{\theta_1}(s) \neq \overline{\theta_2}(s)$ implies $\overline{\theta_1}(r) \neq \overline{\theta_2}(r)$ for all $r \in S$; and thus, $\theta_1(r) \neq \theta_2(r)$ for all $r \in S$.

Since we can set s to any value in R, it follows that $\theta_1(r) \neq \theta_2(r)$ for all $r \in R$. □

Theorem A.2. *Let $p_1, p_2 \in \mathbb{R}[x_1, \ldots, x_i]$ of level i. Let R be a connected analytic submanifold in $\mathbb{R}^{i-1}$ on which p_1 and p_2 are degree-invariant and non-nullified, and on which $p_r = res_{x_i}(p_1, p_2)$ is order-invariant. Let θ_1 be a real root function of p_1 and θ_2 be a real root function of p_2, both defined on R. If there is a point $r \in R$ at which the values of θ_1 and θ_2 differ, then the values of θ_1 and θ_2 differ throughout R.*

Proof. We prove this theorem by contradiction. Suppose there are points at which θ_1 and θ_2 have the same value. Choose such a point $r' \in R$ and path $\pi : [0, 1] \to R$ such that $\pi(0) = r$, $\pi(1) = r'$ and for all $x \in (0, 1)$ we have $\theta_1(\pi(x)) \neq \theta_2(\pi(x))$. Note that we are guaranteed to be able to find such r' and π by the continuity of θ_1 and θ_2.

Let U be a neighbourhood of r' and V a neighbourhood of the origin, both within $\mathbb{R}^{i-1}$. Denote by d the dimension of R. We now refer to the concept of a *coordinate system*, as described by McCallum (1985). By Theorem 2.2.1 of (McCallum, 1985), there is a coordinate system ϕ mapping U to V such that $R \cap U = \{x \in U \mid \phi_{d+1}(x) = 0, \ldots, \phi_{i-1}(x) = 0\}$. In other words, the image S of $R \cap U$ is the set of points in V with zeros for the last $i - 1 - d$ coordinates. We view p_1 and p_2 in this new coordinate system as $\overline{p_1} = p_1(\phi^{-1}(x_1, \ldots, x_{i-1}), x_i)$ and $\overline{p_2} = p_2(\phi^{-1}(x_1, \ldots, x_{i-1}), x_i)$, respectively. Note that for any $a \in U$, p_1 evaluated at a gives the same univariate polynomial in x_i as $\overline{p_1}$ evaluated at $\phi(a)$ (as is also true for p_2). In particular, since p_1 and p_2 are degree-invariant in R, $\overline{p_1}$ and $\overline{p_2}$ are degree-invariant in S, moreover, the degrees of p_1 and $\overline{p_1}$ are the same, as are the degrees of p_2 and $\overline{p_2}$. Because ϕ^{-1} only acts on the coefficients of p_1 and p_2 (as polynomials in x_i), and because the degrees remain unchanged, the resultant construction commutes with respect to ϕ^{-1}. In other words, recalling that $p_r = \mathrm{res}_{x_i}(p_1, p_2)$, we have $p_r \circ \phi^{-1} = \mathrm{res}_{x_i}(\overline{p_1}, \overline{p_2})$. Moreover, by the "Remark" on p. 45 of (McCallum, 1985), the order of p_r at point a is the same as the order of $p_r \circ \phi^{-1}$ at $\phi(a)$, for any $a \in U$. This means that p_r is order-invariant in $R \cap U$ if and only if $p_r \circ \phi^{-1}$ is order-invariant in S. We will show that the order-invariance of $\mathrm{res}_{x_i}(\overline{p_1}, \overline{p_2})$ in S contradicts the assumption that $\theta_1(r') = \theta_2(r')$.

For convenience, let $f = \theta_1 \circ \phi^{-1}$ and $g = \theta_2 \circ \phi^{-1}$, noting that f and g are themselves analytic functions and thus proper real-root functions for $\overline{p_1}$ and $\overline{p_2}$, and that they agree in value at $\phi(r')$. The function f defines a root of $\overline{p_1}$ on S, not on all of V. Define $f^*(y_1, \ldots, y_{i-1}) = f(y_1, \ldots, y_d, 0, \ldots, 0)$, which extends the domain of f to all of V, and define g^* analogously. Note that f^* and g^* are *analytic* functions of V. [This is actually the key to this proof. Had we tried in the same way to extend θ_1 and θ_2 from functions of R to functions of U, we would not have been guaranteed to get analytic functions.] By division we get $\overline{p_1} = (x - f^*)q_1 + \overline{p_1}(f^*)$. The remainder, $\overline{p_1}(f^*)$, is zero throughout S because f, which f^* extends, is a root function of $\overline{p_1}$ on S. However, since the value of f^* is independent of $y_{d+1}, \ldots, y_{i-1}$, the remainder $\overline{p_1}(f^*)$ is actually zero throughout V. Thus, $\overline{p_1}$ factors as $\overline{p_1} = (x - f^*)q_1$ over V. By the same logic, we get the factorization $\overline{p_2} = (x - f^*)q_2$. Note that the coefficients of q_1 are all polynomials in f^*, so all coefficients are analytic functions of V.

