Analyzing Qualitative Spatio-Temporal Calculi using Algebraic Geometry

Diedrich Wolter
SFB/TR 8, Universität Bremen

Qualitative spatial reasoning is based on calculi which comprise relations and operation tables that encode operations like relation composition. Designing a calculus involves determining these tables and analyzing reasoning properties—a demanding task that is susceptible to errors if performed manually. This paper is concerned with automating computation of operation tables and analysis of qualitative calculi over real-valued domains like the plane $\mathbb{R}^2$. We present an approach to specify qualitative relations using polynomial equations that allows methods from algebraic geometry to be applied. This paper shows how reasoning with qualitative relations can be posed algebraically and demonstrates algebraic reasoning using Gröbner base analysis. We evaluate this approach and describe our implementation which is freely available as part of the spatial reasoning toolbox SparQ.
Keywords: qualitative spatio-temporal reasoning, algebraic geometry, geometric theorem proving

1   Introduction

Qualitative spatio-temporal reasoning (QSR) is the subfield of knowledge representation and symbolic reasoning concerned with knowledge about spatial and temporal domains on the basis of finite sets of so-called *qualitative relations* (Cohn & Hazarika, 2001; Renz & Nebel, 2007). One particular aim of QSR is to grasp human common-sense understanding of space and time. Qualitative representations are compact and enable complex decision tasks. They are valuable, for instance, in human-machine interaction contexts, in GIS, mobile robot navigation, or in any application reaching for near human-level spatial intelligence.

Technically, qualitative representations are based on relations that partition a domain into discrete, meaningful parts. These relations are used to represent knowledge by stating relationships between variables that range over the spatial or temporal domain. In other words, qualitative relations constrain the valuation of variables. Thus, qualitative spatial reasoning is related to constraint-based reasoning (CBR) and both fields share much terminology and apply similar techniques. In QSR, the algebraic closure algorithm—an adaption of enforcing path-consistency in CBR—is combined with a backtracking search and forms the central method for deciding consistency in qualitative representations (Renz & Nebel, 2007). As a prerequisite, certain operations have to be defined. Relations and operations together constitute the relation algebraic structure called a *qualitative calculus* (Ligozat & Renz, 2004). Qualitative calculi of binary relations often meet the conditions of relation algebras in the sense of Tarksi.

Commonly, calculus operations are defined by giving exhaustive tables. Surveying literature that introduces new calculi we observe that many operation tables have been computed manually and have not been verified formally. Computing operation tables is a demanding task: all possible object constellations in an infinite domain need to be evaluated—errors occur easily with fatal effects for reasoning. Operation tables can be very large which increases the risk that they contain an erroneous entry. Furthermore, the task of determining table contents is very time-consuming. For example, the (binary) composition operation lists all relation triples which can hold between three objects. So, in case of Allen's interval algebra (Allen, 1983) that defines 13 relations we have $13^3 = 2197$ combinations to evaluate and for Moratz' coarse-grained dipole calculus (Moratz, Renz, & Wolter, 2000) we have $24^3 = 13824$.[1] Hence, automated computation of composition tables is desirable.

Furthermore, automated computation and verification of operation tables and analysis of calculus properties adds new possibilities to QSR: besides modeling general domain structures qualitatively, qualitative representations can be task-specific as they aim at making exactly the distinctions necessary to tackle a problem at hand (cp. Freksa, 2004). Ultimately, we want to be able to instantiate task-specific calculi on-the-fly. To this end, automatic calculus analysis is required.

The work presented in this article draws motivation from the challenges of determining qualitative reasoning operations and analyzing reasoning properties. Our aim is to develop a generic method that is applicable to a wide range of qualitative representations and that provides an effective tool for anybody designing a qualitative calculus. We investigate into algebraic geometry for analyzing qualitative calculi over real-valued domains $\mathbb{R}^n$ and develop an algebraic modeling of qualitative relations that allows us to pose calculus analysis algebraically. We are confronted with the problem of deciding joint satisfiability of a set of polynomial

---

[1] As Bennett (1994) notes, by exploiting inherent symmetries one can reduce the number of configurations to evaluate, but there remains an $O(n^3)$ dependency with respect to the number of relations.

equations. By customizing algebraic techniques to the kind of equations that arise we obtain an effective tool whose utility we evaluate experimentally. The method described in this article is an advancement of a previous approach presented in (Wolter & Moshagen, 2008). Our method is provided as part of the free reasoning toolbox SparQ[2] and we explain how it can be used.

Our approach is applicable to any qualitative calculus which represents knowledge over a real-valued domain using basic entities which can be described using real-valued variables. This includes temporal calculi such as temporal interval calculi like Allen's interval algebra (Allen, 1983) or the INDU calculus (Pujari & Sattar, 1999) and spatial calculi like cardinal direction calculi (Frank, 1991), the $\mathcal{STAR}$ calculus (Renz & Mitra, 2004), $\mathcal{OPRA}_n$ (Moratz, 2006) et cetera, but the approach is applicable to the RCC calculus (Randell, Cui, & Cohn, 1992) which represents knowledge about region connectivity in a general topological space.

This paper is organized as follows. In Section 2 we briefly summarize some fundamental definitions in QSR and in Section 3 we present an overview of related work on automated calculus analysis. We then describe algebraic reasoning (Section 4) and detail the techniques employed in our work. Section 5 and 6 detail algebraic specifications for modeling qualitative relations. Section 7 discusses specifics of our implementation and in Section 8 we report on the experimental evaluation. We conclude by analyzing the results and discussing future research directions. In the Appendix we describe how the algebraic reasoning methods presented here can be used with SparQ.

2   Qualitative Constraint Calculi

Qualitative constraint calculi are characterized by the finite set of relations to partition the domain, formally we have:

Definition 1 (qualitative relations)**.** *Let* $\mathcal{B} = \{B_1, \ldots, B_k\}$ *be a set of* n-*ary relations over a domain* $\mathcal{D}$, *i.e.* $B_i \subseteq \mathcal{D}^n, i = 1, 2, \ldots, k$. *The set* $\mathcal{B}$ *is called the set of* base relations. *The set of all possible unions of base relations*

$$\mathcal{R}_\mathcal{B} := \left\{ \bigcup_{i \in I} B_i \,\middle|\, I \in 2^{\{1,2,\ldots,k\}} \right\}$$

is called the set of qualitative relations *or* general relations. *A qualitative relation* $R = B_{i_1} \cup B_{i_2} \cup \ldots \cup B_{i_j}$ *is commonly denoted* $\{B_{i_1}, B_{i_2}, \ldots, B_{i_j}\}$. *The relation* $R_e = \emptyset$ *is called the* empty relation *or* impossible relation *and* $R_U = B_{i_1} \cup B_{i_2} \cup \ldots \cup B_{i_k}$ *is called the* universal relation.

Base relations are required to meet the JEPD property, i.e., they are jointly exhaustive and pairwise disjoint. The JEPD property ensures that we have a normal form

---

for representing knowledge, the empty relation only occurring in contradicting, unsatisfiable knowledge bases. With the usual set-theoretic operations qualitative relations form a Boolean set algebra.

For reasoning about argument permutation, calculi of binary relations define the converse operation that allows us to infer from a fact $R(a, b)$ that $R^{\smile}(b, a)$ holds where $R^{\smile}$ is called the converse of R. In ternary and higher arity calculi other permutation operators need to be defined, commonly these are inverse $(R(a, b, c) \Leftrightarrow R^{inv}(b, a, c))$, shortcut $(R(a, b, c) \Leftrightarrow R^{sc}(c, a, b))$, and homing $(R(a, b, c) \Leftrightarrow R^{hom}(c, b, a))$. The most important operation in qualitative reasoning is the composition operation.

Definition 2 (binary composition). *The* composition *of two n-ary relations R, S over a domain $\mathcal{D}$ is defined as*

$$R \circ S := \{(x_1, \ldots, x_n) | \exists x \in \mathcal{D} :$$
$$R(x_1, \ldots, x_{n-1}, x) \wedge S(x_2, \ldots, x_{n-1}, x, x_n)\} \quad (1)$$

Here we have given the definition of the binary composition operation which is commonly used. In ternary and higher arity calculi alternative operations can also be considered (see Condotta, Ligozat, & Saade, 2006).

