ABSTRACT PARTIAL CAD I: THE LIFTING PHASE (EXTENDED TECHNICAL REPORT)

GRANT OLNEY PASSMORE AND PAUL B. JACKSON

LFCS, Edinburgh and Clare Hall, Cambridge, 10 Crichton Street, Edinburgh EH8 9AB, UK
e-mail address: grant.passmore@cl.cam.ac.uk

LFCS, Edinburgh, 10 Crichton Street, Edinburgh EH8 9AB, UK
e-mail address: pbj@inf.ed.ac.uk

Abstract. Though decidable, the theory of real closed fields (**RCF**) is fundamentally infeasible. This is unfortunate, as automatic proof methods for nonlinear real arithmetic are crucially needed in both formalised mathematics and the verification of real-world cyber-physical systems. Consequently, many researchers have proposed fast, sound but incomplete **RCF** proof procedures which are useful in various practical applications. We show how such practically useful, sound but incomplete **RCF** proof methods may be systematically utilised in the context of a complete **RCF** proof method without sacrificing its completeness. In particular, we present an extension of the **RCF** quantifier elimination method Partial CAD (P-CAD) which uses incomplete $\exists$ **RCF** proof procedures to "short-circuit" expensive computations during the lifting phase of P-CAD. We present the theoretical framework as well as preliminary experiments with an implementation we have undertaken in the open-source computer algebra system SAGE. These experiments include the use of RealPaver, a high-performance interval constraint solver, to short-circuit expensive computations during P-CAD construction.

1. Introduction

Tarski's theorem that the elementary theory of real closed fields (**RCF**) admits effective elimination of quantifiers is one of the longstanding hallmarks of mathematical logic [18]. From this result, the decidability of elementary algebra and geometry readily follow, and a most tantalising situation arises: In principle, every elementary arithmetical conjecture over finite-dimensional real and complex spaces may be decided simply by formalising the conjecture and asking a computer of its truth. All one needs is the fortitude to implement a decision method, the dedication to express conjectures formally, access to high-powered

1998 ACM Subject Classification: I.1.2, I.2.3.

Key words and phrases: decision procedures, nonlinear arithmetic, real closed fields.

This paper includes and extends work presented in Chapters 7 and 8 of the first author's 2011 University of Edinburgh Ph.D. thesis [13], supervised by the second author. This research was supported by the EPSRC [grant numbers EP/I011005/1 and EP/I010335/1]. We thank Jeremy Avigad, James Davenport, Leonardo de Moura and Larry Paulson for valuable conversations. We thank the referees of an earlier version of this article for their helpful comments and suggestions.

© ABSTRACT PARTIAL CAD I: THE LIFTING PHASE(EXTENDED TECHNICAL REPORT)
Creative Commons

computing machinery, and the game is won. So why then do we still not know how many unit hyperspheres may kiss[1] in five dimensions? Is it 41? 42?

The issue is one of complexity. Though decidable, **RCF** is fundamentally infeasible. This observation is made precise by a landmark algorithmic complexity result of the 1980s [7]:

Theorem 1.1 (Davenport-Heintz)**.** *There are families of n-dimensional **RCF** formulas of length $O(n)$ whose only quantifier-free equivalences must contain polynomials of degree $2^{2^{\Omega(n)}}$ and of length $2^{2^{\Omega(n)}}$.*

Thus, arithmetical problems (especially those high-dimensional, i.e., many-variable) will not in general be realistically solvable by full **RCF** quantifier elimination methods. Yet, there are many examples of difficult high-dimensional **RCF** problems solved in mathematical and engineering practice. What is the disconnect?

(1) **RCF** problems solved in practice — especially those solved by hand — are most often solved using an ad hoc combination of methods, *not* by a general decision method.
(2) **RCF** problems arising in practice commonly have structural properties dictated by the application domain from which they originated. Such structural properties can often be exploited making such problems more amenable to analysis and pushing them within the reaches of restricted, more efficient variants of known decision methods.

With this in mind, many researchers have proposed fast, sound but *incomplete* **RCF** proof procedures, many of them being of substantial practical use [2, 10, 19, 14, 16, 8, 6]. This is especially true for formal methods, where improved automated **RCF** proof methods are needed in the formal verification of cyber-physical systems. In these cases, as the **RCF** problems to be analysed are usually machine-generated (and incomprehensibly large), incomplete proof procedures can go a long way. For example, there is no denying the fact that applying a full quantifier elimination algorithm to decide the falsity of a formula such as

$$\exists x_1, \ldots, x_{100} \in \mathbb{R}\ (x_1 * x_1 + \ldots + x_{100} * x_{100} < 0)$$

is an obvious misappropriation of resources. While such an example may seem contrived, consider the fact that when an **RCF** proof method is used in formal verification efforts, it is often fed huge collections of machine-generated formulas which may be (un)satisfiable for extremely simple reasons. Ideally, one would like to be able to use fast, sound but incomplete proof procedures as much as possible, falling back on the far more computationally expensive complete methods only when necessary. It would be desirable to have a principled manner in which incomplete proof methods could be used to improve the performance of a complete method without sacrificing its completeness.

In this paper, we present *Abstract Partial Cylindrical Algebraic Decomposition* (AP-CAD), an extension of the **RCF** quantifier elimination procedure partial CAD. In AP-CAD, arbitrary sound but possibly incomplete $\exists$ **RCF** proof procedures may be used to "short-circuit" certain expensive computations during CAD construction. This is done in

[1]The n-dimensional kissing problem [15] asks: Given an n-dimensional unit hypersphere U centered at the origin in $\mathbb{R}^n$, how many other identical hyperspheres may be arranged so that they each "kiss" U (touch U at a single point) without further overlaps ? In principle, this problem may be solved for each n through iterated application of **RCF** quantifier elimination. But, in practice, this approach is hopeless for reasons we soon discuss.

such a way that the completeness of the combined proof method is guaranteed. We restrict our AP-CAD presentation to the practically useful case of $\exists$ **RCF**. For full-dimensional cell decompositions, we have implemented AP-CAD within our **RCF** proof tool RAHD[2] [13] and within the open-source computer algebra system SAGE [17]. In Section 4 we present experiments utilising RealPaver [9], a modern high-performance interval constraint solver to short-circuit CAD construction.

2. CAD Preliminaries

For a detailed account of CAD, we refer the reader to [3]. We present only the background on (P-)CAD required to understand AP-CAD for $\exists$ **RCF**. P-CAD is currently the most efficient known general quantifier elimination method for **RCF**[3]. An important fact is that the complexity of the (P-)CAD decision algorithm is doubly exponential in the dimension (number of variables) of its input formula. Generally, the most expensive phase of the (P-)CAD algorithm is the so-called "lifting phase." Let us fix some notation.

A *semialgebraic set* is a subset of $\mathbb{R}^n$ definable by a quantifier-free formula in the language of ordered rings. A *region* of $\mathbb{R}^n$ is a connected component of $\mathbb{R}^n$. An *algebraic decomposition* of $\mathbb{R}^n$ is a decomposition of $\mathbb{R}^n$ into finitely many semialgebraic regions. A *cylindrical algebraic decomposition* is a special type of algebraic decomposition whose regions are in a sense "well-behaved" with respect to projections onto lower dimensions. A *cell* is a region of a CAD.

Before delving into technical details, let us discuss how we can use a CAD to make $\exists$ **RCF** decisions. By "the polynomials of (an $\exists$ **RCF** formula) φ," we shall mean the collection of polynomials obtained by zeroing the RHS of every atom in φ through subtracting the RHS from both sides. We assume each such $\exists$ **RCF** formula is in prenex normal form, so that it is an $\exists$-closed boolean combination of *sign conditions*, i.e., of atoms of the form $(p \odot 0)$ with $p \in \mathbb{Z}[x_1, \dots, x_n]$, $\odot \in \{<, \leq, =, \geq, >\}$. We use $QF(\varphi)$ to mean the quantifier-free matrix of φ.

The key point is that if we have in hand a suitable CAD $C = \{c_1, \dots, c_m\} \subset 2^{\mathbb{R}^n}$ derived from an $\exists$ **RCF** formula φ, we can decide the truth of φ from the CAD directly. The reason is simple: C will have the property that every polynomial of φ has *constant sign* on each c_i, i.e., given p a polynomial of φ and a c_i a cell, it shall hold that

$$\forall \vec{r} \in c_i(p(\vec{r}) = 0) \ \vee \ \forall \vec{r} \in c_i(p(\vec{r}) > 0) \ \vee \ \forall \vec{r} \in c_i(p(\vec{r}) < 0).$$

Consequently, $QF(\varphi)$ has constant truth value at every point in a given cell. Thus, to decide φ, we simply substitute a single sample point from each c_i into $QF(\varphi)$ and see if it ever evaluates to **true**. It will evaluate to **true** on at least one sample point if and only if φ is **true** over $\mathbb{R}^n$.

