Double-Crossing: Decidability and Computational Complexity of a Qualitative Calculus for Navigation

Alexander Scivos and Bernhard Nebel

alexander.scivos@sap.com and nebel@informatik.uni-freibrg.de
Universität Freiburg, Institut für Informatik
Georges-Köhler-Allee, Geb. 52, D-79110 Freiburg, Germany

Abstract. The *Double Cross* calculus has been proposed for the purpose of navigation based on qualitative information about spatial configurations. Up until now, however, no results about algorithmic properties of this calculus are known. First, we explore the possibility of applying constraint propagation techniques to solve the reasoning problem in this calculus. For this purpose, we have to generalize the known techniques for binary relations because the Double Cross calculus is based on ternary relations. We will show, however, that such a generalization leads to problems. The Double Cross calculus is not closed under composition and permutation. Further, as we will show, there exists no finite refinement of the base relations with such a closure property. Finally, we show that determining satisfiability of constraint systems over Double Cross relations is NP-hard, even if only the base relations of the Double Cross calculus are used. On the positive side, however, we show that the reasoning problem is solvable in PSPACE.

Keywords: languages of spatial relations, spatial and temporal reasoning, incomplete or imprecise spatial knowledge, qualitative spatial reasoning, constraint propagation, ternary relation systems

1 Introduction

Representing temporal and spatial information and reasoning about this information is an important subproblem in many applications, such as geographical information systems (GIS), natural language understanding, and document interpretation. Often this information is only available qualitatively, for instance when a natural language sentence or a GIS query has to be interpreted. In this case, qualitative approaches to reasoning about space and time have to be used.

Meanwhile there exist a number of qualitative temporal and spatial calculi, such as Allen's [1] interval calculus for reasoning about time, a calculus for reasoning about qualitative directions [4], and a calculus for reasoning about topological relations [2, 21]. For all of these calculi, their computational complexity has been analyzed, computationally tractable fragments have been identified, and algorithms have been specified [27, 13, 20, 19, 15, 9, 24, 23], which are variations of Ladkin and Reinefeld's [13] scheme of using backtracking employing the *path-consistency algorithm* [17] as a forward checking technique.

Another qualitative spatial calculus is Freksa's [5, 6, 28] calculus for reasoning about orientation. In contrast to the calculi mentioned above, almost no formal properties of this calculus are known. In particular, no results about decidability and complexity are known, a topic we will address in this paper.

Freksa's calculus, which is often called *Double Cross calculus*, allows one to specify the qualitative position of one point with respect to an oriented line segment. With this calculus, path descriptions can be combined and evaluated:

1. From a go to b and make a right turn aiming forward to a point c.
2. From b goto to c and make a (perpendicular) right turn going to d.

From these two descriptions, it is possible to infer that a and d cannot be identical (see Figure 1).

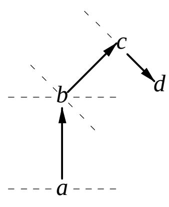


Fig. 1. Example for using qualitative path information

As is evident from the example above, the Double Cross calculus uses *ternary relations* between points, which distinguishes it from the qualitative calculi mentioned above where binary relations between the objects of interest are used. For this reason, the standard techniques (such as using the path-consistency algorithm [17]) do not appear to be applicable to the Double Cross calculus. However, there is, of course, the hope that suitable generalizations will provide us with methods for deciding semantic properties with simple syntactic operations – perhaps on a fragment of the whole calculus.

The remainder of the paper is structured as follows. In the next section we present the calculus. In the following Section 3, we use a generalization of techniques developed for binary constraint systems to ternary systems [10] and show that the Double Cross calculus is not a calculus with a finite set of atomic relations. In Section 4, we show that the Double Cross calculus is computationally inherently difficult. Even the problem of deciding satisfiability for a constraint system containing only base relations is already NP-hard. Finally, in Section 5 we show that reasoning in the calculus is decidable—in fact, reasoning is in PSPACE.

2 The Double-Cross Calculus

Freksa [5] tried to identify an abstract vocabulary to express the qualitative information an observer has in a spatial (2-dimensional) situation. He argues that relative spatial orientation and a front/back dichotomy are natural qualitative dimensions. These ideas

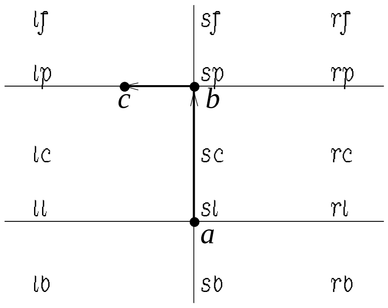


Fig. 2. The 15 base relations of the *Double Cross* calculus

led him to develop what is now called the *Double Cross* calculus, in which three points are related by one of 15 base relations (see Figure 2).

Starting from observer position a and looking to location b, one can describe qualitatively the position of a location c.[1] In Figure 2, this location is to the left of the oriented line given by (a, b) and on a line that is perpendicular to (a, b) going through b. Such a configuration is described using the relation $\mathfrak{lp}$ (left-perpendicular). Similarly, we use

- $\mathfrak{lf}$ (left-forward) to describe configurations where point c is left of (a, b) and "in front of" the perpendicular line going through b,
- $\mathfrak{lc}$ (left-center) to describe configurations where c is left of (a, b) and between the two perpendicular lines going through a and b,
- $\mathfrak{ll}$ (left-line) to describe configurations where c is left of (a, b) and on the perpendicular line going through a, and
- $\mathfrak{lb}$ (left-back) to describe configurations where c is left of (a, b) and "in the back" of the perpendicular line going through a.