Operations define relations, i.e. subsets of $\mathcal{D}^n$. It depends on the qualitative relations at hand whether the calculus is closed under an operation, i.e., whether the result of applying an operation to a qualitative relation also yields a qualitative relation. As a simple counter-example consider the set of base relations $\{<, =, >\}$ over the domain of natural numbers. The composition $< \circ <$ yields $\{(x, z) | \exists y \in \mathbb{N} : x < y < z\}$, or, informally, "larger by at least 2" which is not expressible as qualitative relation using $\{<, =, >\}$. In cases like these, QSR employs so-called *weak* operations which approximate the true operation. Technically speaking, a weak operation Φ is the tightest upper approximation of an operation such that the relations are closed under Φ.

Definition 3 (weak composition). *The* weak composition *of two n-ary relations R, S in a calculus with base relations $\mathcal{B} = \{B_1, \ldots, B_k\}$ over a domain $\mathcal{D}$ is defined as*

$$R \diamond S := \bigcup \{B_i \mid B_i \in \mathcal{B}, B_i \cap (R \circ S) \neq \emptyset\} \quad (2)$$

Operations are called *strong* if they are closed on the set qualitative relations $\mathcal{R}_{\mathcal{B}}$ and *weak* otherwise. For analyzing reasoning properties it can be helpful to decide whether a qualitative calculus provides a strong or weak composition operation. A more decisive piece of information is offered by the *closure under constraints* property (Renz & Ligozat, 2005), which can be viewed as a generalization of the strong composition property. Closure under constraints means that the relations in a qualitative calculus do not allow the valuation of a variable to be constrained to a sub-relation range. Deciding this property is essentially equivalent to deciding whether composition-based reasoning can decide consistency.

3 Analyzing Constraint Calculi

One can differentiate two main categories of approaches to analyze qualitative constraint calculi. On the one hand side, there are approaches to determine tractable sub-algebras, given that certain calculus properties are met and the calculus operations are known (Renz & Li, 2008; Renz, 2007, for example). On the other hand side, there are methods that allow us to examine the fundamental features of a calculus, including its operations. In this article we are concerned with the latter kind. Here, in particular logic-based approaches have been pursuit, which, with the help of (semi-)automated theorem provers, provide a framework to express and to verify the inference rules qualitative calculi capture in there operation tables. This approach has, for example, been applied to the RCC calculus (Randell, Cohn, & Cui, 1992). More recently, Benthem and Bezhanishvili (2007); Aiello, Benthem, and Bezhanishvili (2003) studied logic-based methods as a backbone for qualitative reasoning. Logics have also been applied to examine the relation algebraic structure of qualitative calculi (Wölfl & Mossakowski, 2005) and, based on CASL specifications of qualitative calculi, also verification of qualitative calculi is possible (Wölfl, Mossakowski, & Schröder, 2007). In this interactive approach automated reasoners are employed to test operation tables against an axiomatized structure of the underlying domain. This approach is suitable for calculi in richly structured domains, which can be described by axioms in a decidable logic.

However, many calculi that involve directional relations (for example, $\mathcal{OPRA}_n$ (Moratz, 2006) or double cross (Freksa, 1992)) cannot be described in simple logic-based frameworks as directional relations often require the construction of specific angles (e.g., the perpendicular or $45°$) and specifying such relationships involves non-linear equations. Therefore, we are particularly interested in methods that can cope with non-linear equations.

In this work we examine an approach closely related to algebraic geometry, which is the field of research that is involved with solving systems of multivariate polynomial equations, i.e., equations with multiple unknowns. We apply techniques developed in this area (Kapur, 1998; Cox, Little, & O'Shea, 1998) by posing calculus properties or calculus operations as geometric theorems which we then try to prove. On the theoretical side, the contribution of this work is the design of the algebraic model of qualitative calculi as well as the adaption of algebraic techniques to provide us with an effective method for analyzing these calculi. The method of algebraic reasoning as presented here is applicable to calculi which meet three conditions:

- the calculus domain is $\mathbb{R}^n$

- basis objects can be modeled algebraically

- relations can be modeled algebraically

Many spatio-temporal calculi are designed with a real-valued domain in mind and they also comprise basis objects that can be modeled algebraically, for instance points (x, y), line segments (x_1, y_1, x_2, y_2), etc. Algebraic modeling is not applicable, for example, to the RCC calculus (Randell, Cui, & Cohn, 1992) describing arbitrarily shaped regions in a general topological space. It can however be used to analyze RCC relations in $\mathbb{R}^n$ for some fixed n considering primitive regions like circles (which can be described algebraically). To the best of our knowledge no calculus has been developed that meets the first two conditions but whose base relations cannot be described as the solutions of multivariate polynomial equations. Thus, the third condition may not pose an actual restriction.

4   Algebraic Geometry

In algebraic geometry a geometric relationship is described by the set of solutions—the *zero set*—of a system of equations. For example, to describe a circle with radius 1 centered at the origin we can write $x^2 + y^2 - 1 = 0$. Methods for analyzing solutions of equation systems are of key interest in algebraic geometry.

Different methods have been proposed for algebraic geometry and geometric theorem proving, but many either utilize Gröbner bases (Buchberger, 1985), Wu's method (Wen-tsün, 1978), or Cylindrical Algebraic Decomposition (CAD) (Collins, 1975). All of these methods are designed for analyzing sets of polynomial equations. Wu's methods is related to methods utilizing Gröbner basis in that both methods aim at computing elimination polynomials. Roughly speaking, equations are rewritten by eliminating variables. Rewritten equations have the same zero set but involve fewer variables and are thus easier to handle. In our approach we use Gröbner bases. CAD is different in that it iteratively eliminates variables by computing distinct ranges for variables such that a polynomial does not change its sign over the determined range. In contrast to approaches based on Gröbner bases or Wu's method, CAD requires more sophisticated techniques to compute with algebraic numbers, i.e., symbolically to compute with terms containing roots (see Basu, Pollack, & Roy, 2006, Chapter 11). The advantage of CAD is its ability to compute a sample point in every component of the zero set. On the downside, and setting aside its challenging implementation, CAD becomes computationally infeasible for complicated systems of equations.

In principle also numerical techniques for solving non-linear equations are of interest. Such methods commonly require the set of roots to be comprised of single, discrete points. However, the set of configurations that satisfies a qualitative relation is usually at least scale-independet, i.e., the zero set is an extended region in parameter space. In our case—as we discuss later—the set of real roots we are interested in touches complex roots we are not interested in. In presence of such gradual changeovers numerical methods seem particularly difficult to apply.

4.1 Ideal Bases and Gröbner Bases

In our approach we use multivariate polynomials to model geometric relations, i.e., polynomial equations that involve multiple variables. For analyzing these equations we simplify them by computing the Gröbner basis introduced by Buchberger (1985). In the following we give a very brief account of algebraic reasoning with Gröbner bases, restricted to the level of detail necessary here. The interested reader is pointed to the literature (Cox et al., 1998; Becker & Weisspfennig, 1998; Kapur, 1998).

Definition 4. *The set of all polynomials $f(x_1, \ldots, x_n)$ with coefficients from a field F is called the polynomial ring*

$$F[x_1, \ldots, x_n] = \left\{ f = \sum_{|\alpha| \leq k} a_\alpha x^\alpha, k \in \mathbb{N}_0, \alpha \in \mathbb{N}_0^n, a_\alpha \in F \right\}$$

We call x^α a monomial and $\alpha = (\alpha_1, \ldots, \alpha_n)$ the exponent vector. For example, the monomial $x^{(1,0,3)}$ stands for $x_1^1 \cdot x_2^0 \cdot x_2^3$.

We restrict ourselves to the field of reals as many qualitative calculi are defined over a real-valued domain. The common zero set of a set of polynomials generates an ideal. Roughly speaking, an ideal is the set of all polynomials sharing the same common zero set.

By constructing specific polynomials we model a geometric relationship as zero set and then analyze the ideal generated. Our aim is to look for other polynomials inside the ideal which are particularly easy to handle. This is legitimate since all polynomials inside the ideal have the same zero set as the original polynomials. Ideals can be described by finite bases, Gröbner bases are "just" a specific ideal basis which appears to be particularly easy to handle. Gröbner bases can be computed by canceling down polynomials using polynomial division—this is known as Buchberger's algorithm (Buchberger, 1985). Polynomial division with multivariate polynomials requires fixing a monomial ordering. In contrast to the univariate case where the ordering of terms is naturally given by exponents (e.g., x^4 is the lead term of $x^4 + x - 2$), the multivariate case offers a great variety of orderings. Since polynomial division proceeds by the order of terms, the choice of the monomial ordering affects the outcome of polynomial division and, hence, the Gröbner basis itself.

