Preprints of the Federated Conference on
Computer Science and Information Systems pp. 55–63

Encoding Relative Orientation and Mereotopology Relations with Geometric Constraints in CLP(QS)

Carl Schultz
Institute for Geoinformatics
University of Münster, Germany
Email: schultzc@uni-muenster.de

Mehul Bhatt
Department of Computer Science
University of Bremen, Germany
Email: bhatt@informatik.uni-bremen.de

***Abstract*—One approach for encoding the semantics of spatial relations within a declarative programming framework is by systems of polynomial constraints. However, solving such constraints is computationally intractable in general (i.e. the theory of real-closed fields), and thus far the proposed symbolic algebraic methods do not scale to real world problems. In this paper we address this intractability by investigating the use of constructive geometric constraint solving (in combination with numerical optimisation) within the context of constraint logic programming over qualitative spaces, CLP(QS). We present novel geometric encodings for relative orientation and mereotopology relations and show that our encodings are significantly more robust than other proposed approaches for directly encoding inequalities, due to our encodings being based on a standard, well known set of relations encoded as quadratic equations. Our encodings are general (i.e. not implementation specific) and can thus be directly employed in *all* standard geometric constraint solvers, particularly solvers that are prominent within the Computer Aided Design and Manufacturing communities. We empirically evaluate our approach on a range of benchmark problems from the Qualitative Spatial Reasoning community, and show that our method outperforms other symbolic algebraic approaches to spatial reasoning by orders of magnitude on those benchmark problems (such as Cylindrical Algebraic Decomposition).**

I. Introduction

Many complex, real world problems can be elegantly expressed in a declarative manner within the paradigm of logic programming. In particular, the user specifies *what* needs to be accomplished, rather than procedurally prescribing *how* the problem should be solved. Often such real world problems inherently involve spatial aspects: variables ranging over spatial object domains (polygons, circles, points, etc.) and spatial relations between those objects.

Qualitative spatial representation and reasoning has received considerable attention, especially from the viewpoint of the research communities of spatial information theory, and knowledge representation and reasoning. Knowledge representation and reasoning about *space* may be formally interpreted within diverse frameworks such as: (a) geometric reasoning & constructive (solid) geometry [20]; (b) relational algebraic semantics of 'qualitative spatial calculi' [25]; and (c) by axiomatically constructed formal systems of mereotopology and mereogeometry [1]. Independent of formal semantics, commonsense spatio-linguistic abstractions offer a human-centred and cognitively adequate mechanism for logic-based automated reasoning about spatio-temporal information [4].

However, what is clearly still lacking is a systematic knowledge representation (KR) centred methodological foundation that provides a computational backbone —*formal semantics in the context of mainstream KR methods, declarative (spatial) programming paradigm, general toolsets* – for commonsense reasoning about space. This is essential in order to provide a developmental basis and seamless, systematic integration within large-scale AI, and hybrid intelligent systems that involve diverse types of knowledge representation, reasoning, and learning modules (e.g., IBM Watson).

Our long-term research agenda is to integrate spatial reasoning natively within declarative, logic-based frameworks [3]. We have partially realised aspects of this in the Constraint Logic Programming over Qualitative Spatial domains system: CLP(QS) [3]. Users can specify logic programs that are solved with a seamless combination of spatial reasoning and Prolog's standard variable unification, i.e. logic programming extended to natively handle qualitative and geometric spatial constraints. Specifically, we utilise the following features of logic programming:

- *declarativeness* - a Prolog program is a specification of the conditions that need to be satisfied, or the goals that must be accomplished, rather than a procedural set of instructions; that is, the user specifies *what* the program should accomplish, rather than *how* the problem at hand should be solved
- *recursive variable generation* - languages such as Prolog are able to infer the existence of objects that are required to solve the problem that were not explicitly specified by the user; combined with recursion, this enables a user to explore the unboundedness and density of objects in space

Relation algebraic qualitative spatial reasoning (e.g. the left-right calculus LR [26]), while efficient, is incomplete in general [21, 22, 25].[1] Alternatively, constraint logic programming based systems such as CLP(QS) [3] and others (see

[1] *Incompleteness* refers to the inability of a spatial reasoning method to determine, in general, whether a given set of qualitative spatial constraints is consistent or inconsistent. Relation-algebraic spatial reasoning (i.e. using algebraic closure based on weak composition) has been shown to be incomplete for a number of spatial languages and cannot guarantee *consistency* in general, e.g. relative directions [22] and containment relations between linearly ordered intervals [21], Theorem 5.9.

[6, 7, 28, 29]) adopt an analytic geometry approach where spatial relations are encoded as systems of polynomial equation and inequality constraints. Thus, the task of determining whether a set of spatial constraints is consistent becomes equivalent to determining the satisfiability of a system of polynomial constraints with variables ranging over reals.[2]

We have investigated a range of polynomial constraint solving methods including CLP(R) for linear constraints, and SAT Modulo Theories and quantifier elimination by Cylindrical Algebraic Decomposition (CAD) for general systems of non-linear constraints [3, 31, 32]. However, solving such constraints is computationally intractable in general (i.e. the theory of real-closed fields; for example, the CAD algorithm has double exponential complexity, $O(a^{b^n})$, in the number of variables in the polynomial constraints, n) [2], and thus far, prominent symbolic algebraic methods do not scale to real world problems [24]. In this paper we address this intractability by investigating the use of constructive geometric constraint solving for spatial reasoning within the context of CLP(QS).

The paper is structured as follows. In the remainder of introduction we highlight some key benefits of the utilisation of geometric constraint solving within constraint logic programming over qualitative spaces. Section II provides an introduction to spatial reasoning by polynomial encodings, and formally specifies the standard geometric constraint language. In Section III we empirically evaluate our approach on a range of well known benchmark problems from the Qualitative Spatial Reasoning community, and show that our system outperforms other symbolic algebraic approaches to spatial reasoning by orders of magnitude. Section IV evaluates our encodings using a range of spatial benchmark problems. We then present our conclusions about the main contributions of the paper and directions for future research.

A. Motivations for utilising geometric constraint solving

Geometric constraint solving for higher-level qualitative spatial reasoning has a range of benefits: problems can have a combination of numerical and qualitative information, the methods scale to real world sized problems (in the order of hundreds of objects), and they can produce consistent and "best found" instantiations for both solved and unsolved problems, respectively. Moreover, at each iteration, the solver produces increasingly better configurations as it attempts to converge on the solution; we can directly study and visualise these intermediate configurations as a spatial *history* of configurations [17, 18] giving further insight into the nature of the problem at hand. However, a key limitation is that these methods do not handle inequalities well [16], which are used for encoding a large range of qualitative spatial relations, including relative orientation and mereotopology.