[2]RAHD contains many **RCF** proof methods and allows users to combine them into their own heuristic **RCF** proof procedures through a *proof strategy language*. This is ideal for AP-CAD, as the proof procedure parameters used by AP-CAD can be formally realised as RAHD proof strategies.

[3]See [11] for an explanation as to why P-CAD is also currently the best known general decision method for practical $\exists$ **RCF** problems, despite the fact that $\exists$ **RCF** has a theoretical exponential speed-up over **RCF**.

We shall define CAD by induction on dimension[4]. A CAD of $\mathbb{R}$ is a decomposition of $\mathbb{R}$ into finitely many cells $c_i \subseteq \mathbb{R}$ s.t. each c_i is of the form (i) $\{\alpha_1\}$, or (ii) $]\alpha_1, \alpha_2[$, or (iii) $]\text{-}\infty, \alpha_1[$ or $]\alpha_1, +\infty[$ for algebraic real numbers α_i.
Given a set of univariate polynomials $P = \{p_1, \ldots, p_k\} \in \mathbb{Z}[x]$, the CAD induced by P is simply the collection of roots of the polynomials p_i and the open intervals they induce.

Example 1. Let $P = \{x-1, x^2-2\}$. Then, the CAD induced by P is as follows:

$$c_1 =]\text{-}\infty, -\sqrt{2}[,\ c_2 = \{-\sqrt{2}\},\ c_3 =]-\sqrt{2}, 1[,\ c_4 = \{1\},$$
$$c_5 =]1, \sqrt{2}[,\ c_6 = \{\sqrt{2}\},\ c_7 =]\sqrt{2}, +\infty[.$$

An important observation about a CAD of $\mathbb{R}^1$ is that there is a *natural ordering* between the cells. That is, in the above example it makes sense to say

$$c_1 < c_2 < c_3 < c_4 < c_5 < c_6 < c_7.$$

This ordering property of cells is fundamental to the definition of CADs in higher dimensions and will allow us to obtain a CAD for $\mathbb{R}^{i+1}$ from a CAD for $\mathbb{R}^i$. Observe that for CADs of $\mathbb{R}$, there are essentially two types of cells — singleton pointsets and open intervals. This dichotomy will continue in higher dimensions with the distinction between *sections* and *sectors.*

In what follows, let $\mathcal{A}$ be a region of $\mathbb{R}^i$. We shall call $\mathcal{A} \times \mathbb{R}$ the *cylinder over* $\mathcal{A}$ and denote it by $Z(\mathcal{A})$.

Definition 2.1 (Stack)**.** Let $f_1, \ldots, f_k \in \mathcal{C}(\mathcal{A}, \mathbb{R})$. That is, f_j is a continuous function from $\mathcal{A}$ to $\mathbb{R}$. Furthermore, suppose that the images of the f_j are ordered over $\mathcal{A}$ s.t. $\forall \alpha \in \mathcal{A}\,(f_j(\alpha) < f_{j+1}(\alpha))$. Then, $f_1, \ldots, f_k$ induce a *stack* $\mathfrak{S}$ over $\mathcal{A}$, where $\mathfrak{S}$ is a decomposition of $Z(\mathcal{A})$ into $2k+1$ regions of the following form:

- $r_1 = \{\langle \alpha, x\rangle \mid \alpha \in \mathcal{A}, x < f_1(\alpha)\}$,
  $r_3 = \{\langle \alpha, x\rangle \mid \alpha \in \mathcal{A}, f_1(\alpha) < x < f_2(\alpha)\}$,
  $\vdots$
  $r_{2k-1} = \{\langle \alpha, x\rangle \mid \alpha \in \mathcal{A}, f_{k-1}(\alpha) < x < f_k(\alpha)\}$,
  $r_{2k+1} = \{\langle \alpha, x\rangle \mid \alpha \in \mathcal{A}, f_k(\alpha) < x\}$,
- $r_2 = \{\langle \alpha, x\rangle \mid \alpha \in \mathcal{A}, x = f_1(\alpha)\}$,
  $\vdots$
  $r_{2k} = \{\langle \alpha, x\rangle \mid \alpha \in \mathcal{A}, x = f_k(\alpha)\}$.

We call regions of the odd index form *sectors* and regions of the even index form *sections.*

We may now give the inductive step of the CAD definition. A CAD of $\mathbb{R}^{i+1}$ will be obtained from a CAD C of $\mathbb{R}^i$ by constructing a stack over every cell in C.

Definition 2.2 (CAD in $\mathbb{R}^{i+1}$)**.** An algebraic decomposition C_{i+1} of $\mathbb{R}^{i+1}$ is a CAD iff C_{i+1} is a *union of stacks* $C_{i+1} = \bigcup_{j=1}^{k} w_j$, s.t. the stack w_j is constructed over cell c_j in a CAD $C_i = \{c_1, \ldots, c_k\}$ of $\mathbb{R}^i$.

The P-invariance property will allow us to use CADs to make $\exists$ **RCF** decisions.

[4]We shall speak freely of the symbolic manipulation and arithmetic of (irrational) real algebraic numbers. See, e.g., [3] for an algorithmic account.

Definition 2.3 (P-invariance)**.** Let $P = \{p_1, \ldots, p_k\} \subset \mathbb{Z}[x_1, \ldots, x_n]$ and $\mathcal{A}$ be a region of $\mathbb{R}^n$. Then, we say $\mathcal{A}$ is *P-invariant* iff every member of P has constant sign on $\mathcal{A}$. That is given any $p_i \in P$,

$$\forall \vec{r} \in \mathcal{A}(p_i(\vec{r}) = 0) \quad \vee \quad \forall \vec{r} \in \mathcal{A}(p_i(\vec{r}) > 0) \quad \vee \quad \forall \vec{r} \in \mathcal{A}(p_i(\vec{r}) < 0).$$

Given a CAD C, we say C is P-invariant iff every cell of C is P-invariant.

2.1. CAD Construction and Evaluation for $\exists$ RCF.

The use of CADs for deciding $\exists$ **RCF** sentences will take place in four steps. In what follows, φ is an $\exists$ **RCF** sentence and $P = \{p_1, \ldots, p_k\} \subset \mathbb{Z}[x_1, \ldots, x_n]$ is the collection of polynomials of φ.

: Projection The *projection phase* will begin with P and iteratively apply a *projection operator $Proj_i$* of the form $Proj_i : 2^{\mathbb{Z}[x_1,\ldots,x_{i+1}]} \to 2^{\mathbb{Z}[x_1,\ldots,x_i]}$ until a set of polynomials is obtained over $\mathbb{Z}[x_1]$. This process will consist of levels, one for each dimension, s.t. at each level i we will have what is called a *level-i projection set*, $P_i \subset \mathbb{Z}[x_1, \ldots, x_i]$. These level-$i$ projection sets will have a special property: If we have a P_i-invariant CAD of $\mathbb{R}^i$, then we can use this CAD to construct a P_{i+1}-invariant CAD of $\mathbb{R}^{i+1}$.

: Base The *base phase* consists of computing a P_1-invariant CAD of $\mathbb{R}^1$, implicitly described as a sequence of sample points, one for each cell in the CAD. This can be done by *univariate real root isolation* and basic machinery for arithmetic with real algebraic numbers. Let us suppose we have done this and our sequence of sample points is $\vec{s_1} < \vec{s_2} < \ldots < \vec{s_{2m+1}}$.

: Lifting The *lifting phase* will take an implicit description of a P_1-invariant CAD of $\mathbb{R}^1$ and progressively transform it into an implicit description of P_n-invariant CAD of $\mathbb{R}^n$. Let $C = \{c_1, \ldots, c_m\}$ be the P_i-invariant CAD for $\mathbb{R}^i$ which we will lift to a P_{i+1}-invariant CAD of $\mathbb{R}^{i+1}$. Let $S = \{\vec{s_1}, \ldots, \vec{s_m}\}$ be our set of sample points, one from each cell in C. Then, for each cell c_j, we will use the sample point $\vec{s_j} \in c_j$ to construct a set of sample points in $\mathbb{R}^{i+1}$ corresponding to a *stack over c_j*:

1. As $\vec{s_j} \in \mathbb{R}^i$, we have that $\vec{s_j} = \langle r_1, \ldots, r_i \rangle$ for some $r_1, \ldots, r_i \in \mathbb{R}$.
2. Let $P_{i+1}[\vec{s_j}]$ denote $P_{i+1}[x_1 \mapsto r_1, x_2 \mapsto r_2, \ldots, x_i \mapsto r_i]$. Then $P_{i+1}[\vec{s_j}] \subset \mathbb{Z}[x_{i+1}]$ is a univariate family of polynomials.
3. Using the same process as we did in the base phase, compute a $P_{i+1}[\vec{s_j}]$-invariant CAD of $\mathbb{R}^1$. Let this CAD be represented by a sequence of sample points $\vec{t_1} < \vec{t_2} < \ldots < \vec{t_{2v+1}} \in \mathbb{R}$.
4. Then, the *stack over c_j* will be represented by the set of $2v + 1$ sample points obtained by appending each $\vec{t_j}$ to the lower-dimensional sample point $\vec{s_j}$. That is, our stack over c_j will be represented by the following sequence of sample points $\vec{z_1}, \ldots, \vec{z_{2v+1}}$ in $\mathbb{R}^{i+1}$: $\vec{z_1} = \langle r_1, \ldots, r_i, t_1 \rangle$, $\vec{z_2} = \langle r_1, \ldots, r_i, t_2 \rangle$, $\ldots$, $\vec{z_{2v+1}} = \langle r_1, \ldots, r_i, t_{2v+1} \rangle$.