Definition 5 (monomial ordering)**.** *Let x^α be a monomial and $\alpha \in \mathbb{N}_0^n$ be the exponent vector. We call $\prec \subset \mathbb{N}_0^n \times \mathbb{N}_0^n$ a* monomial ordering *if the following properties are satisfied:*

1. $\prec$ is total, $\forall \alpha, \beta \in \mathbb{N}_0^n$ we have either $x^\alpha \prec x^\beta$ or $x^\beta \prec x^\alpha$

2. $\prec$ is monotonous, $\forall \alpha, \beta, \gamma \in \mathbb{N}_0^n : x^\alpha \prec x^\beta \rightarrow x^{\alpha+\gamma} \prec x^{\beta+\gamma}$

3. $\prec$ is a well-ordering on $\mathbb{N}_0^n$, i.e. there exists a well-defined minimum wrt. $\prec$.

There are two important representatives of monomial orderings: lexicographic and graded lexicographic ordering. In lexicographic ordering $<_{\text{lex}}$ we have $x^\alpha <_{\text{lex}} x^\beta$ if for the first differing position i of α and β we have $\alpha_i < \beta_i$. For example, consider the variables $x_1, \ldots, x_4$ and the monomials $x_1 = x^{(1,0,0,0)}$ and $x_2^2 = x^{(0,2,0,0)}$. As for the first differing position we have $1 > 0$, it holds that $x_2^2 <_{\text{lex}} x_1$ and, analogously, $x_1 x_4^4 <_{\text{lex}} x_1 x_3^2$. In the graded lexicographic ordering the monomials are first ordered by their total degree (the sum of all components in the exponent vector) and, if the total degrees are identical, further differentiated by lexicographic ordering. For example we have $x_2^3 >_{\text{grlex}} x_1 x_2$ and $x_2^3 <_{\text{grlex}} x_1 x_2 x_3$ since the total degree of x_2^3 is 3 and the total degree of $x_1 x_2$ is only 2, but the total degree of $x_1 x_2 x_3$ is also 3 and $x_2^3 <_{\text{lex}} x_1 x_2 x_3$.

When computing the Gröbner basis a normalization step can be carried out to obtain the basis in normal form, called the *reduced Gröbner basis* (see Cox, Little, & O'Shea, 2007). One remarkable effect is that we also obtain a normal form when there is no common solution of the polynomials involved:

Theorem 6 (Buchberger (1985)). *A polynomial ideal over the field $\mathbb{C}$ has an empty zero set if and only if the polynomial 1 is contained in the reduced Gröbner basis.* □

4.2 Solutions Over the Field of Reals

Since we are analyzing operations in a qualitative calculus over a real-valued domain we are interested in real-valued solutions only. Theorem 6 already provides us with straightforward means to identify the non-existence of solutions over the complex field which implies that there are no real-valued solutions either. However, if solutions over the complex field exist, we still need to evaluate existence of real-valued solutions. The problem of computing real-valued solutions has been intensively studied in mathematics and methods have been developed (see Basu et al., 2006). Due to the inherent difficulty of this problem that requires exponential runtime, all implementations remain limited to few variables only. To the best of our knowledge there exists no practical equation solver yet that can handle polynomial systems with about 10 or more variables as required in our application. However, in context of deciding consistency of a qualitative constraint problem we do not need to *compute* a solution, we only need to test whether a real-valued solution *exists*. For example, the composition operation as introduced in Definition 2 is based on Equation 1 which only involves the existence of a value, but not the value itself.

So we avoid computing a solution by designing a slender rule-based approach to decide existence of real-valued solutions. To this end, we examine the Gröbner basis and try to deduce that no real-valued solution can exist by identifying a

	rule pattern		implication
1.	$0 = x_i^k + c$	$\Rightarrow$	$x_i = \sqrt[k]{-c}$
2.	$0 = a_i x_i^k + a_j x_j^l$	$\Rightarrow$	$x_i = -\sqrt[k]{\frac{-a_j}{a_i} x_j^l}$
	Condition: k divides l		
3.	$0 = \sum\limits_{i=1,\ldots,n} a_i x_{i,1}^{2j_{i,2}} x_{i,2}^{2j_{i,2}} \cdots x_{i,n}^{2j_{i,n}}$	$\Rightarrow$	$\forall i \in \{1,\ldots,n\} : \exists k : x_{i,k} = 0$
	Condition: $\forall i,j = 1,\ldots,n \; : \; \mathrm{sgn}(a_i) = \mathrm{sgn}(a_j)$		
	Example: $0 = x_1^2 x_2^4 + x_3^2 \Rightarrow x_1 = x_3 = 0$ or $x_2 = x_3 = 0$		
4.	$0 = \prod\limits_{k=i_1,i_2\ldots,i_n} x_k^{j_k}$	$\Rightarrow$	$\exists k \in \{i_1, i_2 \ldots, i_n\} \; : \; x_k = 0$
5.	$0 = \alpha + \beta \wedge 0 = \beta + \gamma$	$\Rightarrow$	$0 = \alpha - \gamma$
6.	$1 = c x_i^{2k} x_j^{2n}$	$\Rightarrow$	$x_i \neq 0 \wedge x_j \neq 0$

Table 1: Set of transformation rules used to analyze solvability

variable for which no real value can be chosen. As a simple example, consider $x^2 + 1 = 0$ which does not have a real-valued solution. The inference process is based on the set of transformation rules listed in Table 1. Our simple example is covered by the first rule. The way we design equations to analyze calculi (detailed in the next section) does not involve numbers except for coefficients -1 and $+1$. Therefore, neither Rule 1 nor 2 requires us to compute with algebraic or, alternatively floating point, numbers. Instead, we mostly make use of Rule 3 which detects that variables must take the value zero.

Algorithm 1 summarizes our reasoning method which we now explain in detail. We start by computing the reduced Gröbner basis for a set of polynomials P using Buchberger's algorithm (Buchberger, 1985) (lines 2–14). This algorithm iteratively adds a new polynomial h to the set of polynomials and then simplifies the set until the desired basis is obtained. We extend this iteration by examining every new polynomial h added to P (lines 7–12). Our motivation is to improve the efficiency of reasoning by identifying conditions that allow us to further simplify P. Furthermore, we can identify unsatisfiable polynomials early in the process. If h matches a rule that implies a variable binding, then we apply this variable binding, thereby removing the variable from P. We immediately stop if a rule implies a binding that violates a side condition $x_i \neq 0$ implied by Rule 6 or if a binding involves a non-real value (e.g., as follows from $x_i^2 + 1 = 0$ according to Rule 1). While verifying the composition table of Allen's interval algebra, this extension allows for an early exit of the proof in 1598 cases out of the 1788 inconsistent re-

lation triples, i.e., this optimization was effective in about 73% of all 2197 relation triples.

After the Gröbner basis has been computed we continue applying rules until no more rules are applicable (lines 16–22). Since Rules 3 and 4 imply alternative variable bindings a backtracking search is required. We prioritize application of rules that allow us to remove a variable from the basis since this facilitates efficiency. For example, whenever possible we compute the value for a variable (e.g., by Rules 1 or 2) and replace the variable by its value. Rule 5 and Rule 6 are special in that they do not imply new variable bindings. While Rule 6 captures the side conditions introduced by turning an inequality $f < 0$ into an equality (see Section 5.1), Rule 5 generates a new polynomial to be analyzed. To prevent a blow-up of the set of polynomials to be analyzed, Rule 5 is only applied once to every pair of polynomials. In our implementation Rule 6 is handled separately by keeping track of a list of variables that cannot take the value zero.

Summing up, the aim of our method is to search for necessary conditions that inhibit existence of real-valued solutions. Thus, our method searches for an "proof of unsatisfiability" by using a heuristic selection of rules, preferring those that are more likely to finish the proof. We note that our approach is sound but it is not complete in the sense that not all systems of equations that do not have a real-valued solution will be identified as such. As noted before it appears to be infeasible to achieve completeness in a practical realization. Incompleteness means that whenever we cannot prove a system of polynomials to be unsatisfiable, we cannot decide satisfiability. It may either be the case that we missed to recognize that no real-valued solution can exist or, simply, that a real-valued solution exists. As we are often confronted with satisfiable cases, we complement the search for an proof of unsatisfiability with a simple constructive method to "guess" a solution (line 23). Our approach is simply to try out different valuations (choosing variables from a discrete grid) in a backtracking search. The motivation here is that the proof of unsatisfiability is effective, i.e., there are only few unsatisfiable cases not identified as such. Hence, if we start to search for a solution then we are likely to find one.