Configurations where point c is on the oriented line given by (a, b) are described using the relations $\mathfrak{sf}$ (straight-front), $\mathfrak{sp}$ (straight-second-point), $\mathfrak{sc}$ (straight-center), $\mathfrak{sl}$ (straight-same-location), and $\mathfrak{sb}$ (straight-back). Furthermore, the relations for configurations where c is right of (a, b) are named in a similar manner as the relations describing the situations when c is on the left side. Finally, since we want to describe all configurations with three points involved, we will also consider the pathological situation when $a = b$, which gives us two additional relations: $\mathfrak{eq}$ when $a = b = c$ and $\mathfrak{ex}$ when $a = b \neq c$. The resulting set of 17 ternary relations will be denoted by $\mathcal{D}$ in the sequel.

In general we will use the notation $r(a, b : c)$ in order to express that the points a, b, and c are in relation r. Further, we will also consider *unions* of relations in order to express incomplete information. Such unions will be expressed by writing the set of relations that are used in the union. For example, $\{\mathfrak{lf},\mathfrak{sf},\mathfrak{rf}\}(a, b, c)$ expresses that c is in front of the perpendicular line going through b. The symbol $\top$ denotes the special *universal relation* that is the union of all relations (and therefore does not restrict anything at all). The symbol $\bot$ denotes the empty, impossible relation. The set of all 2^{17} possible relations including $\top$ and $\bot$ is denoted by $\mathcal{DC}$.

[1] We assume that all points are elements of $\mathbb{R}^2$.

A set Θ of atomic formulae using relations from $\mathcal{DC}$ over a set of (implicitly) existentially quantified variables is called a *constraint system*. For instance,

$$\Theta = \{\mathfrak{rf}(a, b : c), \mathfrak{rp}(b, c : d)\},$$

is such a constraint system over the variables $a, b, c,$ and d. A *solution* of a constraint system is an instantiation of all variables to objects of interest, i.e. in our case points in $\mathbb{R}^2$, such that all atomic formulae (constraints) are satisfied. For instance, a solution of Θ would be $\{a = (0, 0), b = (0, 1), c = (1, 2), d = (2, 1)\}$.

The computational problem we are mainly interested in is to decide *satisfiability* of a constraint system Θ.[2] For example, given the two descriptions in the Introduction $\{\mathfrak{rf}(a, b : c), \mathfrak{rp}(b, c : d)\}$, we want to know whether this set is satisfiable if we identify a with d, i.e., if we add $\mathfrak{eq}(a, d, e)$. This means we want to decide the satisfiability of the constraint system

$$\Theta' = \{\mathfrak{rf}(a, b : c), \mathfrak{rp}(b, c : d), \mathfrak{eq}(a, d, e)\}.$$

3 Generalizing Binary Constraint Systems

Since the Double Cross calculus is similar to other qualitative calculi, it seems reasonable to try to adapt techniques used in this context to the Double Cross calculus. Although the other qualitative calculi address quite different aspects of time and space, their formal framework is very similar. A finite set of *jointly exhaustive* and *pairwise disjoint* (*JEPD*), *binary* relations is used to describe the relationships between the objects of interest. Unions of such relations can be used to assert incomplete information. Such description can be interpreted as *binary constraint problems*, where we have infinite domains for the variables. Interestingly, for all of the above mentioned calculi, the JEPD relations are atoms of a relation algebra [12].

The algorithmic technique used to deal with such constraint systems is the *path-consistency algorithm* [17]. This algorithm can be used to transform a given constraint network into a more restricted one that is equivalent to the original one in the sense that it has the same solutions. The main idea is to apply the operations of *composition*, *conversion*, and *intersection* to a constraint system until a fix-point is reached. Furthermore, the path-consistency algorithm is usually *refutation-complete* for fragments of the calculi mentioned above. In particular, for all the calculi mentioned in the Introduction, the path-consistency algorithm is refutation complete for the set of relations containing only the base relations and the universal relation $\top$ [26, 15, 18].

As mentioned already, the above techniques are not immediately applicable to the Double Cross calculus, because it uses ternary relations. However, one may hope that a suitable generalization is. One possible way to go is to generalize composition and conversion operations [10]. The next question will then be if we get interesting properties (such as k-consistency and/or refutation-completeness) when we close a constraint system under these operations and intersection.

[2] The reason for our interest in satisfiability is that all other interesting reasoning problems in the context of CSPs can be reduced in polynomial time to satisfiability [8].

3.1 Permutations

Considering first the converse of relations (exchanging the arguments of a binary relation), it is obvious that we now have more than one way to exchange arguments. Because we have three arguments, we have $3! = 6$ possible ways of arranging the arguments. Following Zimmermann and Freksa [28], we use the following terminology and symbols to refer to these *permutations* of the arguments $(a, b : c)$:

term	symbol	arguments
identical	$\textsc{Id}$	$a, b : c$
inversion	$\textsc{Inv}$	$b, a : c$
short cut	$\textsc{Sc}$	$a, c : b$
inverse short cut	$\textsc{Sci}$	$c, a : b$
homing	$\textsc{Hm}$	$b, c : a$
inverse homing	$\textsc{Hmi}$	$c, b : a$

The inversion of a relation is exemplified in Figure 3, demonstrating that $\textsc{Inv}(\mathfrak{rc}) = \mathfrak{lc}$.