In the above construction, we call the cell c_j (or the sample point representing it, $\vec{s_j}$) the *parent* of the stack $\{\vec{z_1}, \ldots, \vec{z_{2v+1}}\}$.

: Evaluation Let $S = \{\vec{s_1}, \ldots, \vec{s_m}\} \subset \mathbb{R}^n$ be our final set of sample points. Return the boolean value $\bigvee_{\vec{r} \in S} QF(\varphi)[\vec{r}]$.

2.2. **Partial CAD.** Let us now sketch the idea of partial CAD, due to Collins and Hong [5]. As it stands, the CAD construction algorithm will build a P-invariant CAD induced by the polynomials P of an $\exists$ **RCF** formula φ without paying any attention to the logical content of the formula itself. But, when performing lifting, i.e., constructing a stack of regions of $\mathbb{R}^{i+1}$ over a lower-dimensional cell $c_j \subset \mathbb{R}^i$, we may be easily able to see — simply by substitution and evaluation — that the formula $QF(\varphi)$ could *never* be satisfied over c_j. For instance, let $QF(\varphi) = \left((x_4^4 + x_3x_2^3 + 3x_1 > 2x_1^4) \ \wedge \ (x_1^2 > x_2 + x_3)\right)$. If c_j is a cell in a P_3-invariant CAD of $\mathbb{R}^3$ represented by the sample point $\vec{s_j} = \langle 0, 1, 5\rangle$, then we can see $QF(\varphi)$ will *never* be satisfied over a cell in a stack which is a child of c_j. Thus, we need not lift over c_j and can eliminate it.

This is the idea behind *partial CAD* when applied to $\exists$ **RCF** formulas: Before lifting over a cell in a CAD of $\mathbb{R}^i$, check if there are any atoms in your formula only involving the variables $x_1, \ldots, x_i$. If so, then perform *partial* evaluation of your formula by evaluating those atoms upon your sample point in $\mathbb{R}^i$, and then use simple propositional reasoning to try to deduce the truth of your formula. This can also allow us to find a *satisfying* assignment for the variables in $QF(\varphi)$ without constructing a whole CAD. For instance, let $QF(\varphi) = \left((x_4^4 + x_3x_2^3 + 3x_1 > 2x_1^4) \ \vee \ (x_1^2 < x_2)\right)$. If c_j is a cell in a P_2-invariant CAD of $\mathbb{R}^2$ represented by the sample point $\vec{s_j} = \langle -1, 2\rangle$, then we can see immediately by substitution that $QF(\varphi)$ is satisfiable over $\mathbb{R}^4$. As a witness to this satisfiability, we may return $\langle -1, 2, r_3, r_4\rangle$ where $r_3, r_4 \in \mathbb{R}$ are arbitrary.

3. Abstract Partial CAD

From a high level of abstraction, we can see partial CAD to be normal CAD augmented with three pieces of algorithmic data:

(1) A strategy for selecting lower-dimensional cells to use for evaluating lower-dimensional atoms in our input formula,
(2) An algorithm which when given a cell c_j will construct a formula $F(c_j)$ which, if it both has a truth value and is decided, can be used to tell (i) if the cell c_j can be thrown away (i.e., $F(c_j)$ is decided to be **false**), or (ii) if a satisfying assignment for our formula can be extracted already from a lower-dimensional cell (i.e., $F(c_j)$ is decided to be **true**),
(3) A proof procedure which will be used to decide the formulas $F(c_j)$ generated by the algorithm above.

In fact, in their original paper on partial CAD, Collins and Hong make the point that different cell selection strategies could be used and even implement and experiment with a number of them[5]. For partial CAD restricted to $\exists$ **RCF**, these three pieces of algorithmic data described above would be:

(1) Select cells $c_i \in C$ in some specified enumeration order (specified by s): $c_{s(1)}, c_{s(2)}, c_{s(3)}, \ldots$.
(2) Given a cell c_j represented by a sample point $\vec{s_j} = \langle r_1, \ldots, r_i\rangle \in \mathbb{R}^i$, the formula $F(c_j)$ will be constructed from our original $\exists$ **RCF** formula φ by the following process:

[5]For Collins and Hong, a cell selection strategy selects *single* cells in some specified order. In Abstract Partial CAD, cell selection strategies will select *sets* of cells in some specified order and $\exists$ **RCF** proof procedures will be applied to see if every cell in a selected set of cells may be eliminated.

   (a) Let φ' be $QF(\varphi)$ augmented by instantiating x_1 with r_1, x_2 with r_2, ...
   (b) Evaluate all variable-free atoms in φ' to obtain a new formula φ''.
   (c) Replace all (unique) variable-containing atoms in φ'' with fresh propositional variables to obtain a new formula $F(c_j)$.
(3) Use a propositional logic proof procedure to attempt to decide $F(c_j)$.

If $F(c_j)$ is **false** (i.e., unsatisfiable), cell c_j can be abandoned and we need not lift over it. If $F(c_j)$ is **true** (i.e., tautologous), then we can extract a witness to the truth of φ from the sample point $\vec{s_j}$. Otherwise, we lift over c_j. These three pieces of data give us the widely-used partial CAD of Collins and Hong. But, from this point of view, we see that there are *many other choices* we could make.

3.1. Stages, Theatres and Lifting.

The fundamental notion of AP-CAD will be that of an *stage*[6]. Let $\mathcal{L}_{\exists OR}$ be the fragment of the language of ordered rings consisting of purely $\exists$ prenex sentences.

Definition 3.1 (Stage)**.** A *stage* $\mathfrak{A} = \langle\langle \mathbb{S}, w\rangle, \mathbb{F}, \mathbb{P}\rangle$ will be given by three pieces of algorithmic data. We describe a stage by how it acts in the context of a fixed (but arbitrary) i-dimensional space $\mathbb{R}^i$. These data are as follows:

(1) A *cell selection strategy* for selecting *subsets* of C_i for analysis (we call such a subset a "selection of cells"),
(2) A *formula construction strategy* for constructing an $\exists$ **RCF** formula whose truth value will correspond to the relevance of a selection of cells (we call such a formula a "cell selection relevance formula"),
(3) An $\exists$ **RCF** proof procedure used to (attempt to) decide the truth or falsity of a cell selection relevance formula.

In the context of CAD construction, sample points will be eliminated when their corresponding cells are deemed to be irrelevant to the $\exists$ **RCF** formula inducing the CAD. This removal might then result in a set of sample points for which the cell selection function behaves differently than it did initially. This motivates the *containment axiom* for covering width functions, so that these dynamics do not result in a non-terminating CAD-based decision algorithm employing the stage machinery. In what follows, let $R_i = \{s \subset \mathbb{R}^i \mid |s| < \omega\}$.