5 Algebraic Calculus Modeling

An n-ary qualitative relation R over domain $\mathcal{D}$ is modeled as zero set of a set multivariate polynomials F_R over real-valued variables $y_1, \ldots, y_k$:

$$\forall x_1, \ldots, x_n \in \mathcal{D} : R(x_1, \ldots, x_n) \Leftrightarrow$$
$$\exists y_1, \ldots, y_k \in \mathbb{R} : \forall f \in F_R \ : \ f(y_1, \ldots, y_k) = 0$$

Basis objects of the qualitative relations need to be expressible by real-valued variables. For example, a point A positioned in the plane could be represented

---

Algorithm 1 The algebraic reasoning algorithm

1: **function** DECIDESATISFIABILITY(P)   ▷ P is a set of polynomials
2:     **while** P is not a Gröbner basis **do**   ▷ Gröbner basis algorithm
3:         $(p, q) \leftarrow$ CRITICALPAIR(P)
4:         $h \leftarrow$ NORMALFORM(p, q, P)
5:         **if** $h \neq 0$ **then**
6:             $P \leftarrow P \cup \{h\}$ ▷ check rule implications for every new h
7:             **if** h matches some $r \in rules$ and r implies variable binding **then**
8:                 **if** implied binding violates side conditions **then**
9:                     **return** "not satisfiable"
10:                 **end if**
11:                 $P \leftarrow$ APPLYRULE(r, P)
12:             **end if**
13:             $P \leftarrow$ REDUCE(P)
14:         **end if**
15:     **end while**   ▷ P is now a reduced Gröbner basis
16:     **while** there is $p \in P, r \in rules$ such that p matches r **do**
17:         **try** application of rule r ▷ initiate backtracking search
18:         **if** implied binding violates side conditions or is not real-valued **then**
19:             **return** "not satisfiable"
20:         **end if**
21:         $P \leftarrow$ APPLYRULE(r, P)
22:     **end while**
23:     $P \leftarrow$ TRYVALUES(P)   ▷ if proving unsatisfiability fails, try to guess
24:     **if** $P = \{0\}$ **then**   ▷ satisfiable valuation found
25:         **return** "satisfiable"
26:     **else**
27:         **return** "don't know"   ▷ fail
28:     **end if**
29: **end function**

---

by x_A, y_A, a circle by its origin and radius, etc. Several sets of polynomials are combined to model a qualitative reasoning problem.

5.1   Resolving Inequalities

Several qualitative relations can be modeled naturally using inequalities. For example, $x < y$ could model in a real-valued temporal domain the precedence of time point x over y. Algebraic reasoning methods require us to transform inequality into equations. To accomplish this we introduce new variables s, v which we call *slack variables*:

$$
\begin{aligned}
f(x_1,\ldots,x_n) < 0 &\;\rightsquigarrow\; f(x_1,\ldots,x_n) + s^2 = 0 \;\wedge\; s^2v^2 - 1 = 0 \\
f(x_1,\ldots,x_n) \leq 0 &\;\rightsquigarrow\; f(x_1,\ldots,x_n) + s^2 = 0 \\
f(x_1,\ldots,x_n) > 0 &\;\rightsquigarrow\; f(x_1,\ldots,x_n) - s^2 = 0 \;\wedge\; s^2v^2 - 1 = 0 \\
f(x_1,\ldots,x_n) \geq 0 &\;\rightsquigarrow\; f(x_1,\ldots,x_n) - s^2 = 0 \\
f(x_1,\ldots,x_n) \neq 0 &\;\rightsquigarrow\; f(x_1,\ldots,x_n) > 0 \;\vee\; f(x_1,\ldots,x_n) < 0
\end{aligned}
$$

In case of transforming $<$ and $>$ the second equation ensures that slack variables cannot take the value 0. The use of a slack variable s in squared form s^2 ensures that the term stands for a non-negative value.

5.2 Handling Conjunction, Disjunction, and Negation

A solution to a system of equations is a solution if it satisfies all equations in the system. In other words, equation systems have conjunctive semantics and thus no effort is required to express that $f(x_1,\ldots,x_n) = 0 \wedge g(x_1,\ldots,x_n) = 0$ holds. Suppose that $f(x_1,\ldots,x_n) = 0 \vee g(x_1,\ldots,x_n) = 0$ shall be satisfied. This can be achieved by considering $f(x_1,\ldots,x_n) \cdot g(x_1,\ldots,x_n) = 0$. Negation of $f(x_1,\ldots,x_n) = 0$ can be modeled by changing the equation to $f(x_1,\ldots,x_n) \neq 0$ and then resolving the inequality as described above.

5.3 Example: Ligozat's $\mathcal{LR}$ calculus

Let us have a closer look at the $\mathcal{LR}$ calculus (Ligozat, 1993) which defines 9 ternary base relations of point positions in the plane (see Figure 1 (a)). Seven relations differentiate the case of different reference points ($A \neq B$, Figure 1 (a) left), and two relations differentiate the degenerate case of identical reference points (Figure 1 (a) right). We represent a point A by two variables x_A, y_A which represent the point's x and y coordinate. For modeling qualitative relations we use the following set of geometric primitives from which qualitative relations are constructed by conjunction:

$inFrontOf(A,B,C), notBehindOf(A,B,C)$
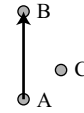
$x_C = x_A + \lambda(x_B - x_A)$, new variable λ

$y_C = y_A + \lambda(y_B - y_A)$

$inFrontOf:\; \lambda > 1$

$notBehindOf:\; \lambda > 0$

$rightOfLine(A,B,C)$

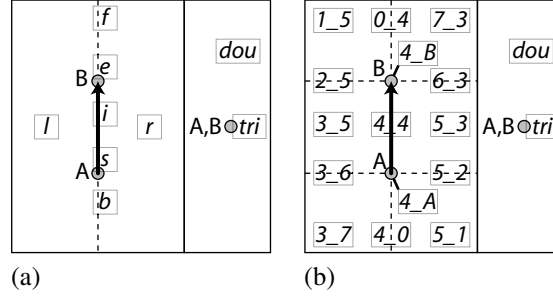$x_C \cdot y_B - x_A \cdot y_B - x_C \cdot y_A + x_A \cdot y_C - x_B \cdot y_C + x_B \cdot y_A > 0$

Figure 1.: (a) Base relations of the ternary $\mathcal{LR}$ calculus: 7 point-to-vector-relations and two degenerate cases for $A = B$ (b) Base relations of the ternary $\mathcal{DCC}$ calculus: 15 point-to-vector-relations and two degenerate cases

(derived from scalar product $<\overrightarrow{AB}, \left(\begin{smallmatrix} 0 & 1 \\ -1 & 0 \end{smallmatrix} \right) \cdot \overrightarrow{AC}>$)

$Equals(A, B)$
$x_A = x_B$
$y_A = y_B$

$NotEquals(A, B)$
$x_A^2 - 2x_A x_B + x_B^2 + y_A^2 - 2y_A y_B + y_B^2 > 0$

Using these primitives we model the $\mathcal{LR}$ relations:

$R(A, B, C)$	description using geometric primitives
s	$NotEquals(A, B) \wedge Equals(A, C)$
e	$NotEquals(A, B) \wedge Equals(B, C)$
b	$NotEquals(A, B) \wedge inFrontOf(B, A, C)$
i	$NotEquals(A, B) \wedge notBehindOf(A, B, C)$
f	$NotEquals(A, B) \wedge inFrontOf(A, B, C)$
l	$NotEquals(A, B) \wedge rightOfLine(B, A, C)$
r	$NotEquals(A, B) \wedge rightOfLine(A, B, C)$
dou	$Equals(A, B) \wedge notEquals(A, C)$
tri	$Equals(A, B) \wedge Equals(A, C)$

6   Posing Qualitative Reasoning Tasks Algebraically

We use algebraic reasoning to check whether there exists an instantiation of objects satisfying all modeled relations at the same time. In the terminology of qualitative reasoning, sets of relations over objects are called *qualitative constraint networks* and deciding existence of a satisfying instantiation is referred to as de-

ciding *consistency* of a constraint network. In other words, algebraic reasoning decides consistency of qualitative constraint networks. We use specifically constructed constraint networks to determine a variety of calculus properties.

In the following, let R, S, T stand for a base relation from the set $\mathcal{B}$ of all base relations in a given n-ary calculus and let k denote the total number of base relations.

6.1 Permutation Operations

Let π be a permutation such that $R(x_1, x_2, \ldots, x_n)$ implies $R^\pi(x_{\pi(1)}, x_{\pi(2)}, \ldots, x_{\pi(n)})$. For computing the (weak) permutation operation we construct constraint networks

$$C^\pi_{R,S} := \big\{ R(x_1, x_2, \ldots, x_n), S(x_{\pi(1)}, x_{\pi(2)}, \ldots, x_{\pi(n)}) \big\} \qquad (3)$$

for all base relations R, S. The desired relation R^π is given by

$$\{ S \in \mathcal{B} | C^\pi_{R,S} \text{ is satisfiable} \} \qquad (4)$$

Computing a permutation operation involves k^n satisfiability tests.