[2]Tarski famously proved that the theory of real-closed fields is decidable via quantifier elimination (see [2, 12, 13] for an overview and algorithms); i.e. in a finite amount of time we can determine the consistency (or inconsistency) of any formula consisting of quantifiers ($\forall$, $\exists$) over the reals, and polynomial equations and inequalities combined using logical connectors ($\wedge$, $\vee$, $\neg$). Thus, by encoding spatial relations as systems of polynomial constraints (i.e. analytic geometry) we can employ polynomial constraint solving methods that are guaranteed to determine (in)consistency, giving us sound and complete spatial reasoning.

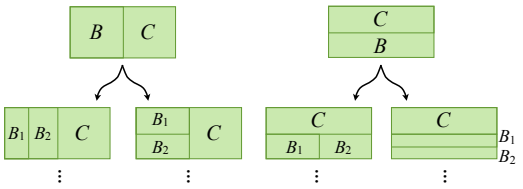


Fig. 1. Recursively enumerating the ways that a rectangle A can be partitioned into a union of rectangles, $\text{union}(B, C)$.

Building on results within the geometric constraint community, we present novel geometric encodings for relative orientation and topology relations using only equations (i.e. corresponding to robust ruler and compass construction methods [27], and thus avoid problematic *saddles* near local optima that are common with inequality encodings using numerical optimisation methods [16]. Moreover, as our encodings rely on a standard set of geometric constraints, they can be used to enhance *all* standard geometric constraint solvers with a variety of qualitative spatial relations that are prominent within the Computer Aided Design and Manufacturing communities.

B. An example of Spatial Reasoning in CLP(QS)

Using recursive variable generation we can declaratively explore the ways that a rectangle can be partitioned into a set of rectangles (employing CLP(QS) mereology relations between boolean combinations of rectangles) [32]:

```
partition(A, union(B,C)) :-
  mereology(equal, A, union(Br,Cr)),
  mereology(discrete_from, Br, Cr),
  ((B = Br, C = Cr)
   ;
   partition(Br, B),
   partition(Cr, C)
  ).

?- partition(rectangle(_,_,_), union(B,C)).
```

Importantly, the initial specification of the partitioning task is completely *qualitative* and does not contain any numerical information. We can continually request further solutions from Prolog to generate new solutions. Each solution is defined by a set of qualitative constraints between rectangles and thus represents an infinite set of rectangle configurations. For example, in Figure 1 the left solution in the first row is defined by the constraints: $A_w = B_w + C_w$, $A_h = B_h = C_h$, $A_x = B_x$, $B_x < C_x$ where A_w, A_h is the width and height of A, and (A_x, A_y) is the coordinate of the bottom left corner of A. For each qualitatively distinct solution we can then generate a concrete, numerically defined instantiation of the objects using geometric constraint solving.

We can further constrain the problem to enumerate the ways that a *square* can be partitioned into n squares, $n > 1$. As illustrated in Figure 2, CLP(QS) determines that solutions exist for $n = 4, 6, 7, 8, 9$ and no solutions exist for $n = 2, 3, 5$.

Fig. 2. Square partitioning solutions for $n = 4, 6, 7, 8, 9$.

```
all_squares(square(_,_)).
all_squares(union(A,B)) :-
  all_squares(A), all_squares(B).

?- A=square(_,_),
   partition(A, union(B,C)),
   all_squares(union(B,C)).
```

II. Spatial Reasoning by Polynomial Encodings

Analytic geometry methods parameterise classes of objects and encode spatial relations as systems of polynomial equations and inequalities [10]. For example, we can define a sphere as having a 3D centroid point (x, y, z) and a radius r, where x, y, z, r are reals. Two spheres s_1, s_2 *externally connect* or *touch* if

$$(x_{s_1} - x_{s_2})^2 + (y_{s_1} - y_{s_2})^2 + (z_{s_1} - z_{s_2})^2 = (r_{s_1} + r_{s_2})^2 \quad (1)$$

If the system of polynomial constraints is satisfiable then the spatial constraints are consistent. Specifically, the system of polynomial (in)equalities over variables X is satisfiable if there exists a real number assignment for each $x \in X$ such that the (in)equalities are *true*. Partial geometric information (i.e. a combination of numerical and qualitative spatial information) is utilised by assigning the given real numerical values to the corresponding object parameters.

A. Constructive Geometric Constraint Solving

A plethora of methods have been developed for geometric constraint solving via solving systems of polynomial constraints, and can be broadly categorised as: numerical optimisation (e.g. [16]), symbolic methods (e.g. [10, 14]), and constructive (synthetic) methods (e.g. [8, 15, 27]). We focus on graph-based constructive methods due to their practical efficiency and popularity in industry (e.g. Autodesk Inventor,[3] LEDAS LGS2D,[4] FreeCAD[5]), although our encodings can be similarly applied to any of the aforementioned geometric constraint solving approaches.

In a seminal paper, Owen [27] identifies a particular set of spatial relations that, on one hand, are useful for a wide range of applications, particularly engineering and computer aided manufacturing, and on the other hand, can be reasoned about efficiently enough to address real-world scale problems. The particular set of relations correspond to distance, incidence, and angle constraints that can be encoded as quadratic equations over 2D points, lines, and circles. Geometrically, these correspond to relations that can be constructed using the familiar idealised ruler and compass from Euclid's *Elements* [19]. We refer to this restricted set of spatial relations as the *standard geometric constraint language*. This set of relations is now standard within the geometric constraint solving community, and all prominent, industry-standard constraint solvers that we are aware of adopt precisely this set of relations (although sometimes such systems also support ellipses), particularly within Computer Aided Design and Manufacturing.

Owen [27] also presents an influential graph-based reduction method for constructive geometric constraint solving. Spatial information is represented as a graph, where nodes are variables that range over a spatial domain of geometries, and edges represent spatial constraints. Firstly, the graph is analysed, and then a sequence of construction steps is determined that produces a configuration of objects that satisfies the (declarative) geometric constraints.

We emphasise that, as our encodings are based on this standard geometric constraint language, they can be utilised within *all* prominent geometric constraint solvers that also adopt this language. That is, our encodings are not solver-implementation specific.

B. The Standard Geometric Constraint Language

The spatial domains of objects in the standard geometric constraint language are points **P**, lines **L**, and circles **C**:

- a point $p \in \mathbf{P}$ is a pair of reals, $(x_p, y_p) \in \mathbb{R}^2$
- a line $l_{ab} \in \mathbf{L}$ is a pair of distinct points, $a, b \in \mathbf{P}$, $a \neq b$
- a circle $C_i \in \mathbf{C}$ is a circle with centre point $p_i \in \mathbf{P}$ and radius $r_i \in \mathbb{R}$, $0 < r_i$

We use lower case letters to refer to points. We use $l_{p_1 p_2}$ to refer to lines between points in the subscript. We use upper case C_i with a subscript number (if needed) to refer to circles. For brevity, if two points a, b have been declared, then we can refer to the line l_{ab} without explicitly quantifying l, and doing so implicitly constrains a, b to be distinct, e.g. let φ be a predicate, then:

$$\exists a, b \in \mathbf{P}, \exists l_{ab} \in \mathbf{L} \left(\varphi(l_{ab})\right) \equiv \exists a, b \in \mathbf{P} \left(\varphi(l_{ab})\right).$$

Table I presents polynomial encodings for the standard set of geometric constraints between points, lines, circles. They correspond to:

- *incidence* between points-lines, and points-circles (collinear, coincident);
- *orientation* between lines (parallel, perpendicular);
- *constant distance and angles* for lines and circles (distance between points, radii of circles, angle between points a, b about a reference point p).