(1) A *cell selection strategy* consists of two components:
   (a) A *covering width function* $w : R_i \to \mathbb{N}$,
   (b) A *cell selection function* $\mathbb{S} : R_i \times \mathbb{N} \to R_i$ obeying for all $S_i \in R_i$ and all $j \in \{1, \ldots, w(S_i)\}$ the *containment axiom*: $\mathbb{S}(S_i, j) \subset S_i$.
(2) A *formula construction strategy* is a function $\mathbb{F} : \mathcal{L}_{\exists OR} \times R_i \to \mathcal{L}_{\exists OR}$ obeying certain *relevance judgment axioms*. To describe these axioms, we need the context of a fixed (but arbitrary) $\exists$ **RCF** formula and an associated P_i-invariant CAD of $\mathbb{R}^i$. Let φ be an $\exists$ **RCF** formula with polynomials $P \subset \mathbb{Z}[x_1, \ldots, x_n]$ and let $P_n, \ldots, P_1$ be a sequence of level-$(n, \ldots, 1)$ projection sets rooted in P (recall $P_n = P$). Let $C_i = \{c_1, \ldots, c_m\}$ be a P_i-invariant CAD of $\mathbb{R}^i$ with S_i a set of sample points drawn from a subset of the cells in C_i. If we are given a set of sample points $\{\vec{s_{a_1}}, \ldots \vec{s_{a_v}}\} \subseteq S_i$, then $\triangle(\{\vec{s_{a_1}}, \ldots \vec{s_{a_v}}\})$ will denote the set of cells from which the sample points $\vec{s_{a_j}}$ are drawn. Then, for each set of sample points S_i and each $j \in \{1, \ldots, w(S_i)\}$ the following *relevance judgment axioms* must hold:

[6]The intended connotation is of a *stage* in a *theatre*.

$$\mathbf{RCF} \models \neg\mathbb{F}(\varphi, \mathbb{S}(S_i, j)) \quad \Longrightarrow \quad \mathcal{N}(\varphi, \mathbb{S}(S_i, j)),$$

and

$$\mathbf{RCF} \models \mathbb{F}(\varphi, \mathbb{S}(S_i, j)) \quad \Longrightarrow \quad \mathbf{RCF} \models \varphi,$$

where $\mathcal{N}(\varphi, \{\vec{s_{a_1}}, \ldots \vec{s_{a_v}}\})$ means that no child (at any ancestral depth, i.e., in a P_{i+1}-invariant CAD of $\mathbb{R}^{i+1}$, in a P_{i+2}-invariant CAD of $\mathbb{R}^{i+2}, \ldots$, in a P_n-invariant CAD of $\mathbb{R}^n$) of any cell in the set $\Delta(\{\vec{s_{a_1}}, \ldots, \vec{s_{a_v}}\})$ will satisfy $QF(\varphi)$.

(3) An $\exists$ **RCF** *proof procedure* is a function

$$\mathbb{P} : \mathcal{L}_{\exists OR} \to \left(\{\mathbf{true}, \mathbf{false}, \mathbf{unknown}\} \cup \bigcup_{j \in \mathbb{N}^+} \mathbb{R}^j \right)$$

obeying the *soundness axioms*:

$$\mathbb{P}(\psi) = \mathbf{true} \quad \Longrightarrow \quad \mathbf{RCF} \models \psi,$$

$$\mathbb{P}(\psi = \mathbf{false} \quad \Longrightarrow \quad \mathbf{RCF} \models \neg\psi,$$

$$\mathbb{P}(\psi) \in \mathbb{R}^j \quad \Longrightarrow \quad \mathbf{RCF} \models QF(\psi)[\mathbb{P}(\psi)]$$

for arbitrary $\psi \in \mathcal{L}_{\exists OR}$ and with $QF(\psi)[\mathbb{P}(\psi)]$ in the final axiom being the substitution of the j-vector $\mathbb{P}(\psi)$ (or an arbitrary extension of it to the dimension of the polynomials appearing in ψ) into ψ, resulting in a variable-free formula. In this case, we call $\mathbb{P}(\psi)$ a *witness* to the truth of ψ.

We will want to have the freedom to give our AP-CAD algorithm a *sequence* of stages, one for each dimension $1, \ldots, n$. The intuition is as follows: Stages are introduced so that one can present a *strategy* to an underlying CAD decision algorithm which will prescribe a method for the algorithm to recognise when it can short-circuit certain expensive computations. If we can abandon a cell at a low dimension, for instance at the base phase or when beginning to lift over cells of $\mathbb{R}^2$, then this can potentially give us *hyper-exponential* savings down the line. Thus, it makes sense to arrange stages $\mathfrak{A}_1, \mathfrak{A}_2, \ldots \mathfrak{A}_n$ so that stage $\mathfrak{A}_1$ works *hardest* to make relevance judgments about cells. For if $\mathfrak{A}_1$ causes us to throw away cell $c_j \subset \mathbb{R}^1$, then this could lead to huge savings later. Then, $\mathfrak{A}_2$ might still work hard but a bit *less* hard, and so on. This collection of stages gives rise to the notion of an *n-theatre.* In what follows, let Θ be the set of all stages.

Definition 3.2 (Theatre)**.** An *n-theatre* $\mathbb{T}$ is a function $\mathbb{T} : \{1, \ldots, n\} \to \Theta$.

Stage i in a theatre will be used to make judgments about cells in a P_i-invariant (partial) CAD of $\mathbb{R}^i$ (i.e., at level i). Let us describe a decision method we will use for deciding $\exists$ **RCF** sentences in the framework of AP-CAD.

Algorithm 2 (AP-CAD with Theatrical Lifting)**.** Suppose we are given an $\exists$ **RCF** sentence φ with polynomials $P \subset \mathbb{Z}[x_1, \ldots, x_n]$, and an n-theatre $\mathbb{T}$.

(1) **Projection** As with standard P-CAD, compute a sequence of projection sets $P_1, \ldots, P_n$.
(2) **Base** As with standard P-CAD, compute a P_1-invariant CAD of $\mathbb{R}^1$, $C_1 = \{c_1, \ldots, c_{2m+1}\}$ represented by sample points $S_1 = \{\vec{s_1}, \ldots, \vec{s_{2m+1}}\}$. Set the current dimension $i := 1$.

```
(3) **Lifting** Let $\mathbb{T}(i) = \mathfrak{A}_i = \langle\langle \mathbb{S}_i, w_i\rangle, \mathbb{F}_i, \mathbb{P}_i\rangle$ be the stage for dimension $i$, and $S_i$ the
    set of sample points for the $P_i$-invariant (partial) CAD of $\mathbb{R}^i$ over which we need to
    lift.
    (a) Let $U := w_i(S_i)$ and let $j := 1$.
    (b) **While** $j \leq U$ **do**
        (i) Let $\{s_{\vec{a_1}}, \ldots, s_{\vec{a_v}}\} := \mathbb{S}_i(S_i, j)$.
        (ii) Let $\chi := \mathbb{P}_i(\mathbb{F}_i(\{s_{\vec{a_1}}, \ldots, s_{\vec{a_v}}\}))$.
        (iii) If $\chi =$ **true**, then **return true**.
        (iv) If $\chi = \langle x_1, \ldots, x_w\rangle \in \mathbb{R}^w$ for some $w \leq n$, then
            (A) Fix an $n$-dim. extension of $\chi$, e.g., $\vec{r} = \langle x_1, \ldots, x_w, \vec{0}\rangle \in \mathbb{R}^n$.
            (B) Evaluate $QF(\psi)[\vec{r}]$ and set $R \in \{\mathbf{true}, \mathbf{false}\}$ to this result.
            (C) If $R =$ **true**, then **return** $\vec{r}$ as a witness to the truth of $\varphi$.
            (D) If $R =$ **false**, then **return true**[7].
        (v) If $\chi =$ **false**, then set $S_i' := S_i \setminus \{s_{\vec{a_1}}, \ldots, s_{\vec{a_v}}\}$, else set $S_i' := S_i$.
        (vi) If $S_i' = \emptyset$ then **return false**.
        (vii) If $S_i' = S_i$ then set $j := j + 1$.
        (viii) If $S_i' \subset S_i$ then
            (A) Set $S_i := S_i'$.
            (B) Set $U := w_i(S_i)$.
            (C) Set $j := 1$.
    (c) Now, $S_i = \{\vec{t_1}, \ldots, \vec{t_u}\}$ contains sample points corresponding to the cells we
        have not deemed to be irrelevant. We need to lift over them.
        (i) Let $S_{i+1} := \emptyset$.
        (ii) **For** $j$ **from** 1 **to** $u$ **do**
            (A) Substitute the components of $\vec{t_j}$ in for the variables $x_1, \ldots, x_i$ in
                $P_{i+1}$ to obtain a univariate family $P_{i+1}[\vec{t_j}] \subset \mathbb{Z}[x_{i+1}]$.
            (B) Compute a $P_{i+1}[\vec{t_j}]$-invariant CAD of $\mathbb{R}^1$, represented by sample
                points $K_j$.
            (C) Set $S_{i+1} := S_{i+1} \cup K_j$.
    (d) Increase the current dimension by setting $i := i + 1$.
    (e) If $i = n$ then lifting is done and we may proceed to the *evaluation phase.*
    (f) If $i < n$ then we loop and begin the lifting process again, but now with the set
        of sample points $S_{i+1}$.
(4) **Evaluation** Return the boolean value $\bigvee_{\vec{r} \in S_n} QF(\varphi)[\vec{r}]$.
```

Let us prove the correctness of Algorithm 2. This will be straight-forward given the correctness of the classical CAD-based decision algorithm outlined previously, which we accept as given.