6.2 Composition Operation

For all triples of base relations R, S, T we construct a constraint network:

$$C^c_{R,S,T} := \{ R(x_1, \ldots, x_n), S(x_2, \ldots, x_{n+1}), T(x_1, \ldots, x_{n-1}, x_{n+1}) \} \qquad (5)$$

The network is consistent if and only if $T \subseteq R \circ S$ (cp. Definition 2). Computing the full composition operation involves k^3 satisfiability tests. Efficiency of this procedure can be improved by avoiding redundant computations as has been shown in (Bennett, 1994) for binary calculi which define a set of JEPD base relations. In order to apply this optimization, JEPDness needs to be checked and symmetric relations need to be identified.

6.3 JEPDnes, weakness of operations, symmetry, etc.

We check whether a set of base relations meets the JEPD property in two steps

- Exhaustiveness: $\neg B_1(x_1, \ldots, x_n) \wedge \ldots \wedge \neg B_k(x_1, \ldots, x_n)$ is unsatisfiable

- Disjointness: for every pair of relations B_i, B_j its conjunction $B_i(x_1, \ldots, x_n) \wedge B_j(x_1, \ldots, x_n)$ is unsatisfiable

Testing JEPDness involves $\frac{k^2-k}{2}+1$ satisfiability tests. Analogously, we can also analyze whether an operation is strong or weak. Let $R \diamond S = B_{i_1} \cup B_{i_2} \cup \ldots \cup B_{i_s}$ whereby all B_{i_j} are base relations. Composition of R and S is strong if and only if there does not exist a configuration that is not contained in the $R \diamond S$ but which is contained in of the base relation B_{i_k}. Thus, we need to check whether

$$\neg\left(R(x_1, x_2, \ldots, x_n) \wedge S(x_2, \ldots, x_{n+1})\right) \wedge B_{i_k}(x_1, \ldots, x_{n-1}, x_{n+1}) \qquad (6)$$

is unsatisfiable for all base relations B_{i_k} covered by $R \diamond S$.

.

6.4   Interdependency Tables

As introduced by Gerevini and Renz, *interdependency tables* express implications of combining relations stemming from different constraint calculi (Gerivini & Renz, 2002). For example, containment information has implications on size, a region can only contain a smaller region. Interdependency tables are used by the bipath consistency algorithm (see Gerivini & Renz, 2002) for deciding consistency of linked constraint networks over two distinct calculi.

As the algebraic approach to calculus analysis purely builds on relation semantics, constraints expressed in different calculi can directly be combined. If the joined algebraic models of relations R and S are unsatisfiable, R and S cannot hold simultaneously independent of whether R and S belong to the same calculus or not. For computing the complete interdependency table for two calculi which involve n and m base relations, $n \cdot m$ satisfiability tests are required.

Among other relation properties one can check whether a relation R is symmetric by testing the unsatisfiability of $R(x_1, x_2, \ldots, x_n) \wedge \neg R(x_n, x_{n-1}, \ldots, x_1)$.

7   SparQ Implementation

The approach presented here has been implemented and integrated into the publicly available spatial reasoning toolbox SparQ. Interpreting reasoning in a broad sense, SparQ covers mapping information from quantitative to qualitative domains, applying constraint reasoning to qualitative information, reasoning about calculi, and mapping qualitative information back to the quantitative domain. The toolbox is designed for extensibility and released under the GNU GPL public license for free software.

By default, SparQ compiles to utilize the graded lexicographic monomial ordering. The selection of critical pairs (line 3 in Algorithm 1) is performed by the method proposed in (Giovini, Mora, Niesi, Robbiano, & Traverso, 1991). This method helps to avoid critical pairs which would be reduced to a zero polynomial, thereby saving computation time. We tested our implementation of Buchberger's

algorithm against that of the C𝚘C𝚘A system to facilitate a correct implementation. Validating the complete SparQ reasoner appears to be infeasible, however. Since Computation time for determining the Gröbner basis is largely depended on the overall number of variables occurring in the equations, so we try to remove variables by

variable binding Without loss of generality we set some variables freely. In relative calculi one point may be set to 0, one interval may be set to $[0, 1]$, or one line segment may be set to $\overline{(0; 0)\,(0; 1)}$.

variable substitution Whenever the identity of variables is inferred ($x_i = x_k$, $i < k$), we immediately replace all occurrences of x_i by x_k. Substitution is performed in favor of having more slack variables (higher indices) that are subject to a side condition $x_k \neq 0$ (see Section 5.1). This replacement scheme ensures that no side condition is missed.

At the cost of code complexity further improvements are possible, for example by employing a more efficient algorithm for computing the Gröbner basis (Faugère, 1999, 2002), but the present methods proved to be reasonably fast.

The main feature of the implementation is the specific treatment of slack variables. The algebraic modeling of polynomials introduces special polynomials in the form $s^2 \cdot v^2 = 1$ to enforce a non-zero value for s and v. This side condition is reflected by Rule 6. In our implementation we collect side conditions $x_i \neq 0$ and store them separate from the polynomials. Whenever we determine a new variable binding we first check whether the binding violates one of these side conditions. In order to make this approach effective, we try to have polynomials that involve many slack variables rather than having polynomials without slack variables. This is achieved by using variables with high indices as slack variables in conjunction with the (graded) lexicographic variable ordering. This variable ordering causes Buchberger's algorithm to eliminate variables with small indices, while retaining those with high indices. The resulting equations allow for an effective application of the rule-based proof system.

Furthermore, before starting analysis of a composition table, some time is spent on randomly generating object configurations by assigning random numbers to all variables in the equation system except for the slack variables. Our implementation retains knowledge about the original inequalities used to model the qualitative relations and so it becomes an easy task to identify which qualitative relation holds for a given valuation. So we can read off consistent scenarios as they are recorded by the composition table. We do not need to analyze these configurations any further and can hence save additional computation time.

All algebraic reasoning methods are implemented in LISP, taking advantage of the dynamic memory management and exact arithmetics on rational numbers provided by this programming language.

	lexicographic			graded lexicographic		
calculus	success	total time	avg. time	success	total time	avg. time
Allen	83%	249.1s	113ms	83%	122.35s	56ms
$\mathcal{DCC}$	99%	145.9s	30ms	99%	72.3s	15ms
$\mathcal{LR}$	98%	27.4s	38ms	98%	25.5s	35ms
point	93%	3.9s	144ms	93%	3.9s	144ms

Table 2: Success rates and total computation times required to run verification of composition tables

8   Experimental Results for Verifying Operation Tables

We conduct an experimental evaluation of the presented method for selected qualitative spatial calculi using our SparQ implementation. The focus of the experiments is to evaluate the performance of the algebraic reasoning part, i.e., computing and examining the Gröbner basis. In our experiments we analyze, on the temporal side, Allen's interval algebra (Allen, 1983), the point calculus (Vilain, Kautz, & Beek, 1989), and, on the spatial side, the ternary $\mathcal{LR}$ calculus (Ligozat, 1993) and the double cross calculus ($\mathcal{DCC}$) (Freksa, 1992). An Apple Mac Book Pro (Intel Core 2 Duo 2.66 Ghz, 4GB Ram, Mac OS 10.5) serves as experimental platform, but only one core is used by SparQ. SparQ is compiled using the LISP system SBCL[3], version 1.0.38.

For each calculus we evaluate the performance for analyzing the composition table as this is the most challenging operation to analyze. All triples of relations are examined, thus involving some redundant computations (cp. Bennett, 1994). However, our interest here is to evaluate the handling of any relation combination. We set a time limit of 10 seconds per individual computation. Any computation that is not finished within the time limit is aborted. We compare the output of our system with the respective composition table given by the literature. All composition tables have been checked carefully, but we note that to the best of our knowledge none of these calculi has been formally verified yet. Our method is not complete but it is sound by design and no errors have been observed. For our evaluation we count the number of relation triples that we failed to identify as inconsistent scenarios (missed negatives) as well as the failures to construct a solution (missed positives).