III. Encoding Qualitative Spatial Relations using Geometric Constraint Languages

In this section we present a range of novel encodings that enable us to reason about qualitative spatial relations over extended *regions* (in particular, relative orientation and mereotopology over regions) using traditional geometric constraint solving methods that are restricted to the standard geometric constraint language.

[3] www.autodesk.com/products/inventor/overview
[4] ledas.com/products/lgs2d/
[5] www.freecadweb.org/

Relation	Polynomial Encoding
collinear (COLL) (point p, line l_{ab})	$(x_b - x_a)(y_p - y_a) = (x_b - y_a)(x_p - x_a)$
coincident (COIN) (point a, circle C_i)	$(x_{p_i} - x_a)^2 + (y_{p_i} - y_a)^2 = r_i^2$
coincident (COIN) (line l_{ab}, circle C_i)	$\text{COIN}_{a,C_i} \wedge \text{COIN}_{b,C_i}$
perpendicular (PERP) (lines l_{ab}, l_{cd})	$(y_b - y_a)(y_d - y_c) = -(x_b - x_a)(x_d - x_c)$
parallel (PARA) (lines l_{ab}, l_{cd})	$(y_b - y_a)(x_d - x_c) = (y_d - y_c)(x_b - x_a)$
vertical (VERT) (line l_{ab})	$x_a = x_b$
length (LEN) (line l_{ab}, constant k)	$(x_a - x_b)^2 + (y_a - y_b)^2 = k^2$
angle (ANG) (points a, b, p, constant θ)	$\theta = \text{atan2}((y_b - y_p), (x_b - x_p)) - \text{atan2}((y_a - y_p), (x_a - x_p))$

TABLE I
POLYNOMIAL ENCODINGS OF GEOMETRIC CONSTRAINT RELATIONS.

A. Minimum distance

A minimum distance circle MDIST_C is a circle C with a diameter at least ε. This encoding provides a means to implement a minimum bound on the diameter of a circle. The constrained circle can then be used to enforce minimum distances between spatial objects. Many prominent geometric constraint solving systems (including Inventor and FreeCAD) do not support this constraint.

As illustrated in Figure 3, the encoding adds a fixed-length chord to the circle (i.e. a line where the endpoints are coincident with the circle). The length of the chord determines the minimum permitted diameter of the circle. The circle diameter is minimised when the chord intersects the centroid of the circle. The circle diameter has no upper bound, that is, the chord can be positioned an arbitrary distance from the centroid; the circle diameter must then increase in order to maintain the constraint that the chord endpoints are coincident with the circle. Additionally, we impose a vertical constraint on the chord to improve the solving efficiency of the encoding (i.e. the effect of the vertical constraint is to eliminate one x variable from the chord).

$$\text{MDIST}(C) \equiv \exists l_{ab} \in \mathbf{L} \left(\text{VERT}(l_{ab}) \wedge \text{COIN}(l_{ab}, C) \wedge \text{LEN}(l_{ab}, \varepsilon) \right)$$

Various numerical optimisation methods do provide *box constraints*, i.e. constant bounds on variables, e.g. limited BFGS-B [9]. In such systems this encoding is redundant, as we can enforce a box constraint on the radius of the circle.

B. Point-segment coincidence

While the *collinear* constraint between points and lines is common in geometric constraint systems (i.e. a point lies anywhere on an infinite line), the ability to constrain a point to lie coincident on a line *segment* (i.e. between two points) is typically not supported. The following encoding realises a *coincidence* constraint between a point p and a segment l_{ab}.

As illustrated in Figure 4, firstly, the encoding adds a circle C_1 such that the two endpoints of the given line segment l_{ab} are made coincident to C_1, and the centroid of the circle is made collinear with l_{ab}; in Section III-D we formalise this useful construction between a circle and line, and refer to it as a *brace* relation. Importantly, due to the brace relation, the length of l_{ab} is equal to the diameter of the circle. Next, the encoding adds a line l_{cd} with endpoints coincident to C_1 and perpendicular to l_{ab}. The perpendicular constraint ensures that the two lines always intersect within the interior of C_1. Finally, as the non-parallel lines necessarily intersect at a single point, the given point p is constrained to be collinear to both lines, and is thus always constrained to lie on the segment l_{ab}.

$$\begin{aligned}\text{COIN}(p,\ l_{ab}) \equiv \\ \exists C_1 \in \mathbf{C} \big(\text{COIN}(l_{ab}, C_1) \wedge \text{COLL}(p_1, l_{ab}) \wedge \\ \exists l_{cd} \in \mathbf{L} \big(\text{COIN}(l_{cd}, C_1) \wedge \text{PERP}(l_{ab}, l_{cd}) \big) \wedge \\ \text{COLL}(p, l_{ab}) \wedge \text{COLL}(p, l_{cd}) \big)\end{aligned}$$

For convenience and brevity we also define the relation that segment l_{ab} is coincident with segment l_{cd} as: the endpoints a, b are coincident with the segment l_{cd}.

$$\text{COIN}(l_{ab},\ l_{cd}) \equiv \text{COIN}(a,\ l_{cd}) \wedge \text{COIN}(b,\ l_{cd})$$

Points p can not be equal to either endpoint a, b as the line l_{cd} can not have zero length. If we drop this line constraint for l_{cd} so that c, d can also be equal, then p can also equal the endpoints. These are useful predicates for defining topological relations in Section III-D, and thus we refer to them as: $\text{COIN}^{\subseteq}(p,\ l_{ab})$ and $\text{COIN}^{\subseteq}(l_{ab},\ l_{cd})$.

C. Relative Orientation

To the best of our knowledge, no geometric constraint solver is able to directly express qualitative orientation (*left*, *right*) between line segments and regions (nor points). This stems from the inability of standard solvers to robustly express inequalities. That is, one common polynomial encoding of

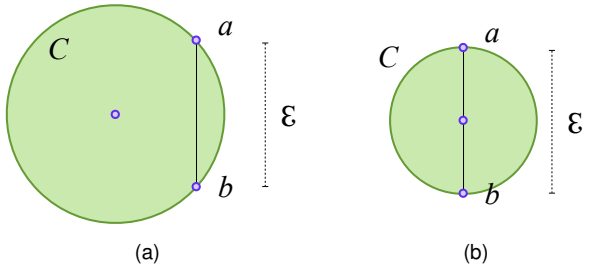


Fig. 3. Minimum distance circle C can not have diameter less than ε.