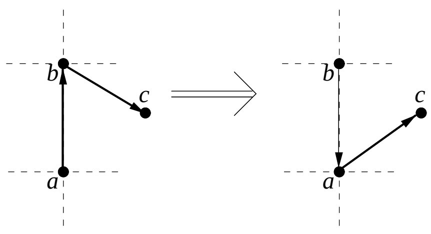


Fig. 3. $\textsc{Inv}$ of the $\mathfrak{rc}$ relation

Taking a closer look at the permutations, it turns out that the $\mathcal{DC}$ relations are *not closed* under permutations in the following sense. For some relation $r \in \mathcal{DC}$ and some permutation π, there is no relation $r' \in \mathcal{DC}$ such that $\pi(r) = r'$.

The original calculus as proposed by Freksa [5] using only 15 relations (without $\mathfrak{ex}$ and $\mathfrak{eq}$) is not closed under permutations since $\textsc{Hm}(\mathfrak{sp}) = \mathfrak{ex}$. However, even with these relations, the set of relations is not closed. Take for example the relation $\mathfrak{lf}$. If we consider the short cut of this relation $\textsc{Sc}(\mathfrak{lf})$, then the three points must be in the $\mathfrak{rc}$ relation (see Figure 4). In other words, we have $\textsc{Sc}(\mathfrak{lf}) \subseteq \mathfrak{rc}$. However, there are some elements in $\mathfrak{rc}$ which are not in $\textsc{Sc}(\mathfrak{lf})$. For example, we have $\mathfrak{rc}(\langle 0, 0\rangle, \langle 0, 2\rangle : \langle 2, 1\rangle)$, but we do not have $\mathfrak{lf}(\langle 0, 0\rangle, \langle 2, 1\rangle : \langle 0, 2\rangle)$.

Proposition 1. *The $\mathcal{DC}$ relations are not closed under permutations.*

It is possible to refine the base relations [14], and with this refinement the set of $\mathcal{DC}$ relations becomes closed under permutations. The main idea is to introduce new relations that make finer distinctions for the $\mathfrak{rc}$ and $\mathfrak{lc}$ relations (see Figure 5). The relation $\mathfrak{elc}$ holds if point c is between the two perpendicular lines and outside the circle going through a and b, $\mathfrak{olc}$ holds if c is on the circle, and $\mathfrak{ilc}$ holds if c is inside the

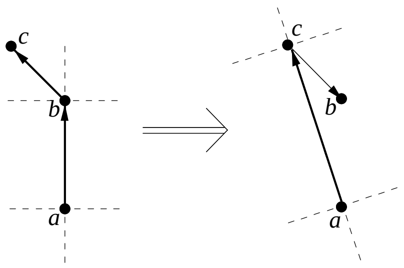


Fig. 4. Sc of *lf* relation

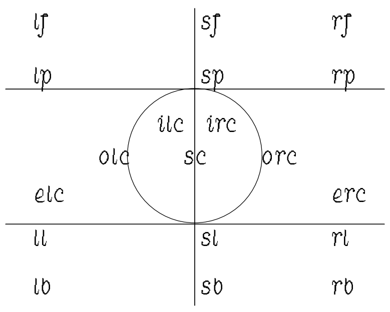


Fig. 5. Refined $\mathcal{D}$ relations: $\mathcal{RD}$

circle. Similarly, we make the corresponding distinctions on the right side. With this refinement, the set of relations is closed under permutations. We call the refined set of base relations $\mathcal{RD}$ and the set of relations resulting from all possible unions of base relations $\mathcal{RDC}$.

Proposition 2. *The $\mathcal{RDC}$ relations are closed under permutations.*

3.2 Compositions

With ternary relations, one can think of different ways of composing them. However, there are only a few ways to compose them in a way such that we can use it for enforcing *local consistency* [16]. In trying to generalize the path-consistency algorithm [17], we want to enforce "4-consistency." A constraint system is called *k-consistent* iff for all subsets of k variables any instantiation of $k-1$ variables that satisfies all involved constraints can be extended to an instantiation of the k variables satisfying all involved constraints.

For this reason, let us consider the *composition* of relations r_1 and r_2, in symbols $r_1 \diamond r_2$, defined as follows:[3]

$$\forall a, b, d \colon (r_1 \diamond r_2)(a, b : d) \leftrightarrow \exists c \colon r_1(a, b : c) \wedge r_2(a, c : d).$$

[3] Note that this kind of composition operation is identical to the one defined by Isli and Cohn [10] for ternary relation systems.

This is not the only possible way to compose two ternary relations. However, all other "reasonable" ways to compose two ternary relations can be expressed by a permutation of composing two permutations of relations.

Let us now assume that we have given a constraint system Θ. Further, let Θ^* denote this constraint system *closed* under intersection, permutation, and composition as defined above. This means that further application of intersection, permutation, or composition do not lead to deriving a new constraint (see the constraint propagation algorithm by Isli and Cohn [10]). Then, we know that Θ^* is 4-consistent. This follows from the fact that the relation between any three points a, b, d has been intersected with any composition over a fourth point c, which implies that we can extend a given instantiation of a, b, d satisfying all constraints to an instantiation of a, b, d and c without violating a constraint.

