ARTICLE IN PRESS
YJSCO:1627

Journal of Symbolic Computation ••• (••••) •••–•••

Contents lists available at ScienceDirect

Journal of Symbolic Computation

www.elsevier.com/locate/jsc

Truth table invariant cylindrical algebraic decomposition ☆

Russell Bradford [a], James H. Davenport [a], Matthew England [b], Scott McCallum [c], David Wilson [a]

[a] *Department of Computer Science, University of Bath, Bath, BA2 7AY, UK*
[b] *School of Computing, Electronics and Maths, Faculty of Engineering, Environment and Computing, Coventry University, Coventry, CV1 5FB, UK*
[c] *Department of Computing, Macquarie University, NSW 2109, Australia*

ARTICLE INFO

Article history:
Received 21 December 2014
Accepted 21 October 2015
Available online xxxx

MSC:
68W30
03C10

Keywords:
Cylindrical algebraic decomposition
Equational constraint

ABSTRACT

When using cylindrical algebraic decomposition (CAD) to solve a problem with respect to a set of polynomials, it is likely not the signs of those polynomials that are of paramount importance but rather the truth values of certain quantifier free formulae involving them. This observation motivates our article and definition of a Truth Table Invariant CAD (TTICAD).

In ISSAC 2013 the current authors presented an algorithm that can efficiently and directly construct a TTICAD for a list of formulae in which each has an equational constraint. This was achieved by generalising McCallum's theory of reduced projection operators. In this paper we present an extended version of our theory which can be applied to an arbitrary list of formulae, achieving savings if at least one has an equational constraint. We also explain how the theory of reduced projection operators can allow for further improvements to the lifting phase of CAD algorithms, even in the context of a single equational constraint.

The algorithm is implemented fully in Maple and we present both promising results from experimentation and a complexity analysis showing the benefits of our contributions.

☆ This work was supported by EPSRC grant EP/J003247/1.

E-mail addresses: R.J.Bradford@bath.ac.uk (R. Bradford), J.H.Davenport@bath.ac.uk (J.H. Davenport), Matthew.England@coventry.ac.uk (M. England), Scott.McCallum@mq.edu.au (S. McCallum), David.John.Wilson@me.com (D. Wilson).

http://dx.doi.org/10.1016/j.jsc.2015.11.002

1. Introduction

A *cylindrical algebraic decomposition* (CAD) is a decomposition of $\mathbb{R}^n$ into cells arranged cylindrically (meaning the projections of any pair of cells are either equal or disjoint) each of which is (semi-)algebraic (describable using polynomial relations). CAD is a key tool in real algebraic geometry, offering a method for quantifier elimination in real closed fields. Applications include the derivation of optimal numerical schemes (Erascu and Hong, 2014), parametric optimisation (Fotiou et al., 2005), robot motion planning (Schwartz and Sharir, 1983), epidemic modelling (Brown et al., 2006), theorem proving (Paulson, 2012) and programming with complex functions (Davenport et al., 2012).

Traditionally CADs are produced *sign-invariant* to a given set of polynomials (the signs of the polynomials do not vary within each cell). However, this gives far more information than required for most applications. Usually a more appropriate object is a *truth-invariant* CAD (the truth of a logical formula does not vary within cells).

In this paper we generalise to define *truth table invariant* CADs (the truth values of a list of quantifier-free formulae do not vary within cells) and give an algorithm to compute these directly. This can be a tool to efficiently produce a truth-invariant CAD for a parent formula (built from the input list), or indeed the required object for solving a problem involving the input list. Examples of both such uses are provided following the formal definition in Section 1.2. We continue the introduction with some background on CAD, before defining our object of study and introducing some examples to demonstrate our ideas which we will return to throughout the paper. We then conclude the introduction by clarifying the contributions and plan of this paper.

1.1. Background on CAD

A *Tarski formula* $F(x_1, \ldots, x_n)$ is a Boolean combination $(\wedge, \vee, \neg, \rightarrow)$ of statements about the signs, $(= 0, > 0, < 0$, but therefore $\neq 0, \geq 0, \leq 0$ as well), of certain polynomials $f_i(x_1, \ldots, x_n)$ with integer coefficients. Such statements may involve the universal or existential quantifiers $(\forall, \exists)$. We denote by QFF a *quantifier-free Tarski formula*.

Given a quantified Tarski formula

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

(where $Q_i \in \{\forall, \exists\}$ and F is a QFF) the *quantifier elimination problem* is to produce $\psi(x_1, \ldots, x_k)$, an equivalent QFF to (1).

Collins developed CAD as a tool for quantifier elimination over the reals. He proposed to decompose $\mathbb{R}^n$ cylindrically such that each cell was sign-invariant for all polynomials f_i used to define F. Then ψ would be the disjunction of the defining formulae of those cells c_i in $\mathbb{R}^k$ such that (1) was true over the whole of c_i, which due to sign-invariance is the same as saying that (1) is true at any one *sample point* of c_i.

A complete description of Collins' original algorithm is given by Arnon et al. (1984a). The first phase, *projection*, applies a projection operator repeatedly to a set of polynomials, each time producing another set in one fewer variables. Together these sets contain the *projection polynomials*. These are used in the second phase, *lifting*, to build the CAD incrementally. First $\mathbb{R}$ is decomposed into cells which are points and intervals corresponding to the real roots of the univariate polynomials. Then $\mathbb{R}^2$ is decomposed by repeating the process over each cell in $\mathbb{R}$ using the bivariate polynomials at a sample point. Over each cell there are *sections* (where a polynomial vanishes) and *sectors* (the regions between) which together form a *stack*. Taking the union of these stacks gives the CAD of $\mathbb{R}^2$. This is repeated until a CAD of $\mathbb{R}^n$ is produced. At each stage the cells are represented by (at least) a sample point and an index: a list of integers corresponding to the ordered roots of the projection polynomials which locates the cell in the CAD.

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