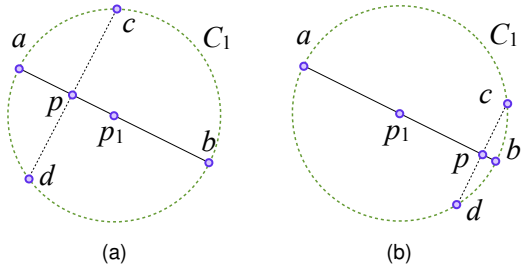


Fig. 4. Point p is constrained to lie on the segment between points a, b.

relative orientation is using the sign of the cross product to determine the orientation of a point with respect to a directed line segment, i.e. a point p is left of the segment a, b according to the following inequality:

$$p \textit{ left of } l_{ab} \equiv_{def} (x_b - x_a)(y_p - y_a) > (y_b - y_a)(x_p - x_a)$$

As reported by [16], a "trick" in numerical optimisation of introducing a new variable to express the inequality fails, as it produces a *saddle* near the optimum, thus making the problem significantly more difficult to solve. We have confirmed this result using the state-of-the-art numerical optimisation method: limited LBFGS-B [9].[6]

We have identified the following geometric encoding of relative orientation using the standard geometric constraint language; as such, it is supported, and robustly solved for, by all prominent geometric constraint solvers (Inventor, FreeCAD, LEDAS).

As illustrated in Figure 5, the encoding for the *left of* relation adds a new point c collinear to the given line l_{ab}, and adds a line l_{cp}, between the given point p and the new point c. The encoding then adds the constraint that the angle between l_{ab} and l_{cp} is 90^o counter-clockwise. The length of the line l_{cp} is unbounded, and thus p can be moved an arbitrary distance away from l_{ab}. The key is that, if p is moved to the right side of l_{ab}, then the angle constraint is violated, and thus p is forced to remain of the left side.

$$\begin{aligned}&\text{LEFT}(p,\ l_{ab}) \equiv \\ &\quad \exists c \in \mathbf{P}\left(\text{ANG}(b, p, c, \tfrac{\pi}{2}) \wedge \text{COLL}(c, l_{ab})\right)\end{aligned}$$

$$\begin{aligned}&\text{RIGHT}(p,\ l_{ab}) \equiv \\ &\quad \exists c \in \mathbf{P}\left(\text{ANG}(b, p, c, -\tfrac{\pi}{2}) \wedge \text{COLL}(c, l_{ab})\right)\end{aligned}$$

We extend this definition to relative orientation relations between lines and circles (Figure 6(a)).

$$\begin{aligned}&\text{LEFT}(C_1,\ l_{ab}) \equiv \\ &\quad \exists c, d \in \mathbf{P}\left(\text{ANG}(b, p_1, c, \tfrac{\pi}{2}) \wedge \text{COLL}(c, l_{ab}) \wedge \right. \\ &\qquad \left. \text{COIN}(d, C_1) \wedge \text{COIN}(d, l_{cp_1})\right)\end{aligned}$$

$$\begin{aligned}&\text{RIGHT}(C_1,\ l_{ab}) \equiv \\ &\quad \exists c, d \in \mathbf{P}\left(\text{ANG}(b, p_1, c, -\tfrac{\pi}{2}) \wedge \text{COLL}(c, l_{ab}) \wedge \right. \\ &\qquad \left. \text{COIN}(d, C_1) \wedge \text{COIN}(d, l_{cp_1})\right)\end{aligned}$$

As illustrated in Figure 6(b), alternative encodings for relative orientation exist that introduce fewer additional constraints and objects per orientation relation, and may also be solved in a more stable way depending on the implementation for the ANG geometric constraint. In the example illustrated in Figure 6(b), a circle C_1 is introduced and constrained to be *left of* the line a, b, and for each point c, d, e, a line is created parallel to l_{ab} with each endpoint c', d', e' coincident to C_1; this encoding aims to minimise the use of the ANG constraint when numerous points are constrained to be left of the same line. Identifying the best encodings for particular tasks is a topic of future work.

[6]Implementation available at: users.iems.northwestern.edu/∼nocedal/lbfgsb.html

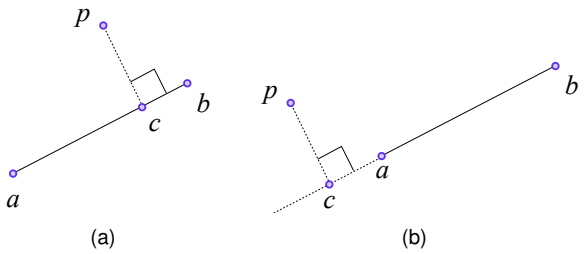
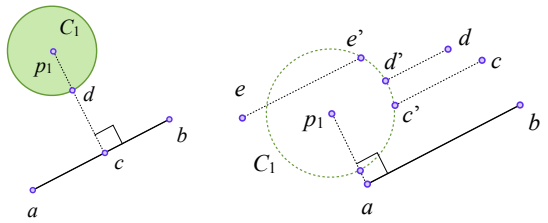


Fig. 5. Point p is constrained to lie anywhere to the *left of* line (a, b). The angle from point b to p about c is fixed at $\frac{\pi}{2}$ counter-clockwise. The distance between c, p is not constrained.



(a) Circle C_1 is constrained to lie anywhere to the left of line (a, b).

(b) Alternative, more robust encoding for numerous *left of* constraints. Points c, d, e are *left of* line (a, b).

Fig. 6. Relative orientation encodings for regions and more efficient encodings.

D. Topological relations between circles

In standard geometric constraint solvers there is no way of directly specifying mereotopological constraints between higher-level objects and regions such as circles, squares, triangles, polygons, and so on. In this section we present encodings for topological relations between circles, and then use these encodings as a basis for defining relations between more general regions. Firstly we define a useful BRACE relation between a line segment and a circle that ensures the diameter of the circle is equal to the length of the segment (Figure 7(a)).

$$\text{BRACE}(l_{ab}, C_i) \equiv \text{COIN}(l_{ab}, C_i) \wedge \text{COLL}(p_i, l_{ab})$$

We adopt the terminology of the prominent topological spatial logic, the *Region Connection Calculus* (RCC) [30]: *disconnects* (DC), *externally connects* (EC), *partial overlap* (PO), *tangential proper part* (TPP), *non-tangential proper part* (NTPP), *proper part* (PP), *part of* (P), *discrete from* (DR), *equal* (EQ). Note that EQ between two circles is trivially satisfied by constraining the centroids and radii to be equal.

The topological relation encodings are illustrated in Figure 7. To ensure circle intersection (e.g. TPP, NTPP, PO), the encodings constrain one or both endpoints of the brace segments of one circle to be coincident to the brace segment of the other circle; a pair of brace endpoints are made equal for boundary contact (e.g. TPP). EC is encoded with a point of boundary contact a that is coincident to a segment $l_{p_1 p_2}$ between the circle centroids. DC is encoded by introducing a

third circle C_3 so that one endpoint of each of the braces of C_1 and C_2 lie on different sides of the centroid of C_3, along the brace of C_3.