The success rate of verifying a composition table is computed as the faction of all scenarios for which we could either proof inconsistency or construct a model. In Table 2 we present the overall success rates and time requirements for running the complete verification with SparQ (see Appendix B.3). The time measurements include all setup and I/O operations. Due to a randomized model construction little

---

[3]http://www.sbcl.org/

calculus	lexicographic			graded lexicographic		
	success	total time	avg. time	success	total time	avg. time
Allen	1788 / 1788	52s	24ms	1786 / 1788	30s	14ms
$\mathcal{DCC}$	4352 / 4357	254s	52ms	4352 / 4357	58s	12ms
$\mathcal{LR}$	637 / 653	166s[3]	228ms[3]	637 / 653	135s[4]	186ms[4]
point	14 / 14	—	<1ms	14 / 14	—	<1ms

[3] 16 timeouts occurred and computation was aborted after 10s
[4] 12 timeouts occurred and computation was aborted after 10s

Table 3: Computation times for testing unsatisfiability

variations of both the exact time and the success rate occur, but these are less than one percentage point in success rate and are neglected. The analysis is carried out for the lexicographic monomial ordering as well as for the graded lexicographic monomial ordering.

In Figure 2 we present the success rates for each calculus broken up into the number of models found and the number of inconsistencies detected. For all calculi, more than 80% of the scenarios have been verified. However, one can observe a difference in the success rate between proving inconsistency algebraically (above 98%) and searching for a model (as little as approximately 20% success rate in case of Allen's interval algebra).

Since our primary interest is to evaluate the algebraic proof system for showing unsatisfiability of configurations (computing the Gröbner basis and running the rule-base reasoner), we analyze this component in little more detail.

For this test we disable the randomized model generation and run the algebraic proof system on all triples of relations. Table 3 gives the results, showing the number of inconsistent scenarios detected, the total computation time, and the average time spend on analyzing a single scenario. Again, this is done separately for the lexicographic and graded lexicographic monomial ordering. Times are measured with a resolution of 1ms, so computation times below 1ms are not recorded. Therefore, we omit time measurements for the point calculus for which mostly computation times below 1ms occur. We analyze the distribution of computation times in little more detail. In Figure 3 the number of scenarios that can be analyzed within a given time slot is plotted for the $\mathcal{DCC}$ calculus. As can be seen, about half of the 4917 scenarios can be analyzed in at most 2ms each.
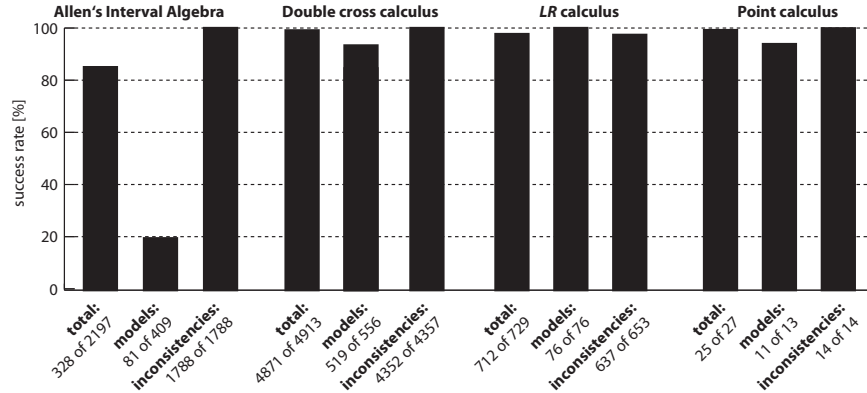
Figure 2.: Success rate of the algebraic reasoning tool in verifying composition tables. The column "total" gives the overall success rate for each calculus, columns "models" and "inconsistencies" itemize the performance for scenarios that have a model and for inconsistent scenarios.

9   Discussion of the Results

The method of algebraic reasoning is able to determine the magnitude of entries in the composition tables of the calculi analyzed. Of the two relative position calculi $\mathcal{LR}$ and $\mathcal{DCC}$, analyzing the $\mathcal{DCC}$ calculus composition required more resources. A reason for this can be seen in the more complex algebraic modeling of $\mathcal{DCC}$ relations which involves more geometric constraints and, thus, more equations.

The experiments shows in particular that most inconsistent scenarios have been identified as such. The computation time analysis shown in Figure 3 reveals that more than 90% of the scenarios have been handled in less than 100ms. This quickness of operation is due to the employed (graded) lexicographic monomial ordering in conjunction with a specific variable assignment scheme. Low indices are used to represent objects in the domain $(x_1, x_2, \ldots, x_k)$ and high indices are used for slack variables $(x_{k+1}, x_{k+2}, \ldots)$. This directs computation of the Gröbner basis towards producing equations containing only slack variables. Since these equations are subject to side conditions $x_i \neq 0$, Rule 3 and 4 (see Table 1) are effective in detecting violations. Note that one is usually able to find a solution to equations modeling an inconsistent scenario if slack variables are allowed to take the value 0 as, in the way we designed the geometric primitives, this means that points coincide and relative direction information vanishes. Unfortunately, it is very difficult to get a theoretical underpinning of this observation. Note that this also applies to the graded lexicographic ordering which gives similar results. Only in case of Allen's interval algebra there is a difference in effectivity (cp. Table 3):
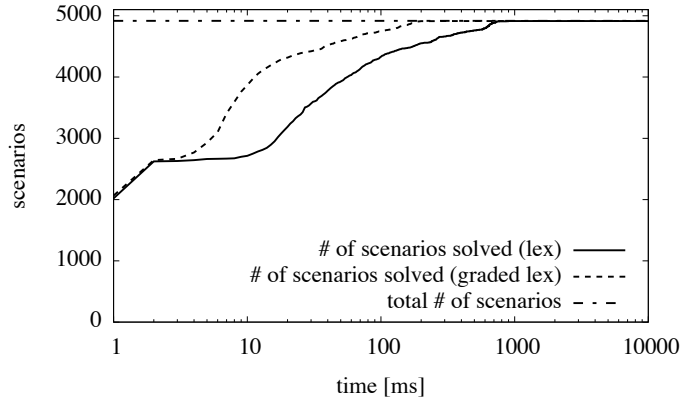
Figure 3.: $\mathcal{DCC}$ scenarios tested for unsatisfiability per computation time

using the graded lexicographic ordering two of the inconsistent scenarios were not identified whereas using the lexicographic ordering all inconsistent scenarios have been identified. With respect to the runtime, the graded lexicographic ordering allows for faster computation times (see Table 2), mainly because it helps to avoid generating polynomials with high exponents that require more time to handle.

For $\mathcal{DCC}$, the point calculus, and the $\mathcal{LR}$ calculus the success rate of the search for a scenario is similar to that of identifying inconsistencies. However, in case of Allen's interval calculus, the method performs poorly with only a success rate of approximately 20%. The poor performance results from the specifics of the search method. When the proof of unsatisfiability is aborted without success, the search method starts with the reduced Gröbner basis computed before. Firstly, this allows the search to take advantage of simplifications of the polynomials (e.g., variable elimination) which were identified while computing the Gröbner basis. Secondly, any variable assignment inferred by the unsatisfiability proof is applicable as well, thereby reducing the total number of free variables. This reduces the overall search depth and improves efficiency in many cases. However, the approach requires the search to find a valuation that reduces *all* polynomials in the Gröbner base to zero, including polynomials over slack variables. The reduction is performed by applying the same transformation rules as in the first step of the proof. In case of Allen's interval algebra, equations frequently appear which cannot be handled by the system. In this particular case, it would have been beneficial not to use the reduced Gröbner basis as starting point for search, but to start with the initial set of inequalities. Further research is needed to identify a universal search method that matches the performance of system's core method of proving unsatisfiability.

In the testing phase of our implementation of Buchberger's algorithm we compared our method with an alternative algorithm for computing the Gröbner basis,

provided the algebra system CoCoA. It turns out that CoCoA can in many cases compute the Gröbner basis more quickly, but also that in many cases when our implementation aborts with time-out, CoCoA does not give a solution either. This indicates that computing Gröbner bases is a computationally very demanding task. However, the key problem is not computing the Gröbner basis but identifying non-existence of real roots. As the experiments suggest, the proposed rule-based approach for examining the reduced Gröbner basis are effective. Unfortunately, it appears infeasible to grasp the performance analytically.

10   Conclusion and Outlook

In this paper we have shown that algebraic reasoning is valuable for designing qualitative calculi as it offers a universal method for reasoning about information in real-valued domains. Algebraic methods can be used to compute or to verify qualitative operations and they can also be used to analyze properties of a given calculus or a single relation. These methods are calculus-independent and can also be used to determine relationships across different calculi. Our approach is applicable to any calculus over the real-valued domain which can be modeled using multivariate polynomials. Although it may generally be infeasible to ensure that this approach offers a sound and complete decision method, the developed system is a valuable tool for analyzing and designing qualitative calculi. Our method is sound and empirical tests show its effectiveness. By providing the tool as part of the SparQ toolbox calculi designer are immediately supported.