Theorem 3.3. *Algorithm 2 is a sound and complete* $\exists$ ***RCF*** *proof procedure.*

Proof. There are two essential differences between this AP-CAD algorithm and the classical one. These both take place during lifting. In Algorithm 2, we may

[7]This is perhaps the one counter-intuitive part of the algorithm. Note, however, that this is actually correct: By the combination of the second *relevance judgment axiom* for $\mathbb{F}_i$ and the *soundness axioms* for $\mathbb{P}_i$, the fact that **RCF** $\models \mathbb{F}_i(\mathbb{S}_i(S_i, j))$ means that φ is **true**. It's just that the witness $\mathbb{P}_i$ computed for the truth of $\mathbb{F}_i(\mathbb{S}_i(S_i, j))$ might fail to be a witness for φ. In this case, we simply know φ is true without knowing a witness for it.

- discard a collection of cells if they are deemed to be irrelevant, and
- quit CAD construction altogether and return either **true** or a witness to the truth of our input formula φ, or return **false** in the case that all cells have been discarded.

If $\{s_{a_1}, \ldots, s_{a_v}\} \subset \mathbb{R}^i$ is a set of sample points for a P_i-invariant partial CAD of $\mathbb{R}^i$, we will say that $\{s_{a_1}, \ldots, s_{a_v}\}$ *respects the truth of* φ to mean that there is some n-dimensional child of a sample point in $\{s_{a_1}, \ldots, s_{a_v}\}$ satisfying $QF(\varphi)$ iff φ is **true**.

We will proceed by induction, assuming that the algorithm has constructed a set of sample points $\{s_{a_1}, \ldots, s_{a_v}\}$ for a P_i-invariant partial CAD of $\mathbb{R}^i$ which respects the truth of φ. The base case is verified by noting that the *base phase* of the algorithm constructs a full set of sample points for a P_1-invariant CAD of $\mathbb{R}^1$ which trivially respects the truth of φ.

Let us first observe that if we discard a collection of cells because they have been deemed to be irrelevant, then we have not affected the soundness of the decision algorithm.

Cells of a P_i-invariant partial CAD of $\mathbb{R}^i$ will only be deemed to be irrelevant when an *stage* $\mathfrak{A}_i$ indicates this is the case. The key line in the algorithm is 3(b)v, where $\chi = \mathbb{P}_i(\mathbb{F}_i(\varphi, \{s_{a_1}, \ldots, s_{a_v}\}))$. For this discarding to have occurred, we must have $\chi =$ **false**. By the *second soundness axiom* for $\mathbb{P}_i$, this means

$$\mathbf{RCF} \models \neg\mathbb{F}_i(\varphi, \{s_{a_1}, \ldots, s_{a_v}\}).$$

By the *first relevance judgment axiom* for $\mathbb{F}_i$, this means that $\mathcal{N}(\varphi, \{s_{a_1}, \ldots, s_{a_v}\})$ must hold. Recall that $\mathcal{N}(\varphi, \{s_{a_1}, \ldots s_{a_v}\})$ means that no child (at any ancestral depth, i.e., in a P_{i+1}-invariant CAD of $\mathbb{R}^{i+1}$, in a P_{i+2}-invariant CAD of $\mathbb{R}^{i+2}, \ldots,$ in a P_n-invariant CAD of $\mathbb{R}^n$) of any cell in the set $\Delta(\{s_{a_1}, \ldots, s_{a_v}\})$ of cells corresponding to the sample points $\{s_{a_1}, \ldots, s_{a_v}\}$ will satisfy $QF(\varphi)$. Thus, by our induction hypothesis, removing the cells from our analysis does not affect the soundness of the decision algorithm. In particular, if we have removed all cells, this means that *no* ancestor of the cells at our current dimension can satisfy $QF(\varphi)$. By our induction hypothesis this means that there exists no n-dimensional real vector satisfying $QF(\varphi)$, and thus φ is **false** as the algorithm will report via line 3(b)vi.

Let us now turn to the second difference: Algorithm 2 may quit CAD construction altogether and return either **true** or a witness satisfying $QF(\varphi)$.

In the latter case, a witness is only returned if the algorithm verified, by evaluation, that the witness satisfies $QF(\varphi)$. That this does not affect soundness is apparent.

Let us examine the remaining case, when the algorithm returns simply **true** during lifting. The first place this occurs is on line 3(b)iii. This happens when $\mathbb{P}_i(\mathbb{F}_i(\varphi, \{s_{a_1}, \ldots, s_{a_v}\}))$ is equal to **true**. By the *first soundness axiom* for $\mathbb{P}_i$, this means

$$\mathbf{RCF} \models \mathbb{F}_i(\varphi, \{s_{a_1}, \ldots, s_{a_v}\}).$$

By the *second relevance judgment axiom* for $\mathbb{F}_i$, it then follows that φ is in fact **true** over **RCF** and so the soundness of the algorithm is not affected.

Finally, let us consider the second scenario in which this could occur, line 3(b)ivD. In this case, $\mathbb{P}_i(\mathbb{F}_i(\varphi, \{s_{a_1}, \ldots, s_{a_v}\})) \in \mathbb{R}^j$ for some $j \in \mathbb{N}$. By the *third soundness axiom* for $\mathbb{P}_i$, this means that

$$\mathbf{RCF} \models QF(\mathbb{F}_i(\varphi, \{s_{a_1}, \ldots, s_{a_v}\}))[\mathbb{P}_i(\mathbb{F}_i(\varphi, \{s_{a_1}, \ldots, s_{a_v}\}))].$$

But this implies that

$$\mathbf{RCF} \models \mathbb{F}_i(\varphi, \{s_{a_1}, \ldots, s_{a_v}\})).$$

So, as in the last case, by the *second relevance judgment axiom* for $\mathbb{F}_i$, this means that φ is in fact **true**.

Finally, a word on termination of the while loop (cf. line 3b): Consider a pass of the loop. If any sample points in S_i are discarded, then $|S_i|$ is reduced. If no sample points in S_i are discarded, then U remains constant and j is incremented by 1. Thus, the lexicographic product measure $\mu = \langle |S_i|, U - j + 1\rangle$ is always decreased along the ordinal ω^2. If ever $|S_i|$ is reduced to 0, then line 3(b)vi guarantees termination. Combining this with the fact that the loop termination condition is $(j > U)$, it follows by the well-foundedness of ω^2 that the loop must terminate.

Thus, by the correctness of the classical CAD-based decision algorithm, it follows by induction that Algorithm 2 is sound and terminating. □

4. Experimental Results

As an experiment, we built a concrete AP-CAD instance combining RealPaver [9], a modern high-performance interval constraint solver, with partial formula evaluation upon sample points as found in classical partial CAD. This is a significant extension of earlier work we have done on an interval-based AP-CAD instance [13], extending this previous work both in terms of the power of the interval method considered and in the scope of the empirical study[8]. As CAD performance is strongly dependent on the number of cells retained at each dimension, we shall compare this for standard P-CAD and this AP-CAD w.r.t. a family of real-world benchmarks derived from the MetiTarski project [1]. Let us first define our AP-CAD theatre and then discuss our experiments with it.

4.1. Definition of our AP-CAD Theatre.

In defining this theatre, it will be useful to allow our functions to work explicitly over *lists* of sample points as opposed to sets of sample points. To do so, we use the maps

$$StoL : R_i \to Lists(R_i)$$

and

$$LtoS : Lists(R_i) \to R_i.$$

StoL will map a set of sample points to a *sorted* representation of the set as a list, and *LtoS* will map a list of sample points to its underlying set. We use the lexicographic product order of the normal ordering $<$ on $\mathbb{R}$ to order the sample points. If l is a list, then $|l|$ will be the length of the list. If l is a list and $0 \leq m \leq n \leq |l|$, then $subseq(l, m, n)$ will be the subsequence of l of the form[9] $\langle l(m), \ldots, l(n-1)\rangle$. We now build a stage for our theatre.

: cell selection function $\mathbb{S}(s, n) = LtoS(\mathbb{S}_{Lists}(StoL(s), n))$ where

$$\mathbb{S}_{Lists}(l, n) = \begin{cases} l & \text{if } n \leq 1, \\ \lfloor S_{Lists}(l, k) \rfloor & \text{if } n = 2k, \\ \lceil S_{Lists}(l, k) \rceil & \text{if } n = 2k+1, \end{cases}$$

[8]The AP-CAD implementation we report upon in the present paper is a new one, implemented within the open-source computer algebra system SAGE [17]. It is available at the following URL: `http://www.cl.cam.ac.uk/~gp351/sage/`.