It is shown by Buchberger et al. (1983) (Theorem 1, Chapter *Computing in Algebraic Extensions*) that $res(A, B) = a_m^n \prod_{i=1}^m B(r_i)$, where m is the degree of A, a_m is the leading coefficient of A, the r_is are the m complex roots of A, and n is the degree of B. It easily follows that

$\mathrm{res}(A, BC) = \mathrm{res}(A, B) \cdot \mathrm{res}(A, C)$, from which we get:

$$\begin{aligned}\mathrm{res}_{x_i}((x_i - f^*)q_1, (x_i - g^*)q_2) = {} & \mathrm{res}_{x_i}(x_i - f^*, x_i - g^*) \cdot \mathrm{res}_{x_i}(q_1, x_i - g^*) \\ & \cdot \mathrm{res}_{x_i}(x_i - f^*, q_2) \cdot \mathrm{res}_{x_i}(q_1, q_2).\end{aligned}$$

For any analytic function, if there are points of order k and also of order $k + j$, the set of points of order at least $k + j$ is a closed subset of the set of all points of order at least k. This means that if the order of a product of analytic functions is constant in S then the orders of each of the individual factors must be constant as well. Thus, since the order of $\mathrm{res}_{x_i}(\overline{p_1}, \overline{p_2})$ is constant in S, the order of $\mathrm{res}_{x_i}(x_i - f^*, x_i - g^*)$, which we note is $f^* - g^*$, is constant as well. There are points in R arbitrarily close to r' at which θ_1 and θ_2 are unequal. Let r'' be such a point, sufficiently close such that $\phi(r'') \in S$. Then at $\phi(r'')$, the functions f^* and g^* are unequal. Thus the order of $\mathrm{res}_{x_i}(x_i - f^*, x_i - g^*)$ is zero at $\phi(r'')$, which means its order is zero everywhere in S. So $f^* \neq g^*$ for every point in S. However, since θ_1 and θ_2 are equal at r', f^* and g^* are equal at $\phi(r')$, which is a contradiction. □

Rule 4.10. Let $i \in \mathbb{N}_{>0}$, $R \subseteq \mathbb{R}^i$, $s \in \mathbb{R}^{i-1}$, $p \in \mathbb{Q}[x_1, \ldots, x_i]$, $\mathrm{level}(p) = i$, $\mathtt{I}$ be a symbolic interval of level i, $\preceq$ be an indexed root ordering of level i, and $\preceq^t$ be the reflexive and transitive closure of $\preceq$.

We choose l, u such that either $\mathtt{I} = (\mathrm{sector}, l, u)$ or $(\mathtt{I} = (\mathrm{section}, b)$ for $b = l = u)$.

Assume that p is irreducible, $\mathrm{irExpr}(p, s) \neq \emptyset$, $\xi.p$ is irreducible for all $\xi \in \mathrm{dom}(\preceq)$, $\preceq$ matches s, and for all $\xi \in \mathrm{irExpr}(p, s)$ it holds either $\xi \preceq^t l$ or $u \preceq^t \xi$.

$$\mathit{sample}(s)(R),\ \mathit{repr}(\mathtt{I}, s)(R),\ \mathit{ir\_ord}(\preceq, s)(R),\ \mathit{an\_del}(p)(R) \qquad \vdash \mathit{sgn\_inv}(p)(R)$$

Lemma A.10. *Rule 4.10 on page 18 is correct.*

Proof. As p is delineable on $R{\downarrow_{[i-1]}}$, the variety of p on $R{\downarrow_{[i-1]}}$ is described by the real root functions $\theta_{\xi,s}$, $\xi \in \mathrm{irExpr}(p, s)$. As $\mathit{repr}(\mathtt{I}, s)$ holds on R, it follows that if $l \neq -\infty$, then $\theta_{l,s}$ exists on $R{\downarrow_{[i-1]}}$, and if $u \neq \infty$, then $\theta_{u,s}$ exist on $R{\downarrow_{[i-1]}}$.