Observe that the brace segment within a circle can be rotated about the circle's centroid. Thus, considering NTPP for example, C_1 can occupy any circular region within C_2 by moving C_1 along the brace of C_2, and rotating the brace of C_2.

$$\begin{aligned}&\mathrm{TPP}(C_1, C_2) \equiv\\ &\quad \exists l_{ab}, l_{ac} \in \mathbf{L}\,\big(\mathrm{BRACE}(l_{ab}, C_2) \wedge\\ &\qquad\qquad \mathrm{BRACE}(l_{ac}, C_1) \wedge \mathrm{COIN}(c, l_{ab})\big)\end{aligned}$$

$$\begin{aligned}&\mathrm{NTPP}(C_1, C_2) \equiv\\ &\quad \exists l_{ab}, l_{cd} \in \mathbf{L}\,\big(\mathrm{BRACE}(l_{ab}, C_2) \wedge\\ &\qquad\qquad \mathrm{BRACE}(l_{cd}, C_1) \wedge \mathrm{COIN}(l_{cd}, l_{ab})\big)\end{aligned}$$

$$\begin{aligned}&\mathrm{PO}(C_1, C_2) \equiv\\ &\quad \exists l_{ab}, l_{cd} \in \mathbf{L}\,\big(\mathrm{BRACE}(l_{ab}, C_2) \wedge \mathrm{BRACE}(l_{cd}, C_1) \wedge\\ &\qquad\qquad \mathrm{COIN}(a, l_{cd}) \wedge \mathrm{COIN}(d, l_{ab})\big)\end{aligned}$$

$$\begin{aligned}&\mathrm{EC}(C_1, C_2) \equiv\\ &\quad \exists a \in \mathbf{P}\,\big(\mathrm{COIN}(a, l_{p_1p_2}) \wedge\\ &\qquad\qquad \mathrm{COIN}(a, C_1) \wedge \mathrm{COIN}(a, C_2)\big)\end{aligned}$$

$$\begin{aligned}&\mathrm{DC}(C_1, C_2) \equiv\\ &\quad \exists a, b \in \mathbf{P}, \exists C_3 \in \mathbf{C}\,\big(\mathrm{BRACE}(l_{p_1p_2}, C_3)\\ &\qquad\qquad \wedge \mathrm{COIN}(a, l_{p_1p_3}) \wedge \mathrm{COIN}(a, C_1)\big)\\ &\qquad\qquad \wedge \mathrm{COIN}(b, l_{p_2p_3}) \wedge \mathrm{COIN}(b, C_2)\big)\end{aligned}$$

We can drop the distinction between boundaries (i.e. corresponding to RCC5 and other RCC relations) by employing the modified coincident constraint between points and segments $\mathrm{COIN}^{\subseteq}(p,\ l_{ab})$, where a point p can also equal the segment endpoints l_{ab}. Thus, we encode the definitions that:

- PP is a disjunction of NTPP and TPP;
- P is a disjunction of PP and EQ;
- DR is a disjunction of DC and EC.

$$\begin{aligned}&\mathrm{PP}(C_1, C_2) \equiv\\ &\quad \exists l_{ab}, l_{cd} \in \mathbf{L}\,\big(\mathrm{BRACE}(l_{ab}, C_2) \wedge \mathrm{BRACE}(l_{cd}, C_1)\\ &\qquad\qquad \wedge \mathrm{COIN}^{\subseteq}(c, l_{ab}) \wedge \mathrm{COIN}(d, l_{ab})\big)\end{aligned}$$

$$\begin{aligned}&\mathrm{P}(C_1, C_2) \equiv\\ &\quad \exists l_{ab}, l_{cd} \in \mathbf{L}\,\big(\mathrm{BRACE}(l_{ab}, C_2) \wedge \mathrm{BRACE}(l_{cd}, C_1)\\ &\qquad\qquad \wedge \mathrm{COIN}^{\subseteq}(l_{cd}, l_{ab})\big)\end{aligned}$$

$$\begin{aligned}&\mathrm{DR}(C_1, C_2) \equiv\\ &\quad \exists a, b \in \mathbf{P}, \exists C_3 \in \mathbf{C}\,\big(\mathrm{BRACE}(l_{p_1p_2),C_3}\\ &\qquad\qquad \wedge \mathrm{COIN}^{\subseteq}(a, l_{p_1p_3}) \wedge \mathrm{COIN}(a, C_1)\big)\\ &\qquad\qquad \wedge \mathrm{COIN}^{\subseteq}(b, l_{p_2p_3}) \wedge \mathrm{COIN}(b, C_2)\big)\end{aligned}$$

E. Egg-yolk approach for defining relations between regions

We employ the egg-yolk method of modelling regions with indeterminante boundaries [11] to characterise a class of regions (including polygons) that satisfies topological and relative orientation relations. Each egg-yolk region is an equivalence class for all regions that are contained within the upper approximation (the *egg white*), and completely contain the lower approximation (the *egg yolk*). Let **R** be the domain of

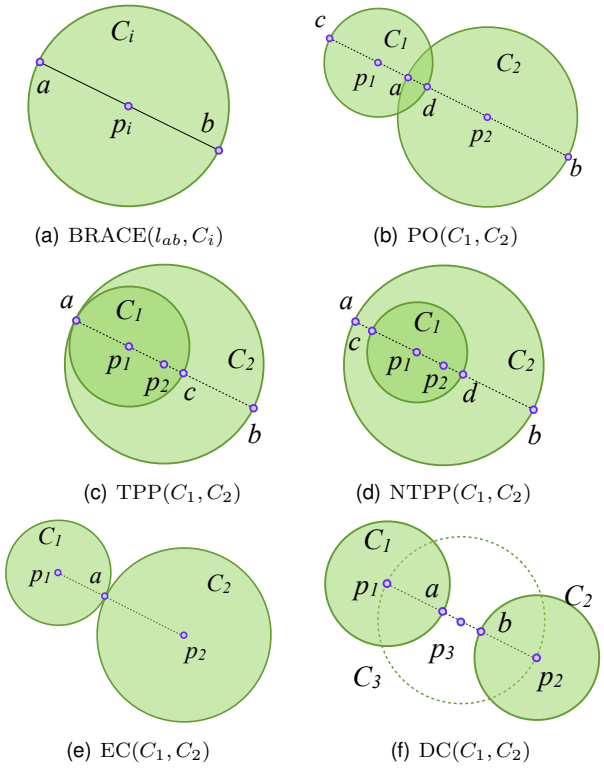


Fig. 7. Topological relations between circles.

egg-yolk regions, where egg-yolk region $R \in \mathbf{R}$ consists of a circular upper and a lower approximation $R^+, R^- \in \mathbf{C}$ such that $\mathrm{NTPP}(R^-, R^+)$ (see Figure 8(a)).