[9]This perhaps strange way of indexing list subsequences is used so that our description matches our actual implemention, as this is how Common Lisp does list subsequencing via the `subseq` function. For example, `subseq(`$\langle a,b,c\rangle$`, 0, 2) = `$\langle a,b\rangle$ and `subseq(`$\langle a,b,c\rangle$`, 0, 0) = nil`, the empty list.

and

$$\lfloor l \rfloor = \begin{cases} subseq(l,0,k) & \text{if } |l| = 2k, \\ subseq(l,0,k+1) & \text{if } |l| = 2k+1, \end{cases}$$

and

$$\lceil l \rceil = \begin{cases} subseq(l,k,|l|) & \text{if } |l| = 2k, \\ subseq(l,k+1,|l|) & \text{if } |l| = 2k+1. \end{cases}$$

Let us explain these functions in words. The function $\lfloor l \rfloor$ returns the first half of the list l if $|l|$ is even, and returns the first $k+1$ elements of l if $|l| = 2k+1$. The function $\lceil l \rceil$ returns the second half of the list l if $|l|$ is even, and returns the final k elements of l if $|l| = 2k+1$. In this way, we always have that the concatenation of $\lfloor l \rfloor$ and $\lceil l \rceil$ is l itself. These two functions are used to "bisect" the list l by the function $\mathbb{S}_{Lists}$, regardless of whether or not $|l|$ is even or odd.

The function $\mathbb{S}_{Lists}(l,n)$ computes subsequences of the list l in a "divide and conquer" fashion, with the parameter n specifying which subsequence should be computed. It is best understood as representing an enumeration of subsequences of l which have been situated in a binary tree. To illustrate a concrete example, let $l = \langle a_1, \ldots, a_7 \rangle$. Then, we have

$$\begin{aligned}
&\mathbb{S}_{Lists}(l,1) = l = \langle a_1, \ldots, a_7 \rangle, \\
&\mathbb{S}_{Lists}(l,2) = \lfloor \langle a_1, \ldots, a_7 \rangle \rfloor = \langle a_1, \ldots, a_4 \rangle, \\
&\mathbb{S}_{Lists}(l,3) = \lceil \langle a_1, \ldots, a_7 \rangle \rceil = \langle a_5, \ldots, a_7 \rangle, \\
&\mathbb{S}_{Lists}(l,4) = \lfloor \lfloor \langle a_1, \ldots, a_7 \rangle \rfloor \rfloor = \lfloor \langle a_1, \ldots, a_4 \rangle \rfloor = \langle a_1, a_2 \rangle, \\
&\mathbb{S}_{Lists}(l,5) = \lceil \lfloor \langle a_1, \ldots, a_7 \rangle \rfloor \rceil = \lceil \langle a_1, \ldots, a_4 \rangle \rceil = \langle a_3, a_4 \rangle, \\
&\mathbb{S}_{Lists}(l,6) = \lfloor \lceil \langle a_1, \ldots, a_7 \rangle \rceil \rfloor = \lfloor \langle a_5, \ldots, a_7 \rangle \rfloor = \langle a_5, a_6 \rangle, \\
&\mathbb{S}_{Lists}(l,7) = \lceil \lceil \langle a_1, \ldots, a_7 \rangle \rceil \rceil = \lceil \langle a_5, \ldots, a_7 \rangle \rceil = \langle a_7 \rangle.
\end{aligned}$$

The cell selection function $\mathbb{S}(s,n)$ then maps s to an underlying sorted list representation $StoL(s)$ and uses $\mathbb{S}_{Lists}$ to compute the nth subsequence of $StoL(s)$ with respect to the "divide and conquer" enumeration order given above.

: **covering width function** We will use a covering width function w of the form

$$w(s) = |s| - 1.$$

Given a collection of sample points s, this covering width will cause the AP-CAD lifting algorithm (cf. **Algorithm 2**) to attempt to eliminate the cell selections $\mathbb{S}(s,0)$ through $\mathbb{S}(s,|s|-1)$.

: **formula construction function** Our formula construction function $\mathbb{F} : \mathcal{L}_{\exists OR} \times R_i \to \mathcal{L}_{\exists OR}$ will accept an $\exists$ **RCF** formula φ and a set of i-dimensional sample points s and work as follows:

(1) Reduce s first by performing the partial formula evaluation process of classical partial CAD. That is, we iterate over s, and for each sample point $\vec{r}$ of s we check whether or not $QF(\varphi)[\vec{r}]$ has a truth value. If $QF(\varphi)[\vec{r}] =$ **false**, then we remove $\vec{r}$ from s. If s is made empty by this process, then

$$\mathbb{F}(\varphi, s) = (0 = 1).$$

If $QF(\varphi)[\vec{r}] =$ **true**, then

$$\mathbb{F}(\varphi, s) = \exists\vec{x}\,[\vec{x} = \vec{r} \ \wedge \ QF(\varphi)]\,.$$

Otherwise, if $\mathbb{F}(\varphi, s)$ has not yet been determined, then we continue.

(2) Let $min_j(s)$ be the minimal value ever appearing as coordinate j in a sample point in s. To be precise,

$$min_j(s) = min\{\pi_j(x) \mid x \in s\},$$

where π_j projects a sample point $x \in \mathbb{R}^i$ onto its jth coordinate.

(3) Similarly, let $max_j(s)$ be s.t.

$$max_j(s) = max\{\pi_j(x) \mid x \in s\}.$$

(4) Then,

$$\mathbb{F}(\varphi, s) = \exists\vec{x}\left[\left(\bigwedge_{j=1}^{i} x_j \geq min_j(s) \ \wedge \ x_j \leq max_j(s)\right) \ \wedge \ QF(\varphi)\right].$$

: $\exists$ RCF proof procedure

As our $\exists$ **RCF** proof procedure, we utilise the high-performance interval constraint solver RealPaver [9]. RealPaver has many solving options, and the exact operation of this $\exists$ **RCF** proof procedure in our AP-CAD theatre may be varied by changing various RealPaver parameters (more on this below). By the nature of interval constraint methods, only answers of *infeasible* (i.e., that the $\exists$ **RCF** sentence under consideration is **false** over $\mathbb{R}$) can in general be trusted. Thus, this $\exists$ **RCF** procedure returns **false** if RealPaver says the formula is infeasible, and **unknown** otherwise.

Lemma 4.1. *$\langle\langle\mathbb{S}, w\rangle, \mathbb{F}, \mathbb{P}\rangle$ as defined above is an AP-CAD stage.*

Proof. Assuming the soundness of RealPaver infeasibility judgments, the only non-trivial property to verify is that our formula construction function $\mathbb{F}$ satisfies the *relevance judgment* axioms. Let φ be an $\mathcal{L}_{\exists OR}$ formula in $x_1, \ldots, x_n$ and $s \subset \mathbb{R}^i$ a finite set of i-dimensional sample points ($1 \leq i \leq n$). We must verify that

$$\mathbf{RCF} \models \neg\mathbb{F}(\varphi, s) \quad \Longrightarrow \quad \mathcal{N}(\varphi, s),$$

and

$$\mathbf{RCF} \models \mathbb{F}(\varphi, s) \quad \Longrightarrow \quad \mathbf{RCF} \models \varphi,$$

where (restating the property a bit more concretely than its original axiomatisation in **Section 3.1**):

$\mathcal{N}(\varphi, s)$ means that no child (at any ancestral depth, i.e., in a P_{i+1}-invariant CAD of $\mathbb{R}^{i+1}$, in a P_{i+2}-invariant CAD of $\mathbb{R}^{i+2}, \ldots,$ in a P_n-invariant CAD of $\mathbb{R}^n$) of any sample point in s will satisfy $QF(\varphi)$.

For the partial formula evaluation reasoning, this is immediate. In the next case, we have that any child of any sample point in s will satisfy

$$\left(\bigwedge_{j=1}^{i} x_j \geq min_j(s) \ \wedge \ x_j \leq max_j(s)\right),$$

and so

$$\mathbf{RCF} \models \neg\mathbb{F}(\varphi, s) \quad \Longrightarrow \quad \mathcal{N}(\varphi, s)$$

obviously holds. In the second case,

$$\mathbf{RCF} \models \mathbb{F}(\varphi, s) \quad \Longrightarrow \quad \mathbf{RCF} \models \varphi$$

is immediate. □

Finally, we will turn this AP-CAD stage $\langle\langle \mathbb{S}, w\rangle, \mathbb{F}, \mathbb{P}\rangle$ into an AP-CAD theatre $\mathbb{T}$ in a straight-forward fashion:

$$\mathbb{T}(n) = \langle\langle \mathbb{S}, w\rangle, \mathbb{F}, \mathbb{P}\rangle.$$

That is, the same stage $\langle\langle \mathbb{S}, w\rangle, \mathbb{F}, \mathbb{P}\rangle$ will be used at every dimension during AP-CAD lifting.