Algebraic geometry also opens up an interesting approach to qualitative reasoning. Compared to composition-based qualitative reasoning, algebraic reasoning is computationally much more demanding. However, for calculi for which composition-based reasoning is known to be too weak for deciding consistency and no practical decision methods are known (e.g., see Lücke, Mossakowski, & Wolter, 2008), algebraic methods can offer a valuable alternative. After all, the existential theory of the reals—which subsumes the approach presented here—is known to be decidable (Tarski, 1948; Renegar, 1992). The challenge is to turn this theoretical result into a practical method.

Besides deciding consistency of spatial constraints, an algebraic approach enables us to tackle the task of computing valuations for qualitative constraint problems. Valuations instantiate spatial variables with concrete objects from the spatial domain in coherence with the given constraints. Thus, valuations are the basis of any visualization and the ability to compute valuations is essential for applications that need to graphically present qualitative spatial knowledge to a human user. Algebraic reasoning for visualization can be performed using the algebraic specification of qualitative constraints as developed in this article.

Acknowledgment

References

Aiello, M., Benthem, J. van, & Bezhanishvili, G. (2003). Reasoning about space: The modal way. *Journal of Logic and Computation*, *13*(6), 889–919.

Allen, J. F. (1983, November). Maintaining knowledge about temporal intervals. *Communications of the ACM*, 832–843.

Basu, S., Pollack, R., & Roy, M.-F. (2006). *Algorithms in real algebraic geometry* (2nd ed.). Springer.

Becker, T., & Weisspfennig, V. (1998). *Gröbner bases*. Springer.

Bennett, B. (1994). Some observations and puzzles about composing spatial and temporal relations. In *Proceedings of the workshop on spatial and temporal reasoning at ECAI94*.

Benthem, J. van, & Bezhanishvili, G. (2007). Modal logics of space. In M. Aiello, I. E. Pratt-Hartmann, & J. F. van Benthem (Eds.), *Handbook of spatial logics*. Springer.

Buchberger, B. (1985). Göbner-bases: An algorithmic method in polynomial ideal theory. In N. Bose (Ed.), *Multidimensional systems theory - progress, directions and open problems in multidimensional systems* (pp. 184–232). Reidel Publishing Company, Dordrecht - Boston - Lancaster.

CoCoATeam. (n.d.). CoCoA*: a system for doing Computations in Commutative Algebra*. Available at `http://cocoa.dima.unige.it`.

Cohn, A. G., & Hazarika, S. M. (2001). Qualitative spatial representation and reasoning: an overview. *Fundamenta Informaticae*, *46*, 1-29.

Collins, G. E. (1975). Quantifier elimination for real closed fields and cylindric algebraic decomposition. In *Automata theory and formal languages, 2nd GI conference Kaiserslautern* (Vol. 33, pp. 134–183). Springer.

Condotta, J.-F., Ligozat, G., & Saade, M. (2006). A generic toolkit for n-ary qualitative temporal and spatial calculi. In *Proceedings of the 13th international symposium on temporal representation and reasoning (TIME'06)* (pp. 78 – 86). Budapest, Hungary.

Cox, D. A., Little, J. B., & O'Shea, D. (1998). *Using algebraic geometry* (2nd ed., Vol. 185). Springer.

Cox, D. A., Little, J. B., & O'Shea, D. (2007). *Ideals, varieties, and algorithms* (3rd ed.). Springer.

Faugère, J.-C. (1999). A new efficient algorithm for computing Gröbner bases F_4. *Journal of Pure and Applied Algebra*, *139*(1), 61–88.

Faugère, J.-C. (2002). A new efficient algorithm for computing Gröbner bases without reduction to zero (F5). In *Proceedings of the International Symposium on Symbolic and Algebraic Computation (ISSAC)* (pp. 75–83). ACM Press.

Frank, A. (1991). Qualitative spatial reasoning about cardinal directions. In *Proceedings of the american congress on surveying and mapping (ACSM-ASPRS)* (pp. 148–167). Baltimore, Maryland, USA.

Freksa, C. (1992). Using orientation information for qualitative spatial reasoning. In *Theories and methods of spatio-temporal reasoning in geographic space. international conference GIS — from space to territory* (pp. 162–178). Springer.

Freksa, C. (2004). Spatial cognition—an AI perspective. In R. L. de Mántaras & L. Saitta (Eds.), *Proceedings of the 16th Eureopean Conference on Artificial Intelligence (ECAI'2004).* IOS Press. (invited paper)

Gerivini, A., & Renz, J. (2002). Combining topological and size information for spatial reasoning. *Artificial Intelligence*, *137*(1–2), 1–42.

Giovini, A., Mora, T., Niesi, G., Robbiano, L., & Traverso, C. (1991). "One sugar cube, please" or selection strategies in the buchberger algorithm. In *Proceedings of the 1991 international symposium on symbolic and algebraic computation (ISSAC '91)* (pp. 49–54). New York, NY, USA: ACM Press.

Kapur, D. (1998). Automated geometric reasoning: Dixon resultants, Gröbner bases, and characteristic sets. In D. Wang, R. Caferra, L. F. del Cerro, & H. Shi (Eds.), *Automated deduction in geometry* (Vol. 1360). Springer.

Ligozat, G. (1993). Qualitative triangulation for spatial reasoning. In A. U. Frank & I. Campari (Eds.), *Spatial information theory: A theoretical basis for GIS (COSIT'93)* (Vol. 716, p. 54-68). Springer.

Ligozat, G., & Renz, J. (2004). What is a qualitative calculus? A general framework. In C. Zhang, H. W. Guesgen, & W. K. Yeap (Eds.), *PRICAI 2004: Trends in Artificial Intelligence—8th Pacific Rim International Conference on Artificial Intelligence* (pp. 53–64). Springer.

Lücke, D., Mossakowski, T., & Wolter, D. (2008). Qualitative reasoning about convex relations. In C. Freksa, S. N. Newcombe, P. Gärdenfors, & S. Wölfl (Eds.), *Spatial Cognition VI* (Vol. 5248, pp. 426–440). Springer.

Moratz, R. (2006, August). Representing relative direction as binary relation of oriented points. In *Proceedings of the 17th European Conference on Artificial Intelligence (ECAI 2006).* Riva del Garda, Italy.

Moratz, R., Renz, J., & Wolter, D. (2000). Qualitative spatial reasoning about line segments. In W. Horn (Ed.), *Ecai 2000 proceedings of the 14th european*

conference on artificial intelligence (pp. 234–238). IOS Press, Amsterdam.

Pujari, A. K., & Sattar, A. (1999). A new framework for reasoning about points, intervals and durations. In *In proceedings of twenty-first international joint conference on Artificial Intelligence (IJCAI-09)* (pp. 1259–1267).

Randell, D. A., Cohn, A. G., & Cui, Z. (1992). Computing transitivity tables: A challenge for automated theorem provers. In *Automated deduction—CADE-11* (Vol. 607). Springer.

Randell, D. A., Cui, Z., & Cohn, A. G. (1992). A spatial logic based on regions and "Connection". In B. Nebel, C. Rich, & W. Swartout (Eds.), *KR'92, principles of knowledge representation and reasoning: Proceedings of the third international conference* (pp. 165–176). San Mateo (CA), USA: Morgan Kaufmann.

Renegar, J. (1992). On the computational complexity and geometry of the first-order theory of the reals, parts I–III. *Journal of Symbolic Computation*, *13*(3), 255–352.

Renz, J. (2007). Qualitative spatial and temporal reasoning: Efficient algorithms for everyone. In *Proceedings of the 20th International Joint Conference on Artificial Intelligence (IJCAI-07)* (pp. 526–531).

Renz, J., & Li, J. J. (2008). Automated complexity proofs for qualitative spatial and temporal calculi. In G. Brewka & J. Lang (Eds.), *Principles of knowledge representation and reasoning, proceedings of the eleventh international conference (KR-08)* (pp. 715–723). AAAI Press.

Renz, J., & Ligozat, G. (2005). Weak composition for qualitative spatial and temporal reasoning. In *Proceedings of the eleventh international conference on principles and practice of constraint programming* (p. 534-548).

Renz, J., & Mitra, D. (2004, August). Qualitative direction calculi with arbitrary granularity. In C. Zhang, H. W. Guesgen, & W. K. Yeap (Eds.), *8th Pacific Rim International Conference on Artificial Intelligence (PRICAI'04)* (Vol. LNAI 3157 / 2004, p. 65-74). Auckland, New Zealand.