We can realise these regions through constructive geometric constraint encodings, giving us a method of generating arbitrary regions that satisfy qualitative spatial constraints. We declaratively define a (simple, non-self-intersecting) polygon as a sequence of vertices such that:

1) all vertices are contained within the upper approximation
2) no segment between adjacent vertices intersects the lower approximation
3) the (absolute) winding number about the centroid of the lower approximation is 1

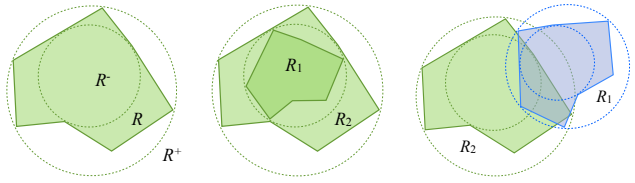
We can generate polygons by placing n vertices on the upper approximate circle, evenly distributed (satisfying Condition 3), such that each vertex and line segment is geometrically constrained to satisfy Conditions 1 and 2 above. The user can explore the space of consistent polygons directly through dynamic geometry [33], or polygons can be randomly generated.

Relative orientation between egg-yolk regions and lines can now be defined based on the upper approximations (see Figure 6(a)):

$$\mathrm{LEFT}(R,\ l_{ab}) \equiv \mathrm{LEFT}(R^+, l_{ab})$$

$$\mathrm{RIGHT}(R,\ l_{ab}) \equiv \mathrm{RIGHT}(R^+, l_{ab})$$

The following topological relations between pairs of egg-yolk regions are defined based on the relation between their approximations (see Figure 8):



(a) Egg-yolk region. (b) $\mathrm{PP}(R_1, R_2)$ (c) $\mathrm{PO}(R_1, R_2)$

Fig. 8. Egg-yolk region R is defined by a lower circular approximation R^- and an upper circular approximation R^+.

$$\mathrm{PP}(R_1, R_2) \equiv \mathrm{P}(R_1^+, R_2^-)$$
$$\mathrm{DC}(R_1, R_2) \equiv \mathrm{DC}(R_1^+, R_2^+)$$
$$\mathrm{DR}(R_1, R_2) \equiv \mathrm{DR}(R_1^+, R_2^+)$$
$$\mathrm{PO}(R_1, R_2) \equiv \mathrm{PO}(R_1^+, R_2^+) \wedge \mathrm{PO}(R_1^-, R_2^-)$$

The *partial overlap* definition requires some explanation: firstly, the partial overlap condition between the upper approximations ensures that the regions each have interior regions not shared by the other, but the regions could still be disconnected. Secondly, the partial overlap condition between the lower approximations ensures that the regions share a common interior region, but one region might completely contain the other. Thus, together the conditions encode the *partial overlap* relation between egg-yolk regions.

The above relative orientation and topological egg-yolk relation encodings are *sound*, i.e. they correctly encode the intended relation between the true regions, and are *incomplete*, i.e. they do not capture all possible ways that the true regions can satisfy the intended relation.

IV. Empirical Evaluation

In this section we empirically evaluate our geometric encodings on a range of benchmark problems from the Qualitative Spatial Reasoning community. We have implemented the encodings in CLP(QS) using geometric constraint solvers FreeCAD[7] and limited BFGS-B [9]. The results show that (a) our system can handle problems from these benchmarks that are unsolvable using relation algebraic methods for qualitative spatial reasoning, and (b) our system outperforms other symbolic algebraic approaches for these benchmark problems by orders of magnitude (such as Cylindrical Algebraic Decomposition). Experiments were run on a MacBookPro, OS X 10.8.5, 2.6 GHz, Intel Core i7.[8]

A. Tent Benchmark Problem

The generalised tent problem [23] is a relative orientation benchmark problem in qualitative spatial reasoning. The problem is specifically designed to be unsolvable using relation algebraic approaches by creating inconsistent relative orientation constraints that cannot be determined to be inconsistent using those methods. Given a set of n distinct 2D points $p_1, \ldots, p_n$, let $\mathrm{LEFT}(p_1, p_2, p_3)$ be *true* if p_3 is *left of* the line (p_1, p_2)

[7] www.freecadweb.org

[8] 3D visualisations have been rendered using glc_player: www.glc-player.net

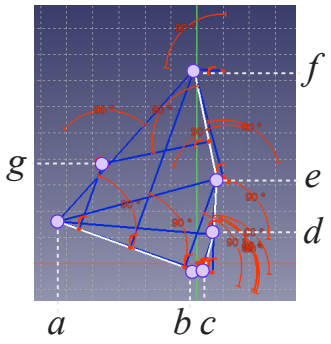


Fig. 9. Consistent tent configuration with $n = 7$ points: $a, \ldots, g$.

where *left of* is interpreted axiomatically (e.g. using relation algebras [26] or a first-order spatial logic [5]) or polynomially (using analytic methods). The consistent tent problem has constraints: $\mathrm{LEFT}(p_i, p_j, p_k)$ for all $0 \leq i < j < k \leq n$. The *in*consistent tent problem has the same constraints with the exception that $\mathrm{LEFT}(p_1, p_2, p_n)$, $\mathrm{LEFT}(p_2, p_3, p_n)$ are replaced with the equivalent RIGHT constraints.

Using an off-the-shelf geometric constraint solver, FreeCAD, our encodings can solve the tent problem for $n = 7$ within 20 seconds for both consistency and inconsistency, which significantly out performs symbolic approaches (in [23] the authors report that quantifier elimination algorithms could not solve for $n = 6$ after 6 hours). We also note that, once the consistent scene has been constructed, incremental updates occur in real-time[9] e.g. the user can move points around the scene and the FreeCAD solver manipulates the other objects to ensure that the relative orientation constraints are maintained (see Figure 9).

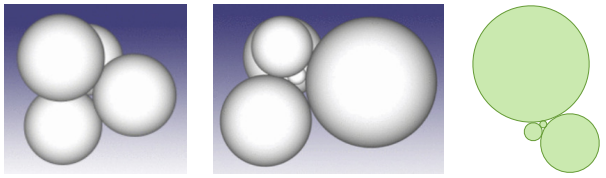
B. Contact problems

A range of contact problems require combining topological and size information that are not solvable using relation algebraic methods. Standard approaches to QSR employ algebraic closure by ensuring that all sub-graphs with 3 vertices are satisfiable. Thus, any problem that inherently requires checking four or more objects simultaneously is beyond algebraic closure. We consider the task of determining the maximum number of spheres and circles that can be mutually externally connected; we also consider the variation where the spheres (circles) must all be the same size.

CLP(QS) correctly solves these spatial contact problems: 5 spheres; 4 same-size spheres; 4 circles; 3 same-size circles (see Figure 10). To demonstrate the scalability of the approach, we consider 100 same size circles (illustrated in Figure 11) and 100 spheres. All of the above contact problems for $n < 10$ are solved within 2 seconds in CLP(QS). Both contact problems with $n = 100$ are solved in 20 seconds.