However, how do we compute Θ^*? Provided the composition of two relations from $\mathcal{RDC}$ is a relation in $\mathcal{RDC}$, this is an easy (spell polynomial) problem. Unfortunately though, $\mathcal{RDC}$ is *not closed under composition*, i.e., there exist two relations $r_1, r_2 \in \mathcal{RDC}$ such that $(r_1 \diamond r_2) \notin \mathcal{RDC}$.

3.3 Closing $\mathcal{RDC}$ under Composition

If we consider the relation $(\mathsf{olc} \diamond \mathsf{olc})$, it turns out that there is no relation in the refined calculus $\mathcal{RDC}$ that is identical to this relation. From the sketch in Figure 6, it is obvious that by varying the position of point c, point d can either be left of (a, b) between the

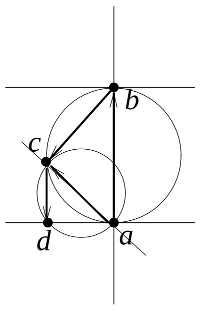


Fig. 6. Composing $\mathsf{olc}(a, b : c)$ and $\mathsf{olc}(a, c : d)$.

two perpendicular lines going through a and b (lc), left of (a, b) and on the line going through a (ll), or left of (a, b) and in the back of the line going through a (lb). In other words, we have $(\mathsf{olc} \diamond \mathsf{olc}) \subseteq \{\mathsf{ll},\ \mathsf{lb},\ \mathsf{lc}\}$. However, $\{\mathsf{ll},\ \mathsf{lb},\ \mathsf{lc}\}$ is not identical to $(\mathsf{olc} \diamond \mathsf{olc})$. There are points d such that for $\mathsf{ll}(a, b : d)$ there does not exist any point c with $\mathsf{olc}(a, b : c)$ and $\mathsf{olc}(a, c : d)$. Any point d on the line through a that is farther away than half the distance between a and b has this property.

In other words, in order to complete our relation system $\mathcal{RDC}$, we must introduce another refinement that distinguishes between points on the line through a according to whether they are close to a or far away from a. In particular, the relation $((\mathsf{olc} \diamond \mathsf{olc}) \cap \mathsf{ll})(a, b, c)$ denotes—for given a and b—the points c on the line through a on the left side of (a, b) and at most half of the distance between a and b away from a. We use the symbol $\mathit{lclose}[0.5]$ to denote this relation. Similarly, we introduce the relation $\mathit{rclose}[0.5]$ for the points on the right side of (a, b). Unfortunately, however, this does not help to complete the relation system since we may be forced to make even more refinements. In fact, there is no finite refinement of the relations system $\mathcal{RDC}$ that is closed under composition and intersection.

Theorem 1. *The $\mathcal{RDC}$ relations are not closed under composition and intersection and there exists no finite refinement of these relations with that property.*

Proof. Using the relations $\mathit{lclose}[0.5]$ and $\mathit{rclose}[0.5]$, we will show that we can make even finer distinction. In particular, we show that for any relation $\mathit{lclose}[r/2]$, with $0 < r \leq 1$, we can construct a new relation $\mathit{lclose}[r/8]$ using composition and intersection.

Assume that there exists the relation $\mathit{lclose}[r/2]$ for some $0 < r \leq 1$. Then $(\mathit{lclose}[r/2] \diamond \mathit{rclose}[0.5]) \cap \mathit{sc}$ denotes—for fixed first and second argument—all points on the line between the first and second argument that are at most $r/4$ away from the first argument. Figure 7 illustrates this fact. With $\mathit{lclose}[r](a, b, c)$ and

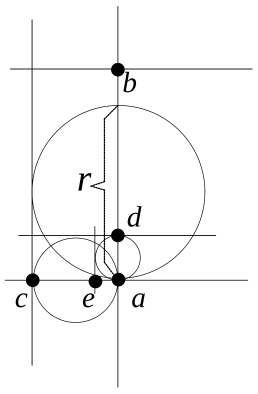


Fig. 7. Constructing $\mathit{lclose}[r/8]$ from $\mathit{lclose}[r/2]$

$\mathit{rclose}[0.5](a, c, d)$ we get for fixed a, b all points d on the line from a to b that are at most $r/4$ away from a.

Using now $((\mathit{lclose}[r] \diamond \mathit{rclose}[0.5]) \cap \mathit{sc})(a, b, d)$ and $\mathit{lclose}[0.5](a, d, e)$, we see that—for fixed a, b—we get all left points e on the line going through a that are no more that $r/8$ away from a. In symbols, we have:

$$\mathit{lclose}[r/8] = (((\mathit{lclose}[r/2] \diamond \mathit{rclose}[0.5]) \cap \mathit{sc}) \diamond \mathit{lclose}[0.5]) \cap \mathit{ll}.$$

This means that there is an infinite number of refinements of $\mathit{lclose}[r/2]$ and hence there cannot exist a finite set of base relations closed under composition and intersection. $\square$

This is very bad news, indeed. From a technical point of view it means that the relation algebra is infinite. In particular, it rules out any backtracking approach, where we consider all constraint systems resulting from refining all constraints to *atomic relations*—relations that cannot be further refined—and testing these atomic constraint systems [13].