As $s \in R{\downarrow_{[i-1]}}$, $\preceq$ matches s, and either $\xi \preceq^t l$ or $u \preceq^t \xi$ for all $\xi \in \mathrm{irExpr}(p, s)$, it holds $\xi(s) \leq l(s)$ or $u(s) \leq \xi(s)$ for all $\xi \in \mathrm{irExpr}(p, s)$. To prove sign-invariance of p on R, we thus only need to show that either $\theta_{\xi,s} < \theta_{l,s}$ on R (if $\xi(s) < l(s)$), $\theta_{u,s} < \theta_{\xi,s}$ on R (if $u(s) < \xi(s)$), or $\theta_{\xi,s} = \theta_{b,s}$ on R (if $\xi(s) = b(s)$) for all $\xi \in \mathrm{irExpr}(p, s)$.

This statement holds as $\preceq$ matches s, $\xi \preceq^t l$ or $u \preceq^t \xi$ for all $\xi \in \mathrm{irExpr}(p, s)$, and $\mathit{ir\_ord}(\preceq, s)$ holds on $R{\downarrow_{[i-1]}}$. □

Rule 4.11. Let $i \in \mathbb{N}_{>0}$, $R \subseteq \mathbb{R}^i$, $s \in \mathbb{R}^{i-1}$, $\mathtt{I}$ be a symbolic interval of level i, $\preceq$ be an indexed root ordering of level i, and $\preceq^t$ be the reflexive and transitive closure of $\preceq$. Assume that $\preceq$ matches s.

Let $Q := \mathit{connected}(i-1)(R) \wedge \mathit{repr}(\mathtt{I}, s)(R)$.

$$\begin{array}{ll}
 & \vdash \mathit{connected}(0)(\mathbb{R}^0) \\
Q\,,\ \mathtt{I} = (\mathit{sector}, l, u),\ l \neq -\infty,\ u \neq \infty,\ l \preceq^t u,\ \mathit{ir\_ord}(\preceq, s) & \vdash \mathit{connected}(i)(R) \\
Q\,,\ \mathtt{I} = (\mathit{sector}, l, u),\ l = -\infty \vee u = \infty & \vdash \mathit{connected}(i)(R) \\
Q\,,\ \mathtt{I} = (\mathit{section}, b) & \vdash \mathit{connected}(i)(R)
\end{array}$$

Lemma A.11. *Rule 4.11 on page 19 is correct.*

Proof. We only consider the case $\mathtt{I} = (\mathit{sector}, l, u), l \neq -\infty, u \neq \infty, l.p \neq u.p$, as the other cases are similar but simpler.

As $\mathit{repr}(\mathtt{I}, s)$ holds, $\theta_{l,s}$ and $\theta_{u,s}$ are well-defined on $R\downarrow_{[i-1]}$. As $s \in R$, $\preceq$ matches s, $l \preceq^t u$, and $\mathit{ir\_ord}(\preceq, s)$ holds on $R\downarrow_{[i-1]}$, it either holds $\theta_{l,s} < \theta_{u,s}$ on $R\downarrow_{[i-1]}$ or $\theta_{l,s} = \theta_{u,s}$ on $R\downarrow_{[i-1]}$. The property $\mathit{repr}(\mathtt{I}, s)$ holds on R and $R\downarrow_{[i-1]}$ is connected, thus R is connected (noting that empty sets are connected). □

Rule 4.12. Let $i \in \mathbb{N}_{>0}$, $R \subseteq \mathbb{R}^i$, $s \in \mathbb{R}^i$, and $\mathtt{I}$ be a symbolic interval of level i. Assume that $s_i \in \mathit{setOf}(s_{[i-1]}, \mathtt{I})$.

$$\begin{array}{ll}
 & \vdash \mathit{sample}(())(\mathbb{R}^0) \\
\mathit{sample}(s_{[i-1]})(R),\ \mathit{repr}(\mathtt{I}, s_{[i-1]})(R) & \vdash \mathit{sample}(s)(R)
\end{array}$$