At each iteration, CLP(QS) produces a series of configurations as it attempts to converge on the solution; an extract of snapshots of the solution history for $n = 100$ same

[9] FreeCAD solver reports solve time of 0.004s.



(a) 4 spheres (equal size). (b) 5 spheres. (c) 4 circles.

Fig. 10. Contact benchmark problems.

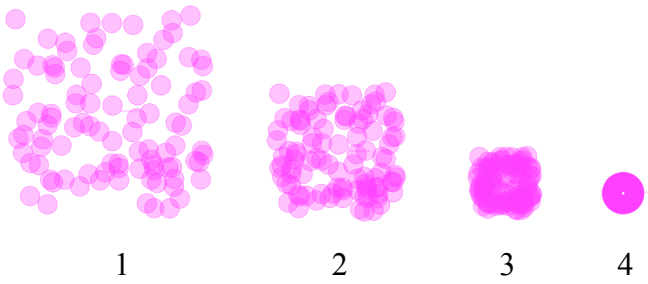


Fig. 11. Four snapshots of the *solving history* of CLP(QS) attempting to solve 100 mutually touching circles, equal size. Snapshot 4 is the best solution found.

size circles is illustrated in Figure 11. By treating time as a third dimension we can construct a *space-time history* region [17, 18] from the sequence of intermediate configurations as illustrated in Figure 12. We can directly visualise and study the qualitative relations between these history regions to provide further insight into the nature of the problem at hand and the underlying solving process, e.g. automatically generating natural language explanations of the search space explored during the solving process.

C. Computer Aided Product Design

The task is to design an adjustable desk lamp with the following qualitative requirements: the lamp has a base and three bars connected by three joints; the joints can only turn *inwards*; the lamp shade connects to the third joint; the bulb must fit completely within the lamp shade. Figure 13 illustrates the constraint graph and corresponding FreeCAD interactive diagram that maintains the specified qualitative relations. As the user manipulates the diagram, the FreeCAD geometric solver maintains the qualitative constraints in real time (reported solving time: 0.001 seconds).

V. Discussion and Conclusions

We have presented novel geometric encodings of qualitative relations for: relative orientation between lines and either points or regions; mereotopological relations between circles (or, more accurately, disks); topological relations between regions represented by an upper and lower approximation (i.e. providing an approach for reasoning about polygons and regions bounded by jordan curves).

Importantly, our encodings are based on a standard geometric constraint language that is well known within the

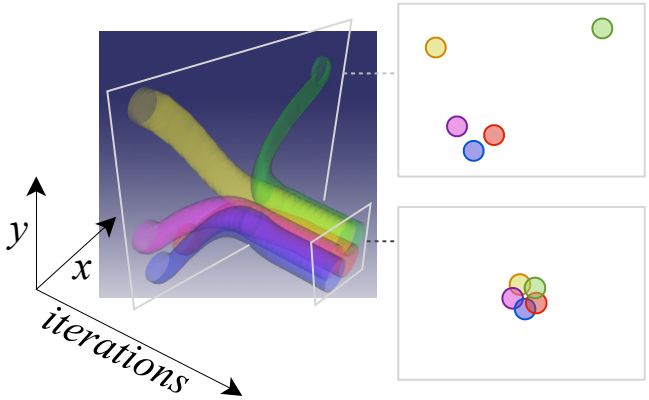
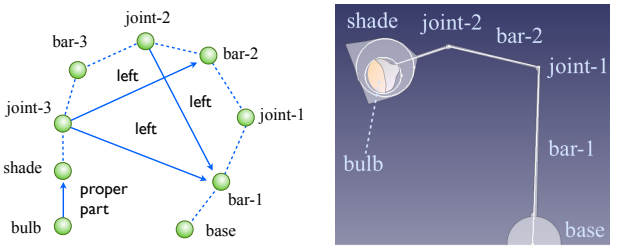


Fig. 12. *Space-time history* volumes derived from intermediate configurations produced during the solving process. Task requirement is five mutually touching, same-sized circles.



(a) Lamp constraint graph. (b) Interactive lamp diagram with qualitative constraints.

Fig. 13. Lamp shade product design with qualitative constraints.

constructive geometric constraint solving community: points, lines, and circles, with relations that correspond to *quadratic equations* (parallel, coincidence, perpendicularity, dimension constraints, etc.). This standard language is adopted by all prominent solvers that we are aware of, particularly in the Computer Aided Design and Manufacturing domain. Thus, our encodings can be directly employed in *all* prominent commercially available constructive geometric constraint solvers (including Autodesk Inventor, FreeCAD, LEDAS LGS2D) to extend those systems to also reason about qualitative spatial relations.

Constructive geometric constraint solving is significantly more computationally efficient compared to other approaches for solving systems of polynomial constraints (e.g. Cylindrical Algebraic Decomposition, the Gröbner basis method, and Wu's characteristic set method all have double exponential complexity), and can scale to real-world problems that involve hundreds of spatial objects. However, the supported spatial language is rather restricted, only permitting relations that can be encoded as quadratic equations. Traditionally, qualitative orientation and mereotopological relations are encoded using *inequalities*, thus ruling out the use of standard geometric constraint solvers; directly encoding inequalities for numerical optimisation methods is also known to be significantly less robust compared to quadratic equations, as it introduces

saddles near optima.

Our key contribution is showing that indeed we can encode these qualitative relations based on the restricted standard geometric constraint language. Thus, we have an efficient method for solving problems involving qualitative orientation and mereotopology beyond computationally intractable symbolic methods. The results of our empirical evaluation, based on an implementation in FreeCAD and limited BFGS-B (providing numerical optimisation, which is effective for solving certain sub-problems), show that our encodings can solve contact and orientation benchmark problems within seconds that take hours (or more) using other symbolic approaches such as Cylindrical Algebraic Decomposition.

One open problem is determining the most efficient encodings for certain problem classes. Our encodings are certainly not unique, and many alternative encodings can be employed that have different properties, e.g. comparing encodings that minimise the number of objects and constraints introduced as the problem size increases, or encodings that avoid the use of computationally more expensive constructions (i.e. encodings that introduce more objects and constraints, but simpler constraints to solve for). We are continuing to identify more efficient encodings to further increase the horizon of solvable real-world problems, and benchmark problems, within the context of declarative constraint logic programming over qualitative spatial domains.

Another interesting open question is how to handle inconsistent qualitative constraints in general: methods such as Cylindrical Algebraic Decomposition are both sound and complete, whereas constructive geometric constraint solving is incomplete in general. Thus, a result of inconsistency using constructive approaches is usually annotated with some measure of confidence (i.e. the problem, or sub-problems, are executed a number of times with different initial randomised parameter values until no further progress towards a solution is made). Identifying tractable classes of qualitative problems that have specific properties with respect to completeness (and statistical confidence in the case of reported inconsistency) is an interesting direction for future research.

VI. Acknowledgements

We sincerely thank the reviewers for their very careful reading of the paper, particularly in their diligence in reviewing our definitions. This work is funded by the German Research Foundation (DFG) under grant for a SketchMapia project (Grant SCHW 1372/7-1).