One possible way to deal with this problem might be to consider a weaker notion of composition. We define the *weak composition* $r_1 * r_2 = r$ to be the most specific relation $r \in \mathcal{RDC}$ with $r \supseteq r_1 \diamond r_2$." It must be noted, however, that the closure of a constraint system under this *weak composition operation*, permutations and intersection does not entail 4-consistency. Nevertheless, there might be the hope that a constraint system in which all constraints are base relations and which is closed under the above operations is a satisfiable constraint system. In this case, Ladkin and Reinefeld's [13] backtracking approach could be used to determine satisfiability. As we show below, however, this hope is unfounded.

4 Computational Complexity

In almost all qualitative calculi that have been investigated so far, there exists some non-trivial fragment containing all the base relations and the universal relation such that satisfiability can be decided in polynomial time [20, 15, 24]. Can we expect something similar for the Double Cross calculus?

As it turns out, this is not the case. Even if we consider only CSPs over $\mathfrak{sf}$ and the universal relations $\top$, the satisfiability problem is already NP-hard. The reason is that with $\mathfrak{sf}$ we can relate three points on a line saying only which point must be in the middle. This enables us to encode the "betweeness" problem.

Theorem 2. *Satisfiability of constraint systems over* $\{\mathfrak{sf}, \top\}$ *is* NP*-hard.*

Proof. The BETWEENNESS problem [7, p.279] can be straightforwardly reduced to the problem at hand. This problem is defined as follows. Given a finite set M, a collection C of ordered triples (a, b, c) of distinct elements from M, is there a one-to-one function $f\colon\ M \rightarrow \{1, 2, , \ldots, |M|\}$ such that for each $(a, b, c) \in C$, we have either $f(a) < f(b) < f(c)$ or $f(c) < f(b) < f(a)$?

Given an instance of BETWEENNESS, we construct a constraint system Θ as follows. For each $m \in M$, we add $\mathfrak{sf}(x, y, m)$ to Θ, where x and y are two fixed elements not in M. This enforces that all m are on a line. Now we add $\mathfrak{sf}(a, b, c)$ for each tuple $(a, b, c) \in C$. Now it is obvious that Θ is satisfiable iff there is a function f with the desired properties, i.e., the construction is a polynomial transformation, which proves the claim. $\square$

So, we cannot expect a tractable fragment containing all base relations. However, can we expect that satisfiability testing is easy (i.e., polynomial) once all constraints have been refined to base relations as hinted at above?

One of the crucial aspects of the reduction in the proof of Theorem 2 is that we do not have to specify base relations between *all* triples of points. If base relations have to specified between all points, the reduction would not work because there are not enough "degrees of freedom" for the placement of the points on the line. So one might hope that the specification of base relations for all variables in a Double Cross CSP might lead to an easy test for satisfiability. Unfortunately, however, this is not the case.

Theorem 3. *Satisfiability of constraint systems over $\mathcal{RD}$ is* NP*-hard.*

Proof Sketch. The claim is proven by a reduction from 3-SAT. Given an instance of 3SAT with clauses $C_1, \ldots, C_m$ over variables $v_1, \ldots, v_n$, a corresponding CSP of $\mathcal{RD}$ will be defined in a way such that the CSP has a solution iff a truth value assignment for the given instance of 3SAT exists. The proof itself is rather long and involved [25] so that we can only give a sketch here.

The key feature that allows us to construct a CSP using only base relations in the reduction is the fact that some of the base relations are very weak (e.g., $\mathfrak{rf}$) while others are very restrictive (e.g. $\mathfrak{rl}$). The former can be used in an almost completely non-restrictive way if the points are far enough away from each other. The latter relations can be used to "transport" particular geometric information even over very long distances.

In the reduction, we design one component for choosing truth values for the variables, one component for expressing that each clause is made up by three particular literals, and finally a truth-value testing component. Each component consists of a set of variables with appropriate constraints between them. Instead of describing the syntactic constraints between the elements, we will describe the geometric relations we enforce in each solution to the CSP with the understanding that all this can be expressed using the base relations of $\mathcal{RD}$.

The components are "placed" in a way such that all points of the truth choice component are located left behind all points of the literals component which are in turn left behind all points of the clause test component. This is achieved by defining the constraints as permutations of the $\mathfrak{lb}$ relation. The truth values are "transfered" from one component to the next using parallel lines. We therefore use in the proof the fact that with $\mathcal{RD}$ relations we can enforce that some lines form a right angle, that some line segments have the same length or are three times as long as another, etc. (see Figure 8).

All points inside each component are related to each other using base relations to describe the general structure of the CSP solution which is independent from any truth value. Although the design of the CSP can totally be done syntactically, it is instructive to keep in mind the basic idea: All constraints are chosen so that in a solution of the CSP, distances between certain pairs of points automatically code for truth values. If such a distance exceeds a predefined unit length, then the value is thought to be true, otherwise false.

Although the length of a distance between two points cannot be represented by $\mathcal{RD}$ relations, such information can implicitly be contained in constraints. Such implicit information is used in our proof. In fact, the same implicit inconsistencies which may or may not inhibit a 3SAT solution, would disallow a $\mathcal{RD}$ solution.