4.2. **Experiments.** We have conducted experiments with this AP-CAD theatre. To study its performance, we gathered a large collection (≈ 740) of real-world $\exists$ **RCF** benchmark problems arising from the MetiTarski [1] project. MetiTarski is an automated theorem prover for an undecidable extension of **RCF** involving transcendental functions such as sin, cos, e^x and log. MetiTarski works to prove universally quantified boolean combinations of inequalities in this extended theory by reasoning about families of upper and lower bounds for the functions expressed as rational functions. In the process of reasoning about these rational functions, MetiTarski derives sequences of $\exists$ **RCF** subproblems. The formulas we consider in our experiments arise in such sequences of subproblems.

For example, one series of $\exists$ **RCF** problems we consider called `sqrt-1mcosq-7` arises during MetiTarski's proof of the following theorem:

$$\forall x, y \in \mathbb{R} \left(0.05 \le x \ \wedge \ x < y \ \wedge \ y \le \frac{\pi}{2} \quad \Longrightarrow \quad \frac{x}{y} \le 0.1 + \frac{\sqrt{1 - cos(x)^2}}{\sqrt{1 - cos(y)^2}}\right).$$

During its proof of this theorem, MetiTarski generates thousands of $\exists$ **RCF** subproblems. One typical subproblem is the existential closure of the following formula, which is **false** over $\mathbb{R}$:

$$\begin{gathered}
y^4 > 4y^2 \wedge \frac{y^{14}}{87178291200} + \frac{y^{10}}{3628800} + \frac{y^6}{720} + \frac{y^2}{2} > \frac{y^{12}}{479001600} + \frac{y^8}{40320} + \frac{y^4}{24} + 1 \ \wedge \\
y^2\left(y^8 + 5040y^4 + 1814400\right) > 90\left(y^8 + 1680y^4 + 40320\right) \ \wedge \\
y^2\left(y^4 + 360\right) > 30\left(y^4 + 24\right) \wedge y^2 > 2 \ \wedge \\
\frac{x^{14}}{87178291200} + \frac{x^{10}}{3628800} + \frac{x^6}{720} + \frac{x^2}{2} < \frac{x^{16}}{20922789888000} + \frac{x^{12}}{479001600} + \frac{x^8}{40320} + \frac{x^4}{24} + 1 \ \wedge \\
\frac{x^{12}}{479001600} + \frac{x^8}{40320} + \frac{x^4}{24} + 1 < \frac{x^{14}}{87178291200} + \frac{x^{10}}{3628800} + \frac{x^6}{720} + \frac{x^2}{2} \ \wedge \\
132x^2\left(x^8 + 5040x^4 + 1814400\right) < x^{12} + 11880x^8 + 19958400x^4 + 479001600 \ \wedge \\
x^2\left(x^8 + 5040x^4 + 1814400\right) > 90\left(x^8 + 1680x^4 + 40320\right) \ \wedge \\
x^2\left(x^4 + 360\right) > 30\left(x^4 + 24\right) \wedge x^2 \ge 2 \wedge x < 0 \wedge x > y \wedge 20y > 1 \wedge 2x < z \ \wedge \\
10000000z < 31415927 \wedge 5000000z > 15707963
\end{gathered}$$

To experiment with our AP-CAD theatre, we compared its execution on a large collection of MetiTarski $\exists$ **RCF** subproblems, both with respect to the number of cells retained and total CPU time spent. For instance, upon the above problem, P-CAD retains 60 cells and takes 5.615 CPU sec, while our AP-CAD theatre retains 0 cells and takes 4.386 CPU sec. In total, we considered 742 $\exists$ **RCF** subproblems arising from six MetiTarski transcendental function problems[10]. Of these 742 problems, 327 are **false** over $\mathbb{R}$, while 415 are **true**. These problems are all full-dimensional (involving only strict inequalities), and we thus are able to make use of some efficiency enhancements for the core CAD machinery which underlies both the P-CAD and AP-CAD method we consider. These enhancements involve appealing to McCallum's theorem [12], which allows us to only consider full-dimensional cells (selecting rational sample points, never having to compute with irrational real algebraic numbers), and the use of the Brown-McCallum projection operator [4], which among its many virtues allows us to avoid the costly computation of subresultants.

Broadly summarising: 27% of problems have less cells retained by the AP-CAD method than by normal P-CAD. Of these problems for which AP-CAD causes a cell reduction, AP-CAD on average reduces the number of cells retained by 89%. For problems which are **false** over $\mathbb{R}$, this cell reduction results in an average of 15% reduction in total CPU time. For problems which are **false** over $\mathbb{R}$ s.t. AP-CAD causes cell reduction and reduces the total CPU time, the average reduction in CPU time is 39%. Over all problems in which AP-CAD reduced cell reduction (including also those problems which are **true** over $\mathbb{R}$), the average total CPU time saved is roughly 2%. For 72 out of the 742 problems, AP-CAD completely reduces the number of cells retained to 0 while P-CAD retains a non-zero number of cells (at least 14 on average, and in 4 cases as many as 200). Each of these 72 problems is **false** over $\mathbb{R}$, and this collection comprises 22% of the **false** problems. The average CPU time saved by AP-CAD for these problems is 41%. Over all 742 problems, the cummulative ratio of $\frac{\text{(total P-CAD CPU time)}}{\text{(total AP-CAD CPU time)}}$ is roughly 73%. Over the 327 **false** problems, this cummulative ratio is roughly 86%. In the following three tables, we present a collection of (psuedo-)randomly sampled results from our experiments. The first table consists of results sampled from all 742 problems. The second table consists of results sampled from the **false** problems. The third table consists of results sampled from the **false** problems for which only AP-CAD was able to eliminate all cells.

Finally, let us close with some experimental observations. First, the AP-CAD instance we constructed and experimented with in this section is but one of many possible such instances. The fact that even such a simple instance of the AP-CAD paradigm shows promise is encouraging. By virtue of the fact that the judgments of RealPaver (and other similar interval constraint solvers) can only be trusted when they decide an $\exists$ **RCF** problem to be infeasible (**false** over $\mathbb{R}$), it is perhaps not surprising that this AP-CAD theatre did not generally improve upon the performance of P-CAD for problems **true** over $\mathbb{R}$ (and these **true** problems comprise a majority of our benchmark set). Also, RealPaver is a powerful engine, and its execution comes with a nontrivial overhead. We have not begun to take advantage of RealPaver's large collection of solving heuristics. We have only used it in a naive way, as it comes "out of the box." Still, using it in the context of AP-CAD execution upon so many derived subproblems, a majority of which are **true** and thus cannot benefit from the RealPaver judgements, seems nevertheless to cause a relatively small slowdown overall. For the problems for which the AP-CAD helped, it was in many cases able to reduce the number of cells retained and the total CPU time taken by substantially. Though these experiments are most preliminary and should be significantly extended, we

[10]These top-level MetiTarski problems may be found in the MetiTarski benchmark set, available at `http://www.cl.cam.ac.uk/~lp15/papers/Arith/`. The problems considered are `polypaver-bench-exp-3d`, `polypaver-bench-sqrt`, `polypaver-sqrt-circles-1a`, `polypaver-sqrt-circles-2a`, `polypaver-sqrt43-int-3vars`, and `sqrt-1mcosq-7`.

problem	p cells	ap cells	p secs	ap secs	status
polypaver-bench-sqrt-3d-0205	36	36	0.215	0.686	false
polypaver-bench-exp-3d-0132	8	8	0.012	0.309	true
polypaver-bench-sqrt-3d-0141	0	0	0.027	0.027	false
sqrt-1mcosq-7-0036	9	9	0.067	0.163	true
sqrt-1mcosq-7-0142	12	12	0.136	0.316	true
polypaver-bench-sqrt-3d-0267	0	0	0.040	0.037	false
sqrt-1mcosq-7-0209	96	12	6.837	6.037	true
sqrt-1mcosq-7-0021	4	4	0.017	0.046	true
polypaver-sqrt43-int-3vars-0046	40	40	0.822	1.743	false
polypaver-bench-sqrt-3d-0275	60	0	0.455	0.136	false
polypaver-bench-sqrt-3d-0302	120	120	0.719	2.163	true
polypaver-bench-exp-3d-0130	10	0	0.132	0.121	false
sqrt-1mcosq-7-0138	8	8	0.101	0.181	true
polypaver-bench-sqrt-3d-0207	50	50	0.285	0.571	false
polypaver-bench-exp-3d-0038	3	3	0.004	0.003	true
polypaver-bench-exp-3d-0027	4	4	0.025	0.027	false
sqrt-1mcosq-7-0225	12	0	0.246	0.139	false
sqrt-1mcosq-7-0117	2	2	0.013	0.010	true
polypaver-bench-sqrt-3d-0395	132	36	0.999	1.751	true
sqrt-1mcosq-7-0153	56	16	2.891	2.280	true