Lemma A.12. *Rule 4.12 on page 19 is correct.*

Proof. Follows immediately from the definitions. □

Rule 4.13. Let $i \in \mathbb{N}_{>0}$, $R \subseteq \mathbb{R}^{i-1}$, $s \in \mathbb{R}^{i-1}$, and $\mathtt{I}$ be a symbolic interval of level i. Assume that $\mathtt{I}.l \in \mathrm{irExpr}(\mathtt{I}.l.p, s)$ (if $\mathtt{I}.l \neq -\infty$), $\mathtt{I}.u \in \mathrm{irExpr}(\mathtt{I}.u.p, s)$ (if $\mathtt{I}.u \neq \infty$) respectively $\mathtt{I}.b \in \mathrm{irExpr}(\mathtt{I}.b.p, s)$.

$$\begin{array}{ll}
\mathit{sample}(s)(R),\ \mathit{holds}(\mathtt{I})(R),\ \mathtt{I} = (\mathrm{section}, b),\ \mathit{an\_del}(b.p)(R) & \vdash \mathit{repr}(\mathtt{I}, s)(R) \\
\mathit{sample}(s)(R),\ \mathit{holds}(\mathtt{I})(R),\ \mathtt{I} = (\mathrm{sector}, l, u),\ l = -\infty \vee \mathit{an\_del}(l.p)(R),\ u = \infty \vee \mathit{an\_del}(u.p)(R) & \vdash \mathit{repr}(\mathtt{I}, s)(R)
\end{array}$$

Lemma A.13. *Rule 4.13 on page 20 is correct.*

Proof. The graph of any delineable polynomial p over a connected subset $R' \supseteq R$ is described by real root functions $\theta_1 < \ldots < \theta_k$ on R'. It follows that for any indexed root expression ξ with $\xi.p = p$ and $\xi.k \leq k$ it holds $\mathrm{dom}(\theta_{\xi,s}) \supseteq R'$ and $\theta_{\xi,s} = \theta_{\xi.k} = \xi$ on R'.

Now, the result follows immediately from the definition of $\mathit{holds}(\mathtt{I})$ and the previous statement applied on the cell's boundaries. □

References

Ábrahám, Erika, Nalbach, Jasper, Kremer, Gereon, 2017. Embedding the virtual substitution method in the model constructing satisfiability calculus framework. In: Proceedings of the 2nd International Workshop on Satisfiability Checking and Symbolic Computation (SC^2 2017). In: CEUR Workshop Proceedings, vol. 1974. http://ceur-ws.org/Vol-1974/.

Ábrahám, Erika, Davenport, James, England, Matthew, Kremer, Gereon, 2021. Deciding the consistency of non-linear real arithmetic constraints with a conflict driven search using cylindrical algebraic coverings. J. Log. Algebraic Methods Program. 119, 100633. https://doi.org/10.1016/j.jlamp.2020.100633.

Barbosa, Haniel, Reynolds, Andrew, Kremer, Gereon, Lachnitt, Hanna, Niemetz, Aina, Nötzli, Andres, Ozdemir, Alex, Preiner, Mathias, Viswanathan, Arjun, Viteri, Scott, Zohar, Yoni, Tinelli, Cesare, Barrett, Clark, 2022. Flexible proof production in an industrial-strength SMT solver. In: Proceedings of the 11th International Joint Conference on Automated Reasoning (IJCAR 2022). Springer, pp. 15–35. https://doi.org/10.1007/978-3-031-10769-6_3.

Barrett, Clark, Stump, Aaron, Tinelli, Cesare, et al., 2010. The SMT-LIB standard: version 2.0. In: Proceedings of the 8th International Workshop on Satisfiability Modulo Theories. https://smtlib.cs.uiowa.edu/.

Bradford, Russell, Davenport, James H., England, Matthew, McCallum, Scott, Wilson, David, 2016. Truth table invariant cylindrical algebraic decomposition. J. Symb. Comput. 76, 1–35. https://doi.org/10.1016/j.jsc.2015.11.002.

Brown, Christopher W., 2001. Improved projection for cylindrical algebraic decomposition. J. Symb. Comput. 32 (5), 447–465. https://doi.org/10.1006/jsco.2001.0463.

Brown, Christopher W., 2013. Constructing a single open cell in a cylindrical algebraic decomposition. In: Proceedings of the 2013 International Symposium on Symbolic and Algebraic Computation (ISSAC 2013). ACM, pp. 133–140. https://doi.org/10.1145/2465506.2465952.

Brown, Christopher W., 2015. Open non-uniform cylindrical algebraic decompositions. In: Proceedings of the 2015 International Symposium on Symbolic and Algebraic Computation (ISSAC 2015). ACM, pp. 85–92. https://doi.org/10.1145/2755996.2756654.