In order to give an idea of the reduction, we will describe some of the geometric features any solution of a constraint system constructed in the reduction must have. First of all, we set up a framework, in which all components can be placed. In a solution of the CSP, the unit length is defined by the distance between the instantiations of two *base line elements*, x_{01} and x_{02}. The *variable component* contains four elements $x_{i1}, x_{i2}, x_{i3}, y_i$

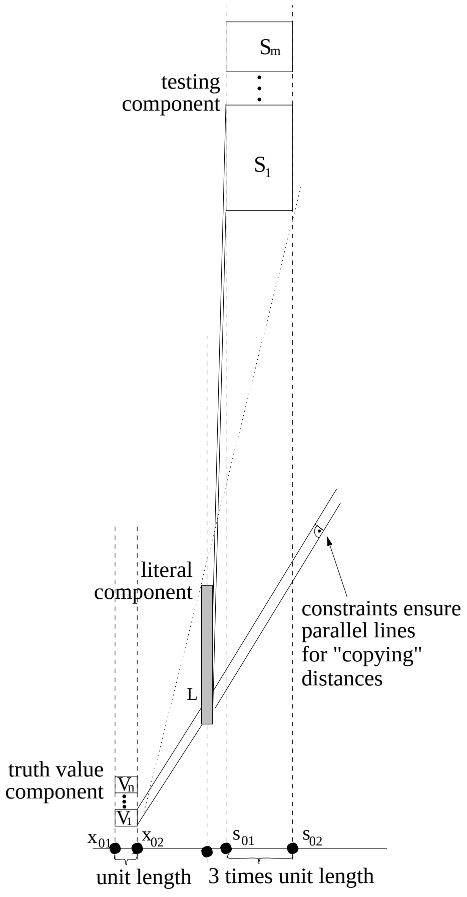
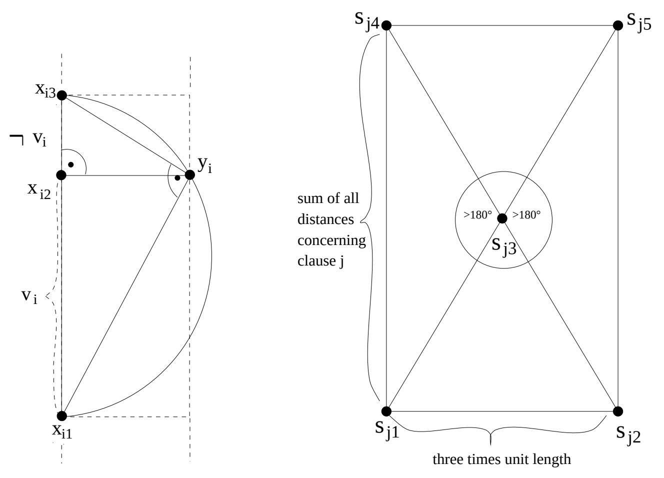


Fig. 8. Components in the reduction

for each variable v_i of the 3SAT problem. The constraints among them are always the same, and can be understood by looking at the intended CSP solution (see Figure 9.a).

For example, by the constraint $\mathit{lp}(x_{i1}, y_i : x_{i3})$, a right angle is required. By permutations of the rp base relation, it can further be ensured that in every CSP solution x_{i2}, y_i, x_{01} and x_{02} form a rectangle, and thus the distance between instantiations of x_{i1} and x_{i2} has to equal the unit length. Note that in every CSP solution it holds that whenever the distance between x_{i1} and x_{i2} is larger than the unit length, then the distance between x_{i2} and x_{i3} is smaller and vice versa. Thus in any instantiation, the first distance can be understood as a representation of v_i's truth value and the latter distance represents the opposite truth value $\neg v_i$.

Now for each clause C_j and every literal l_k that occurs in it, a new pair of elements, l_{jk1} and l_{jk2} is introduced. By right angle constraints and additional supporting elements, it can be ensured that in any solution of the CSP, certain corresponding lines



a. The component V_i b. The component S_j

Fig. 9. Sketch: Example CSP solution of the two major components: a. variable component, b. testing component

are parallel. Then the corresponding distances (themselves representing the truth values of the participating literal) are copied from the *truth value component* to the *literal component*.

Finally, some other components with well-defined constraints ensure that in each CSP solution the distances of all literals of the same clause are added. If in a CSP solution, all distances are small, then its sum will be less then three times the unit distance. The construction of Figure 9.b ensures that the five points $S_{j1}, \ldots, S_{j5}$ which are generated for the clause C_j form a rectangle that is higher than wide, and thus can compare two values. Since the constraints ensure that the distances representing truth values are copied, it is possible to find an instantiation where all sums of distances in a clause exceed 3 times the unit distance, if and only if all clauses can be made true simultaneously.

Some further components have to be added for technical reasons. To ensure parallelism, there have to be two more elements per transfer at which a right angle constraint is defined. Moreover, a basic line has to be defined that helps ordering the elements according to its significance. However, constraints the constraints between these additional elements can be chosen so that there do not arise further complications.[4] □

This result implies that we can rule out the possibility of a polynomial-time refutation-complete algorithm on base relations. This implies that a backtracking approach over base relations, where some additional, polynomial-time computation is done in order to deal with refinements of base relations can be ruled out. In fact, it is not yet clear whether the satisfiability problem for $\mathcal{RDC}$ (or $\mathcal{RD}$) is in NP.

[4] The exact details of the reduction are described in Scivos' [25] Master thesis.

