Chapter 1

QUALITATIVE SPATIAL REASONING USING CONSTRAINT CALCULI

Jochen Renz
Australian National University

Bernhard Nebel
Albert-Ludwigs Universität Freiburg

Second Reader
Ian Pratt-Hartmann
Manchester University

1. Introduction

Qualitative reasoning is an approach for dealing with commonsense knowledge without using numerical computation. Instead, one tries to represent knowledge using a limited vocabulary such as qualitative relationships between entities or qualitative categories of numerical values, for instance, using $\{+, -, 0\}$ for representing real values. An important motivation for using a qualitative approach is that it is considered to be closer to how humans represent and reason about commonsense knowledge. Another motivation is that it is possible to deal with incomplete knowledge. Qualitative reasoning, however, is different from fuzzy computation. While fuzzy categories are approximations to real values, qualitative categories make only as many distinctions as necessary—the granularity depends on the corresponding application.

Two very important concepts of commonsense knowledge are time and space. Time, being a scalar entity, is very well suited for a qualitative approach and, thus, qualitative temporal reasoning has early emerged as a lively sub-field of qualitative reasoning which has generated a lot of research effort and important results. Space, in turn, is much more complex than time. This is mainly due to its inherent multi-dimensionality which leads to a higher degree of freedom and an increased possibility of describing entities and relationships between entities. This becomes clear when enumerating natural language expressions involving space or time. While temporal expressions mainly describe order and duration (like “before”, “during”, “long”, or “a while”) or a personal or general temporal category (like “late” or “morning”), spatial expressions are manifold. They are used for describing, for instance, direction (“left”, “above”), distance (“far”,“near”), size (“large”, “tiny”), shape (“oval”, “convex”), or topology (“touch”, “inside”). It is obvious that most spatial expressions in natural language are purely qualitative.

Although there are doubts that because of its multi-dimensionality, space can be adequately dealt with by using only qualitative methods (the poverty conjecture [Forbus et al., 1987]), qualitative spatial reasoning has become an active research area. Because of the richness of space and its multiple aspects, however, most work in qualitative spatial reasoning has focused on single aspects of space. The most important aspects of space are topology, orientation, and distance. As shown in psychological studies [Piaget and Inhelder, 1948], this is also the order in which children acquire spatial notions. Other aspects of space include size, shape, morphology, and spatial change (motion).

Orthogonal to this view is the question for the right spatial ontology. One line of research considers points as the basic entities, another line considers extended spatial entities such as spatial regions as basic entities. While it is easier to deal with points rather than with regions in a computational framework, taking regions as the basic entities is certainly more adequate for commonsense reasoning—eventually, all physical objects are extended spatial entities. Furthermore, if points are required, they can be constructed from regions [Biacino and Gerla, 1991]. A further ontological distinction is the nature of the embedding space. The most common notion of space is n-dimensional continuous space (R^n). But there are also approaches which consider, e.g., discrete [Galton, 1999] or finite space [Gotts, 1996].

The most popular reasoning methods used in qualitative spatial reasoning are constraint based techniques adopted from previous work in temporal reasoning (see Section 2 for a comprehensive introduction to these techniques). In this chapter, we will focus exclusively on these techniques. In order to apply them, it is necessary to have a set of qualitative binary *basic relations* which have the property of being jointly exhaustive and pairwise disjoint, i.e., between any two spatial entities exactly one of the basic relations holds. The set of all possible relations is then the set of all possible unions of the basic relations. Reasoning can be done by exploiting *composition* of relations. For instance, if the binary relation R_1 holds between entities A and B and the binary relation R_2 holds between B and C, then the composition of R_1 and R_2 restricts the possible relationship between A and C. Compositions of relations are usually pre-computed and stored in a *composition table*.

The rest of the chapter is structured as follows. In Section 2, we explain the general idea of constraint-based reasoning. In Section 3, a number of different constraint calculi are introduced, which cover topology, orientation, and distance. Since we are interested in reasoning in these spatial calculi, we have to consider what computational resources are necessary to accomplish that. For this purpose, we will introduce computational complexity theory and explain how it can be applied in the context of constraint based reasoning. As we will see, almost all qualitative spatial calculi are computationally intractable. However, it is impossible to identify tractable subsets, as we will show in Section 5. Based in these results, we will have a look at the *practical efficiency* of reasoning in spatial calculi using tractable fragments in Section 6. Finally, in Section 7, we consider the combination of different spatial calculi.

2. Constraint-based methods for qualitative spatial representation and reasoning

Knowledge about entities or about the relationships between entities is often given in the form of *constraints*. For instance, when trying to place furniture in a room there are certain constraints on the position of the objects. Unary constraints such as "The room is 5 metres in length and 6 in breadth" restrict the domain of single variables, the length and the breadth of the room. Binary constraints like "The desk should be placed in front of the window", ternary constraints like "The table should be placed between the sofa and the armchair", or in general n-ary constraints restrict the domain of 2, 3, or n variables. Problems like these can be formalised as constraint satisfaction problems.