Renz, J., & Nebel, B. (2007). Qualitative spatial reasoning using constraint calculi. In M. Aiello, I. E. Pratt-Hartmann, & J. F. van Benthem (Eds.), *Handbook of spatial logics* (pp. 161–215). Springer.

Tarski, A. (1948). *A decision method for elementary algebra and geometry*. Manuscript. Santa Monica, CA: RAND Corp.. (Republished as A Decision Method for Elementary Algebra and Geometry, 2nd ed. Berkeley, CA: University of California Press, 1951)

Vilain, M., Kautz, H., & Beek, P. van. (1989). Constraint propagation algorithms for temporal reasoning: a revised report. In *Readings in qualitative reasoning about physical systems* (p. 373 - 381). San Francisco, CA, USA: Morgan Kaufmann Publishers Inc.

Wen-tsün, W. (1978). On the decision problem and the mechanization of theorem proving in elementary geometry. *Scientia Sinica*, *21*, 157–179.

basis entity	algebraic representation	variable representation of A, B, C
1d-point	x	`ax; bx; cx`
interval	a, b	`a1, a2; b1,b2`
2d-point	x, y	`ax, ay; bx, by; cx, cy`
oriented-point	x, y, dx, dy	`ax, ay, adx, ady;`
		`bx,by, bdx, bdy;`
		`cx,cy,cdx,cdy`
dipole	sx, sy, ex, ey	`sax, say, eax, eay`

Table 4: Basis entities and their algebraic representation

Wölfl, S., & Mossakowski, T. (2005). Casl specifications of qualitative calculi. In A. G. Cohn & D. M. Mark (Eds.), *Conference on spatial information theory* (Vol. 3693, p. 200-217). Springer; Berlin.

Wölfl, S., Mossakowski, T., & Schröder, L. (2007). Qualitative constraint calculi: Heterogeneous verification of composition tables. In D. Wilson & G. Sutcliffe (Eds.), *20th International FLAIRS Conference (FLAIRS-20)* (pp. 665–670). AAAI Press.

Wolter, D., & Moshagen, L. (2008). Algebraic methods for analyzing qualitative spatio-temporal calculi. In *Proceedings of ECAI-workshop on spatial and temporal reasoning.*

A Algebraic Specifications in SparQ

For an algebraic calculus specification all base relations need to be specified by systems of polynomial equations over a real-valued domain $\mathbb{R}^n$. In a first step, the base objects of a qualitative calculus need to be represented algebraically. Objects involved in a constrained network (e.g., represented by variables A, B, C, etc.) are represented by tuples of real-valued variables. The base entities currently supported by SparQ are summarized in Table 4. The variable representations given in the table name the variables to denote algebraic specifications. For example, a binary relation between two 1d-points would involve the variables `ax` and `bx`.

In case of representing oriented points, the variables dx, dy give a direction vector. By contrast, specification of a dipole is based on the start point sx, sy and the end point ex, ey. The variable representation is used when specifying qualitative relations. To specify a qualitative relation $r(A, B)$, a set of multivariate

```
(def-calculus "1D Point Calculus (PC)"
          ⋮
     :base-relations (< = >)
          ⋮
     :algebraic-specification
          ((< ((1 ((ax 1))) < (1 ((bx 1)))))
           (= ((1 ((ax 1))) = (1 ((bx 1)))))
           (> ((1 ((ax 1))) > (1 ((bx 1))))))
          ⋮
)
```

Figure 4.: Excerpt from the definition of the point calculus

polynomial equations need to be provided such that

$$p_{r,1}(x_1, x_2, \ldots, x_k) \quad \Psi_1 \quad 0$$
$$\vdots$$
$$p_{r,k}(x_1, x_2, \ldots, x_k) \quad \Psi_k \quad 0$$

with $\Psi_i \in \{<, \leq, =, \geq, >\}$ holds if and only if $r(A, B)$ holds; $x_1, \ldots x_k$ stand for the variable representation introduced above.

In calculus specifications for SparQ polynomials are denoted in a simple LISP-like syntax, for example:
`(0 = ((1 ((ax 1))) (-1 ((bx 1)))))` stands for $0 = 1 \cdot ax^1 - 1 \cdot bx^1$.

The complete algebraic specification needs to written in the form

$$\{\text{'(' relation \{polynomial-equation\}}^\star \text{ ')'}\}^\star$$

For convenience, SparQ allows polynomials to be defined for different relations at once by using an appropriate disjunctive, general relation in the specification. In Figure 4 the algebraic specification of the point calculus with its base relations $<, =, >$ is given as an example.

B Algebraic Reasoning with SparQ

The algebraic reasoning tool presented in this article is now part of the spatial reasoning toolbox SparQ. Four tools are offered:

1. Basic algebraic operations

2. Checking a qualitative constraint network for satisfiability

3. Analyzing an operation of a qualitative calculus

4. Determining qualitative relations from quantitative measures

In the following we briefly show how these tools can be used—for further details please consult the SparQ manual. In this reference, `<calculus>` is a calculus identifier (e.g., `allen` to designate Allen's interval algebra, `pc` to designate the point calculus) and `<network>` stands for a qualitative constraint network.

B.1   Algebraic Operations

SparQ contains basic tools for determining the polynomial modeling of qualitative constraint networks, and for algebraic tests. The tools aim at achieving interoperability with algebraic toolkits as well as at posing specialized queries.

B.1.1   Polynomial Modeling

As a basic service the polynomial equations modeling a given qualitative scenario can be obtained—this tool is useful for interaction with computer algebra tools, e.g., CoCoA.

```
sparq a-reasoning <calculus> polynomials <format>
<network>
```

The parameter `<format>` controls whether the output is printed in text, LaTeX, CoCoA, or LISP notation.
Example:[4]

```
sparq a-reasoning pc polynomials text "((a < b))"
```

This yields the following output

```
;; Variable assignment:
;; constraint variable -> object specifiers -> variables
;; A                    -> (A[AX] )          -> (x[1] )
;; B                    -> (B[AX] )          -> (x[2] )
;;
;; Slack variables with side condition x_i /= 0: x[20]
;;
;; [1:] x[1] - x[2] + x[20]^2
```

---

[4]The quotation marks in the following examples are only necessary to prevent a Unix shell from interpreting parentheses

B.1.2 Testing Ideal Emptiness

SparQ's core method for testing emptiness of the ideal generating by a set of polynomials is accessible with the command

```
sparq a-reasoning <calculus> satisfiability <slacks>
<polynomials>
```

which takes as input a list of slack variables and the set of polynomial equations as returned by the polynomial modeling command using output option "lisp". In continuation of the above example we have

```
sparq a-reasoning pc satisfiability "(20)" "(((1 ((1 1)))
(-1 ((2 1))) (1 ((20 2)))))"
```

which yields the output "IS SATISFIABLE." to signal that a real-valued solutions exists.

B.2 Consistency Checking

Consistency of a qualitative constraint network of base relations can be checked algebraically using

```
sparq a-reasoning <calculus> consistency <network>
```

Possible results:

`SATISFIABLE` The network is proven to be satisfiable.

`NOT SATISFIABLE.` The network is proven to be unsatisfiable.

`CANNOT DECIDE.` Neither one of the above proofs succeeded.

Example:

```
sparq -v a-reasoning allen consistency "((x b y) (y b z)
(x o z))"
```

The test whether three constraints x **b**efore y, y **b**efore z, and x **o**verlapping with z are jointly satisfiable returns "NOT SATISFIABLE." By turning on SparQ's verbose mode (command line option "-v"), the complete proof is printed.

B.3   Verifying Calculus Operations

By a single command the complete verification of an operation table such as, e.g., composition, can be invoked. SparQ generates all constraint networks covered by the operation and tests their satisfiability (in the case of composition all triples of relations r_1, r_2, r_3 are tried for `((A` r_1 `B)  (B` r_2 `C)  (A` r_3 `C))`). The results of the satisfiability test are compared to the entries listed in the operation table of the calculus.

```
sparq a-reasoning <calculus> analyze-operation <operation>
```

B.4   Qualification

Algebraic calculus specifications can also be used for an automatic interpretation of measured values to qualitative relations.

```
sparq a-reasoning <calculus> qualify <option> <scenario>
```

Example:

```
sparq a-reasoning pc qualify all "((a 1) (b 2/3) (c 4))"
```

SparQ determines `all` relations between any pair of objects which yields the constraint network `((A > B)  (A < C)  (B < C))`.