References

[1] M. Aiello, I. E. Pratt-Hartmann, and J. F. v. Benthem. *Handbook of Spatial Logics*. Springer-Verlag New York, Inc., Secaucus, NJ, USA, 2007. ISBN 978-1-4020-5586-7.

[2] D. S. Arnon, G. E. Collins, and S. McCallum. Cylindrical Algebraic Decomposition I: The basic algorithm. *SIAM Journal on Computing*, 13(4):865–877, 1984.

[3] M. Bhatt, J. H. Lee, and C. Schultz. CLP(QS): A Declarative Spatial Reasoning Framework. In *Proceedings of the 10th international conference on Spatial information theory*, COSIT'11, pages 210–230, Berlin, Heidelberg, 2011. Springer-Verlag. ISBN 978-3-642-23195-7.

[4] M. Bhatt, C. Schultz, and C. Freksa. The 'Space' in Spatial Assistance Systems: Conception, Formalisation and Computation. In T. Tenbrink, J. Wiener, and C. Claramunt, editors, *Representing space in cognition: Interrelations of behavior, language, and formal models. Series: Explorations in Language and Space*. 978-0-19-967991-1, Oxford University Press, 2013.

[5] S. Borgo. Euclidean and mereological qualitative spaces: A study of SCC and DCC. In C. Boutilier, editor, *IJCAI 2009, Proceedings of the 21st International Joint Conference on Artificial Intelligence, Pasadena, California, USA, July 11-17, 2009*, pages 708–713, 2009. URL http://ijcai.org/papers09/Papers/IJCAI09-123.pdf.

[6] D. Bouhineau. Solving geometrical constraint systems using CLP based on linear constraint solver. In *Artificial Intelligence and Symbolic Mathematical Computation*, pages 274–288. Springer, 1996.

[7] D. Bouhineau, L. Trilling, and J. Cohen. An application of CLP: Checking the correctness of theorems in geometry. *Constraints*, 4(4):383–405, 1999.

[8] W. Bouma, I. Fudos, C. Hoffmann, J. Cai, and R. Paige. Geometric constraint solver. *Computer-Aided Design*, 27(6):487–501, 1995.

[9] R. H. Byrd, P. Lu, J. Nocedal, and C. Zhu. A limited memory algorithm for bound constrained optimization. *SIAM Journal on Scientific Computing*, 16(5):1190–1208, 1995.

[10] S.-C. Chou. *Mechanical geometry theorem proving*, volume 41. Springer Science & Business Media, 1988.

[11] A. G. Cohn and N. M. Gotts. The Ôegg-yolkÕrepresentation of regions with indeterminate boundaries. *Geographic objects with indeterminate boundaries*, 2: 171–187, 1996.

[12] G. E. Collins. Quantifier elimination for real closed fields by cylindrical algebraic decompostion. In *Automata Theory and Formal Languages 2nd GI Conference Kaiserslautern, May 20–23, 1975*, pages 134–183. Springer, 1975.

[13] G. E. Collins and H. Hong. Partial cylindrical algebraic decomposition for quantifier elimination. *Journal of Symbolic Computation*, 12(3):299 – 328, 1991. ISSN 0747-7171. doi: http://dx.doi.org/10.1016/S0747-7171(08)80152-6. URL http://www.sciencedirect.com/science/article/pii/S0747717108801526.

[14] X.-S. Gao and S.-C. Chou. Solving geometric constraint systems. ii. a symbolic approach and decision of rc-constructibility. *Computer-Aided Design*, 30(2):115–122, 1998.

[15] X.-S. Gao and S.-C. Chou. Solving geometric constraint systems. i. a global propagation approach. *Computer-Aided Design*, 30(1):47–54, 1998.

[16] J.-X. Ge, S.-C. Chou, and X.-S. Gao. Geometric constraint satisfaction using optimization methods. *Computer-Aided Design*, 31(14):867–879, 1999.

[17] P. J. Hayes. The second naive physics manifesto. In J. R. Hubbs and R. C. Moore, editors, *Formal Theories of the Commonsense World*. Ablex Publishing Corporation, Norwood, NJ, 1985.

[18] S. M. Hazarika. *Qualitative Spatial Change : Space-Time Histories and Continuity*. PhD thesis, The University of Leeds, School of Computing, 2005. Supervisor - Anthony Cohn.

[19] T. L. Heath (ed). *The thirteen books of Euclid's Elements*, volume 1. Courier Dover Publications, 1956.

[20] D. Kapur and J. L. Mundy, editors. *Geometric Reasoning*. MIT Press, Cambridge, MA, USA, 1988. ISBN 0-262-61058-2.

[21] P. B. Ladkin and R. D. Maddux. On binary constraint problems. *Journal of the ACM (JACM)*, 41(3):435–469, 1994.

[22] J. H. Lee. The complexity of reasoning with relative directions. In *21st European Conference on Artificial Intelligence (ECAI 2014)*, 2014.

[23] J. H. Lee and D. Wolter. A new perspective on reasoning with qualitative spatial knowledge. In *IJCAI-2011 Workshop 27*, page 3, 2011.

[24] Y.-T. Li, S.-M. Hu, and J.-G. Sun. A constructive approach to solving 3-d geometric constraint systems using dependence analysis. *Computer-Aided Design*, 34(2):97–108, 2002.

[25] G. Ligozat. *Qualitative Spatial and Temporal Reasoning*. Wiley-ISTE, 2011.

[26] G. F. Ligozat. Qualitative triangulation for spatial reasoning. In *Spatial Information Theory A Theoretical Basis for GIS*, pages 54–68. Springer, 1993.

[27] J. C. Owen. Algebraic solution for geometry from dimensional constraints. In *Proceedings of the first ACM symposium on Solid modeling foundations and CAD/CAM applications*, pages 397–407. ACM, 1991.

[28] G. Pesant and M. Boyer. QUAD-CLP (R): Adding the power of quadratic constraints. In *Principles and Practice of Constraint Programming*, pages 95–108. Springer, 1994.

[29] G. Pesant and M. Boyer. Reasoning about solids using constraint logic programming. *Journal of Automated Reasoning*, 22(3):241–262, 1999.

[30] D. A. Randell, Z. Cui, and A. Cohn. A spatial logic based on regions and connection. In *KR'92. Principles of Knowledge Representation and Reasoning*, pages 165–176. Morgan Kaufmann, San Mateo, California, 1992.

[31] C. Schultz and M. Bhatt. Towards a Declarative Spatial Reasoning System. In *20th European Conference on Artificial Intelligence (ECAI 2012)*, 2012.

[32] C. Schultz and M. Bhatt. Declarative spatial reasoning with boolean combinations of axis-aligned rectangular polytopes. In *ECAI 2014 - 21st European Conference on Artificial Intelligence*, pages 795–800, 2014.

[33] H. Winroth. *Dynamic projective geometry*. Tekniska högsk., 1999.