Brown, Christopher W., 2017. Projection and quantifier elimination using non-uniform cylindrical algebraic decomposition. In: Proceedings of the 2017 International Symposium on Symbolic and Algebraic Computation (ISSAC 2017). ACM, pp. 53–60. https://doi.org/10.1145/3087604.3087651.

Brown, Christopher W., Daves, Glenn C., 2020. Applying machine learning to heuristics for real polynomial constraint solving. In: Proceedings of the 7th International Conference on Mathematical Software (ICMS 2020). In: LNCS, vol. 12097. Springer, pp. 292–301. https://doi.org/10.1007/978-3-030-52200-1_29.

Brown, Christopher W., Košta, Marek, 2015. Constructing a single cell in cylindrical algebraic decomposition. J. Symb. Comput. 70, 14–48. https://doi.org/10.1016/j.jsc.2014.09.024.

Brown, Christopher W., McCallum, Scott, 2020. Enhancements to Lazard's method for cylindrical algebraic decomposition. In: Proceedings of the 22nd International Workshop on Computer Algebra in Scientific Computing (CASC 2020). In: LNCS, vol. 12291. Springer, pp. 129–149. https://doi.org/10.1007/978-3-030-60026-6_8.

Buchberger, B., Collins, George E., Loos, Rudiger, Albrecht, Rudolf (Eds.), 1983. Computer Algebra: Symbolic and Algebraic Computation, 2nd ed. Springer. https://doi.org/10.1007/978-3-7091-7551-4.

Caviness, Bob F., Johnson, Jeremy R., 1998. Quantifier Elimination and Cylindrical Algebraic Decomposition. Texts & Monographs in Symbolic Computation. Springer. https://doi.org/10.1007/978-3-7091-9459-1.

Cimatti, Alessandro, Griggio, Alberto, Irfan, Ahmed, Roveri, Marco, Sebastiani, Roberto, 2017. Invariant checking of NRA transition systems via incremental reduction to LRA with EUF. In: Proceedings of the 23rd International Conference on Tools and Algorithms for the Construction and Analysis of Systems (TACAS 2017). Springer, pp. 58–75. https://doi.org/10.1007/978-3-662-54577-5_4.

Collins, George E., 1975. Quantifier elimination for real closed fields by cylindrical algebraic decomposition. In: Proceedings of the 2nd GI Conference on Automata Theory and Formal Languages (1975). Springer, pp. 134–183. https://doi.org/10.1007/3-540-07407-4_17 (reprinted in the collection Caviness and Johnson (1998)).

Collins, George E., Hong, Hoon, 1991. Partial cylindrical algebraic decomposition for quantifier elimination. J. Symb. Comput. 12, 299–328. https://doi.org/10.1016/S0747-7171(08)80152-6.

Corzilius, Florian, Kremer, Gereon, Junges, Sebastian, Schupp, Stefan, Ábrahám, Erika, 2015. SMT-RAT: an open source C++ toolbox for strategic and parallel SMT solving. In: Proceedings of the 18th International Conference on Theory and Applications of Satisfiability Testing (SAT 2015). Springer, pp. 360–368. https://doi.org/10.1007/978-3-319-24318-4_26.

Cox, D.A., Little, J.B., O'Shea, D.B., 2006. Ideals, Varieties and Algorithms. Springer. https://doi.org/10.1007/978-3-662-41154-4.

Davenport, James H., Heintz, Joos, 1988. Real quantifier elimination is doubly exponential. J. Symb. Comput. 5 (1–2), 29–35. https://doi.org/10.1016/S0747-7171(88)80004-X.

de Moura, Leonardo, Jovanović, Dejan, 2013. A model-constructing satisfiability calculus. In: Proceedings of the 14th International Conference on Verification, Model Checking, and Abstract Interpretation (VMCAI 2013). In: LNCS, vol. 7737. Springer, pp. 1–12. https://doi.org/10.1007/978-3-642-35873-9_1.

Ducos, Lionel, 2000. Optimizations of the subresultant algorithm. J. Pure Appl. Algebra 145 (2), 149–163. https://doi.org/10.1016/S0022-4049(98)00081-4.

England, Matthew, Bradford, Russell, Davenport, James H., 2020. Cylindrical algebraic decomposition with equational constraints. J. Symb. Comput. 100, 38–71. https://doi.org/10.1016/j.jsc.2019.07.019.

Gelfand, Israel M., Kapranov, Mikhail M., Zelevinsky, Andrei V., 1994. Discriminants, Resultants, and Multidimensional Determinants. Springer. https://doi.org/10.1007/978-0-8176-4771-1.