5 Decidability of the Double-Cross Calculus

In the previous section we showed that constraint propagation approaches will not work for the Double Cross calculus. More precisely, we demonstrated that even if only base relations are used, satisfiability cannot be decided in polynomial time. While this excludes backtracking algorithms over refinements of constraint systems that contain only base relations, it does of course not imply that the problem cannot be solved. Contrarily, the satisfiability problem can be easily solved by transforming the constraints to inequalities over polynomials with integer coefficients. For example, we can express the constraint $rp(a, b : c)$ as follows. Given that the coordinates of the points are $(a_1, a_2), (b_1, b_2), (c_1, c_2)$, respectively, the right angle can be expressed by stating that the scalar product of $\overrightarrow{(a, b)}$ and $\overrightarrow{(b, c)}$ is zero:

$$(b_1 - a_1)(c_1 - b_1) + (b_2 - a_2)(c_2 - b_2) = 0$$

The orientation is determined by the inequality

$$(b_1 - a_1)(c_2 - a_2) - (b_2 - a_2)(c_1 - a_1) < 0$$

Taking both constraints together ensures that the points can only be instantiated in a way such that the rp relation is satisfied. Using this insight and a result by Renegar [22], it follows immediately that deciding satisfiability of $\mathcal{RDC}$ constraint systems is decidable.

Theorem 4. *Satisfiability of a $\mathcal{RDC}$ constraint system is in* PSPACE.

Proof Sketch. All the $\mathcal{RD}$-relations can be expressed as sets of inequalities $p_1(x_1, \ldots, x_n) \geq 0, \ldots, p_m(x_1, \ldots, x_n) \geq 0$, with the p_i's being polynomials with integer coefficients [25]. This implies by results of Renegar [22] that the problem is in PSPACE. □

This result gives us an algorithmic technique to determine the satisfiability of $\mathcal{DC}$ constraint systems [22] and it provides us with an upper bound on the complexity of reasoning in the Double Cross calculus. However, we do not have a tight upper bound yet. We only know that it is NP-hard and in PSPACE.

6 Conclusions and Discussion

We analyzed the *Double Cross* calculus proposed by Freksa [5] from an algorithmic point of view. First, we addressed the question of whether it is possible to use a constraint propagation approach to solve the reasoning problems. While inferences based on composition and permutation [5, 6, 28] are, of course, correct, they most probably cannot be used to implement a complete inference mechanism.

Considering obvious generalizations from binary to ternary relations, the constraint techniques of enforcing local consistency, which are usually employed, do not apply in our case. The first problem is that the relation system of the Double Cross calculus is not closed under permutations and compositions. In fact, we have shown that there is no finite refinement of the relation system with this closure property. Furthermore, we showed that for all fragments of the Double Cross calculus containing all base relations and the universal relations we get NP-hard satisfiability problems—in contrast to other known temporal and spatial qualitative calculi with binary relations. Finally, we showed that easy adaptions of the backtracking approach—testing all refinements of a constraint satisfaction problem to base relations—do not work because the satisfiability problem is NP-hard even if only base relations are permitted. The main problem appear to be the non-convex regions definable by relations such as elc.

However, there are other means to solve the satisfiability problem in the Double Cross calculus. All relations in the Double Cross calculus can be formulated as inequalities involving polynomials with integer coefficients only. Whether such a set of inequalities has a solution, then, is a problem that can be decided in PSPACE, which gives us an algorithm and an upper bound on the problem complexity. Future work will be concerned with trying to narrow lower and upper bounds and working on empirically efficient solutions which might involve a combination of constraint propagation and reasoning about inequalities with polynomials.

Comparing the Double Cross calculus with other approaches to spatial representation and reasoning one notes that the Double Cross calculus addresses issues that have also been dealt with in Kuipers' [11] TOUR model. However, in contrast to the TOUR model, the Double Cross calculus exclusively formalizes what has been called two-dimensional orientation in the TOUR model and does so with purely qualitative means. Another related calculus is Faltings' [3] qualitative topological calculus which can also be used to reason about qualitative navigation. However, this calculus is designed for reasoning about $d+1$-ary topological relations between d-dimensional objects and does not take orientation into account at all. So, both approaches are fundamentally different from the Double Cross calculus, and cannot be used to replace the Double Cross calculus. However, it might be quite interesting to combine Faltings' topological calculus with Freksa's calculus in order to create a richer qualitative spatial calculus that is useful for navigation.

References

1. J. F. Allen. Maintaining knowledge about temporal intervals. *Communications of the ACM*, 26(11):832–843, Nov. 1983.
2. M. J. Egenhofer. Reasoning about binary topological relations. In O. Günther and H.-J. Schek, editors, *Proceedings of the Second Symposium on Large Spatial Databases, SSD'91*, volume 525 of *Lecture Notes in Computer Science*, pages 143–160. Springer-Verlag, Berlin, Heidelberg, New York, 1991.
3. B. Faltings. Qualitative spatial reaoning using algebraic topology. In A. U. Frank and W. Kuhn, editors, *Spatial Information Theory: a Basis for GIS*, pages 17–30. Springer-Verlag, 1995.
4. A. Frank. Qualitative spatial reasoning with cardinal directions. In *Proceedings of the Seventh Austrian Conference on Artificial Intelligence*, Berlin, Heidelberg, New York, 1991. Springer-Verlag.