Figure 1: A sample of twenty results from the set of all problems

problem	p cells	ap cells	p secs	ap secs	status
polypaver-bench-sqrt-3d-0166	0	0	0.027	0.028	false
polypaver-bench-exp-3d-0027	4	4	0.025	0.027	false
polypaver-bench-exp-3d-0099	12	0	0.080	0.035	false
polypaver-bench-exp-3d-0055	8	8	0.057	0.061	false
polypaver-sqrt43-int-3vars-0049	0	0	0.657	0.620	false
polypaver-bench-sqrt-3d-0262	90	0	0.472	0.147	false
polypaver-bench-exp-3d-0109	10	0	0.062	0.049	false
polypaver-sqrt43-int-3vars-0106	32	32	1.063	1.276	false
sqrt-1mcosq-7-0126	0	0	0.013	0.028	false
polypaver-bench-sqrt-3d-0110	0	0	0.007	0.006	false
polypaver-bench-exp-3d-0090	12	0	0.049	0.035	false
polypaver-bench-sqrt-3d-0318	170	170	1.048	4.309	false
polypaver-bench-exp-3d-0102	30	30	0.112	0.230	false
polypaver-bench-sqrt-3d-0300	0	0	0.104	0.102	false
polypaver-bench-sqrt-3d-0288	50	0	0.352	0.121	false
sqrt-1mcosq-7-0084	0	0	0.051	0.050	false
polypaver-bench-sqrt-3d-0343	189	0	1.489	0.705	false
sqrt-1mcosq-7-0093	4	4	0.049	0.046	false
polypaver-bench-exp-3d-0147	0	0	0.009	0.009	false
polypaver-sqrt-circles-2a-0016	0	0	0.006	0.006	false

Figure 2: A sample of twenty results from the set of false problems

problem	p cells	ap cells	p secs	ap secs	status
polypaver-bench-exp-3d-0088	10	0	0.055	0.038	false
sqrt-1mcosq-7-0045	3	0	0.027	0.018	false
polypaver-bench-sqrt-3d-0371	132	0	0.987	0.473	false
polypaver-bench-exp-3d-0099	12	0	0.080	0.035	false
polypaver-sqrt43-int-3vars-0048	54	0	3.961	3.755	false
polypaver-bench-exp-3d-0130	10	0	0.132	0.121	false
polypaver-bench-exp-3d-0079	10	0	0.085	0.067	false
polypaver-bench-sqrt-3d-0283	64	0	0.754	0.070	false
polypaver-bench-exp-3d-0143	10	0	0.137	0.120	false
polypaver-bench-exp-3d-0061	12	0	0.047	0.033	false
polypaver-bench-sqrt-3d-0342	280	0	3.704	2.442	false
sqrt-1mcosq-7-0049	3	0	0.031	0.019	false
polypaver-bench-sqrt-3d-0291	50	0	0.354	0.123	false
sqrt-1mcosq-7-0228	12	0	0.409	0.275	false
polypaver-bench-sqrt-3d-0343	189	0	1.489	0.705	false
polypaver-bench-sqrt-3d-0202	50	0	0.374	0.076	false
polypaver-bench-sqrt-3d-0235	90	0	0.473	0.146	false
sqrt-1mcosq-7-0235	12	0	0.258	0.145	false
polypaver-bench-exp-3d-0109	10	0	0.062	0.049	false
polypaver-bench-sqrt-3d-0234	50	0	0.397	0.159	false

Figure 3: A sample of twenty results where AP-CAD eliminates all cells

are encouraged by the potential this framework has for orchestrating, in a principled way, a powerful heteregeneous collection of $\exists$ **RCF** proof methods. Now that the experimental framework is in place (and made readily available through the open-source computer algebra system SAGE), we should design many new AP-CAD theatres and experiment with them heavily.

Second, though we have been working solely with full-dimensional variants of CAD-based methods, AP-CAD may prove to be even more useful when it comes to full-on CAD-based methods which require irrational algebraic number computations. The reason is that through cell selection, formula construction and proof procedure execution, one has the ability to eliminate a set of *many* sample points all at once using AP-CAD, and in this way many irrational algebraic sample points may be eliminated with only *rational* number computations. To use our concrete AP-CAD instance as an example in the context of standard CAD not restricted to full-dimensional lifting, one has the potential to eliminate a set of sample points such as $\{-3, -\sqrt{2}, -1, \sqrt{2}, \sqrt{2} + 3\sqrt{3}, 15\}$ simply by constructing and refuting a formula that only references this set of sample points using its minimal and maximal *rational* values, e.g., through a statement of the form $F \wedge (x_1 \geq -3 \ \wedge \ x_1 \leq 15)$ for some F. The ability to eliminate multiple irrational algebraic sample points simply through reasoning about formulas involving rational numbers seems very promising for the extension of these ideas to unrestricted cell decompositions.

5. Conclusion

AP-CAD allows strategic algorithmic data to be used to "short-circuit" expensive computations during the *lifting phase* of a CAD-based decision algorithm. This provides a principled approach for utilising fast, sound but possibly incomplete $\exists$ **RCF** proof procedures to enhance a *complete* decision method without threatening its completeness. We see many ways this work might be extended. It would be very interesting to work out similar machinery to be used during the *projection phase* of P-CAD. For this work to bear serious practical fruit, many more AP-CAD stages should be constructed and experimented with heavily.

References

[1] B. Akbarpour and L. Paulson. MetiTarski: An Automatic Theorem Prover for Real-Valued Special Functions. *Journal of Automated Reasoning*, 44(3):175–205, Mar. 2010.
[2] J. Avigad and H. Friedman. Combining Decision Procedures for the Reals. In *Logical Methods in Computer Science*, 2006.
[3] S. Basu, R. Pollack, and M.-F. Roy. *Algorithms in Real Algebraic Geometry.* Springer-Verlag, Secaucus, NJ, USA, 2006.
[4] C. W. Brown. Improved Projection for Cylindrical Algebraic Decomposition. *Journal of Symbolic Computation*, 32:447–465, November 2001.
[5] G. E. Collins and H. Hong. Partial Cylindrical Algebraic Decomposition for Quantifier Elimination. *J.Sym.Comp.*, 12(3):299–328, 1991.
[6] M. Daumas, D. Lester, and C. Muñoz. Verified Real Number Calculations: A Library for Interval Arithmetic. *IEEE Trans. Comp.*, 58(2):226–237, 2009.
[7] J. H. Davenport and J. Heintz. Real Quantifier Elimination is Doubly Exponential. *J. Symb. Comput.*, 5:29–35, 1988.
[8] S. Gao, M. Ganai, F. Ivancic, A. Gupta, S. Sankaranarayanan, and E. Clarke. Integrating ICP and LRA solvers for deciding nonlinear real arithmetic problems. In *FMCAD, 2010*, pages 81–89, 2010.
[9] L. Granvilliers and F. Benhamou. RealPaver: An Interval Solver using Constraint Satisfaction Techniques. *ACM TRANS. ON MATHEMATICAL SOFTWARE*, 32:138–156, 2006.
[10] J. Harrison. Verifying Nonlinear Real Formulas via Sums of Squares. In *TPHOLs'07*, pages 102–118, Berlin, Heidelberg, 2007. Springer-Verlag.
[11] H. Hong. Comparison of Several Decision Algorithms for the Existential Theory of the Reals. Technical report, RISC, 1991.
[12] S. McCallum. Solving Polynomial Strict Inequalities using Cylindrical Algebraic Decomposition. *The Computer Journal*, 36(5), 1993.
[13] G. O. Passmore. *Combined Decision Procedures for Nonlinear Arithmetics, Real and Complex.* PhD thesis, University of Edinburgh, 2011.
[14] G. O. Passmore and P. B. Jackson. Combined Decision Techniques for the Existential Theory of the Reals. In *Calculemus'09*, 2009.
[15] F. Pfender and G. M. Ziegler. Kissing Numbers, Sphere Packings, and Some Unexpected Proofs. *Notices of the A.M.S.*, 51:873–883, 2004.
[16] A. Platzer, J.-D. Quesel, and P. Rümmer. Real World Verification. In *CADE-22*, pages 485–501, Berlin, Heidelberg, 2009. Springer-Verlag.
[17] W. Stein et al. *Sage Mathematics Software (Version x.y.z).* The Sage Development Team, YYYY. `http://www.sagemath.org`.
[18] A. Tarski. *A Decision Method for Elementary Algebra and Geometry.* RAND Corporation, 1948.
[19] A. Tiwari. An Algebraic Approach for the Unsatisfiability of Nonlinear Constraints. In *CSL 2005*, volume 3634 of *LNCS*, pages 248–262. Springer, 2005.