Jovanović, Dejan, 2017. Solving nonlinear integer arithmetic with MCSAT. In: Proceedings of the 18th International Conference on Verification, Model Checking, and Abstract Interpretation (VMCAI 2017). In: LNCS, vol. 10145. Springer, pp. 330–346. https://doi.org/10.1007/978-3-319-52234-0_18.

Jovanović, Dejan, de Moura, Leonardo, 2012. Solving non-linear arithmetic. In: Proceedings of the 6th International Joint Conference on Automated Reasoning (IJCAR 2012). In: LNCS, vol. 7364. Springer, pp. 339–354. https://doi.org/10.1007/978-3-642-31365-3_27.

Jovanovic, Dejan, Barrett, Clark, De Moura, Leonardo, 2013. The design and implementation of the model constructing satisfiability calculus. In: Proceedings of the 13th International Conference on Formal Methods in Computer-Aided Design (FMCAD 2013). IEEE, pp. 173–180. https://doi.org/10.1109/FMCAD.2013.7027033.

Kremer, Gereon, 2019. Cylindrical Algebraic Decomposition for Nonlinear Arithmetic Problems. PhD thesis. RWTH Aachen University. https://publications.rwth-aachen.de/record/792185.

Lazard, Daniel, 1994. An improved projection for cylindrical algebraic decomposition. In: Algebraic Geometry and Its Applications: Collections of Papers from Abhyankar's 60th Birthday Conference. Springer, pp. 467–476. https://doi.org/10.1007/978-1-4612-2628-4_29.

Li, Haokun, Xia, Bican, 2020. Solving satisfiability of polynomial formulas by sample-cell projection. arXiv:2003.00409. https://arxiv.org/abs/2003.00409.

McCallum, Scott, 1985. An improved projection operation for cylindrical algebraic decomposition. PhD thesis. Published as UW-Madison CS Department Technical Report Number TR578. University of Wisconsin-Madison. https://minds.wisconsin.edu/handle/1793/58594.

McCallum, Scott, 1998. An improved projection operation for cylindrical algebraic decomposition. In: Quantifier Elimination and Cylindrical Algebraic Decomposition. Springer, pp. 242–268. https://doi.org/10.1007/978-3-7091-9459-1_12.

McCallum, Scott, 1999. On projection in CAD-based quantifier elimination with equational constraint. In: Proceedings of the 1999 International Symposium on Symbolic and Algebraic Computation (ISSAC 1999). Association for Computing Machinery, pp. 145–149. https://doi.org/10.1145/309831.309892.

McCallum, Scott, Hong, Hoon, 2016. On using Lazard's projection in CAD construction. J. Symb. Comput. 72, 65–81. https://doi.org/10.1016/j.jsc.2015.02.001.

McCallum, Scott, Parusińiski, Adam, Paunescu, Laurentiu, 2019. Validity proof of Lazard's method for CAD construction. J. Symb. Comput. 92, 52–69. https://doi.org/10.1016/j.jsc.2017.12.002.

Nair, Akshar, Davenport, James H., Sankaran, Gregory, 2019. On benefits of equality constraints in lex-least invariant CAD. In: Proceedings of the 4th Workshop on Satisfiability Checking and Symbolic Computation (SC^2 2019). In: CEUR Workshop Proceedings, vol. 2460. http://ceur-ws.org/Vol-2460/.

Nair, Akshar, Davenport, James H., Sankaran, Gregory, 2020. Curtains in CAD: why are they a problem and how do we fix them? In: Proceedings of the 7th International Conference on Mathematical Software (ICMS 2020). In: LNCS, vol. 12097. Springer, pp. 17–26. https://doi.org/10.1007/978-3-030-52200-1_2.

Nalbach, Jasper, Kremer, Gereon, Ábrahám, Erika, 2019. On variable orderings in MCSAT for non-linear real arithmetic. In: Proceedings of the 4th Workshop on Satisfiability Checking and Symbolic Computation (SC^2 2019). In: CEUR Workshop Proceedings, vol. 2460. http://ceur-ws.org/Vol-2460/.

SMT-RAT, a toolbox for strategic and parallel satisfiability modulo theories solving. https://github.com/smtrat/smtrat.

Tarski, Alfred, 1948. A Decision Method for Elementary Algebra and Geometry. RAND Corporation, Santa Monica, CA (reprinted in the collection Caviness and Johnson (1998)). https://doi.org/10.1007/978-3-7091-9459-1_3.