5. C. Freksa. Using orientation information for qualitative spatial reasoning. In A. U. Frank, I. Campari, and U. Formentini, editors, *Theories and Methods of Spatio-Temporal Reasoning in Geographic Space*, pages 162–178. Springer-Verlag, Berlin, Heidelberg, New York, 1992.
6. C. Freksa and K. Zimmermann. On the utilization of spatial structures for cognitively plausible and efficient reasoning. In *Proc. of the IEEE International Conference on Systems, Man, and Cybernetics*, pages 261–266, Chicago, IL, 1992. IEEE.
7. M. R. Garey and D. S. Johnson. *Computers and Intractability—A Guide to the Theory of NP-Completeness*. Freeman, San Francisco, CA, 1979.
8. M. C. Golumbic and R. Shamir. Complexity and algorithms for reasoning about time: A graph-theoretic approach. *Journal of the Association for Computing Machinery*, 40(5):1128–1133, Nov. 1993.
9. M. Grigni, D. Papadias, and C. Papadimitriou. Topological inference. In *Proceedings of the 14th International Joint Conference on Artificial Intelligence (IJCAI-95)*, pages 901–906, Montreal, Canada, Aug. 1995. Morgan Kaufmann.
10. A. Isli and A. G. Cohn. A new approach to cyclic ordering of 2D orientations using ternary relation algebras. *Artificial Intelligence*, 122(1-2):137–187, 2000.
11. B. J. Kuipers. Modelling spatial knowledge. *Cognitive Science*, 2:129–153, 1978.
12. P. B. Ladkin and R. Maddux. On binary constraint problems. *Journal of the Association for Computing Machinery*, 41(3):435–469, May 1994.
13. P. B. Ladkin and A. Reinefeld. A symbolic approach to interval constraint problems. In J. Calmet and J. A. Campbell, editors, *Artificial Intelligence and Symbolic Mathematical Computing*, volume 737 of *Lecture Notes in Computer Science*, pages 65–84. Springer-Verlag, Berlin, Heidelberg, New York, 1993.
14. L. Latecki and R. Röhrig. Orientation and qualitative angle for spatial reasoning. In *Proceedings of the 13th International Joint Conference on Artificial Intelligence (IJCAI-93)*, pages 1544–1549, Chambery, France, Aug. 1993. Morgan Kaufmann.
15. G. Ligozat. Reasoning about cardinal directions. *Journal of Visual Languages and Computing*, 9(1):23–44, 1998.
16. A. K. Mackworth. Consistency in networks of relations. *Artificial Intelligence*, 8:99–118, 1977.
17. U. Montanari. Networks of constraints: fundamental properties and applications to picture processing. *Information Science*, 7:95–132, 1974.
18. B. Nebel. Computational properties of qualitative spatial reasoning: First results. In I. Wachsmuth, C.-R. Rollinger, and W. Brauer, editors, *KI-95: Advances in Artificial Intelligence*, pages 233–244, Bielefeld, Germany, 1995. Springer-Verlag.
19. B. Nebel. Solving hard qualitative temporal reasoning problems: Evaluating the efficiency of using the ORD-Horn class. In *Proceedings of the 12th European Conference on Artificial Intelligence (ECAI-96)*, pages 38–42, Budapest, Hungary, Aug. 1996. Wiley.
20. B. Nebel and H.-J. Bürckert. Reasoning about temporal relations: A maximal tractable subclass of Allen's interval algebra. *Journal of the Association for Computing Machinery*, 42(1):43–66, Jan. 1995.
21. D. A. Randell, Z. Cui, and A. G. Cohn. A spatial logic based on regions and connection. In B. Nebel, W. Swartout, and C. Rich, editors, *Principles of Knowledge Representation and Reasoning: Proceedings of the 3rd International Conference (KR-92)*, pages 165–176, Cambridge, MA, Oct. 1992. Morgan Kaufmann.
22. J. Renegar. On the computational complexity and geometry of the first order theory of the reals. Part I–III. *Journal of Symbolic Computation*, 13(3):255–300, 301–328, 329–352, 1992.
23. J. Renz and B. Nebel. Efficient methods for qualitative spatial reasoning. In *Proceedings of the 13th European Conference on Artificial Intelligence (ECAI-98)*, pages 562–566, Brighton, UK, Aug. 1998. Wiley.

24. J. Renz and B. Nebel. On the complexity of qualitative spatial reasoning: A maximal tractable fragment of the Region Connection Calculus. *Artificial Intelligence*, 108(1-2):69–123, 1999.
25. A. Scivos. Einführung in eine Theorie der ternären RST-Kalküle für qualitatives räumliches Schließen. Diplomarbeit, Albert-Ludwigs-Universität Freiburg, Mathematische Fakultät, 2000.
26. M. B. Vilain and H. A. Kautz. Constraint propagation algorithms for temporal reasoning. In *Proceedings of the 5th National Conference of the American Association for Artificial Intelligence (AAAI-86)*, pages 377–382, Philadelphia, PA, Aug. 1986.
27. M. B. Vilain, H. A. Kautz, and P. G. van Beek. Constraint propagation algorithms for temporal reasoning: A revised report. In D. S. Weld and J. de Kleer, editors, *Readings in Qualitative Reasoning about Physical Systems*, pages 373–381. Morgan Kaufmann, San Francisco, CA, 1989.
28. K. Zimmermann and C. Freksa. Qualitatives räumliches Schließen mit Wissen über Richtungen, Entfernungen und Pfade. *Künstliche Intelligenz*, 7(4):21–28, 1993.