Given a set of m variables $\mathcal{V} = \{x_1, \ldots, x_m\}$ over a domain $\mathcal{D}$, an *n-ary constraint* consists of an n-ary relation $R_i \subseteq \mathcal{D}^n$ and an n-tuple of variables $\langle x_{i_1}, \ldots, x_{i_n}\rangle$, written $R_i(x_{i_1}, \ldots, x_{i_n})$. For binary constraints, we will also use the infix notation $x_1 R_i x_2$. A (partial) *instantiation* f of variables to values is a (partial) function from the set of variables $\mathcal{V}$ to the set of values $\mathcal{D}$. We say that an instantiation f *satisfies the constraint* $R_i(x_{i_1}, \ldots, x_{i_n})$ if and only if $\langle f(x_{i_1}), \ldots, f(x_{i_n})\rangle \in R_i$.

A *constraint satisfaction problem* (CSP) consists of a set of variables $\mathcal{V}$ over a domain $\mathcal{D}$ and a set of constraints Θ. The intention is to find a *solution* which is an instantiation such that all constraints in Θ are satisfied.

In this work we restrict ourselves to binary CSPs, i.e., CSPs where only binary constraints are used. A binary CSP can be represented by a *constraint network* which is a labelled digraph where each node is labelled by a variable x_i or by the variable index i and each directed edge is labelled by a binary relation. We will use the notation R_{ij} to denote the relation constraining the variable pair $\langle x_i, x_j\rangle$. By overloading notation, we also use R_{ij} to denote the constraint $R_{ij}(x_i, x_j)$ itself.

A CSP is *consistent* if it has a *solution*. If the domain of the variables is finite, CSPs can be solved by *backtracking* over the ordered domains of the single variables. Backtracking works by successively instantiating variables with values of the ordered domain until either all variables are instantiated and a solution is found or an inconsistency is detected in which case the current variable is instantiated with the next value of its domain. If all possible instantiations of the current variable lead to an inconsistency, the previous variable becomes the current variable and the process is repeated. Backtracking is in general exponential in the number of variables. The process can be sped up by propagating constraints between the variables and eliminating impossible values as soon as possible. If the domain of the variables is infinite, backtracking over the domain is not possible and other methods have to be applied.

2.1 Binary Constraint Satisfaction Problems and Relation Algebras

One way of dealing with infinite domains is using constraints over a finite set of binary relations. Ladkin and Maddux [Ladkin and Maddux, 1994] proposed to employ *relation algebras* developed by Tarski [Tarski, 1941] for this purpose. A relation algebra consists of a set of binary relations $\mathcal{R}$ which is closed under several operations on relations and contains some particular relations. The operations are *union* ($\cup$), *intersection* ($\cap$), *composition* ($\circ$), *complement* ($\bar{\cdot}$), and *conversion* ($\cdot^{\smile}$), where conversion and composition are defined as follows:

$$R \circ S \overset{def}{=} \{\langle x, y\rangle \mid \exists z \colon \langle x, z\rangle \in R \wedge \langle z, y\rangle \in S\} \qquad (1.1)$$

$$R^{\smile} \overset{def}{=} \{\langle x, y\rangle \mid \langle y, x\rangle \in R\} \qquad (1.2)$$

In the following, we will—by abusing notation—identify sets of relations with their union. For example, we identify $\{R, S, T\}$ with $R \cup S \cup T$.

The particular binary relations mentioned above are the *empty relation* $\emptyset$ which does not contain any pair, the *universal relation* $*$ which contains all possible pairs, and the *identity relation* Id which contains all pairs of identical elements. inxxset of constraints

We assume that a set of constraints Θ contains one constraint for each pair of variables involved in Θ, i.e., if no information is given about the relation holding between two variables x_i and x_j, then the universal relation $*$ constrains the pair, i.e., $R_{ij} = *$. Another assumption that we make is that whenever a constraint R_{ij} between x_i and x_j is in Θ, the converse relation constrains x_j and x_i, i.e., $(R_{ij})^{\smile} = R_{ji}$.

Determining consistency for CSPs with infinite domains is in general undecidable [Hirsch, 1999]. A partial method for determining inconsistency of a CSP is the *path-consistency method* which enforces path-consistency of a CSP [Montanari, 1974; Mackworth, 1977]. A CSP is *path-consistent* if and only if for any partial instantiation of any two variables satisfying the constraints between the two variables, it is possible for any third variable to extend the partial instantiation to this third variable satisfying the constraints between the three variables.

A straight-forward way to enforce path-consistency on a binary CSP is to strengthen relations by successively applying the following operation until a fixed point is reached:

$$\forall k : R_{ij} := R_{ij} \cap (R_{ik} \circ R_{kj}).$$

The resulting CSP is *equivalent* to the original CSP in the sense that it has the same set of solutions. If the empty relation results while performing this operation, we know that the CSP is inconsistent. Otherwise, the CSP might or might not be consistent. Provided that the composition of relations can be computed in constant time, the algorithm sketched has a running time of $O(n^5)$, where n is the total number of nodes in the graph. More advanced algorithms can enforce path-consistency in time $O(n^3)$ [Mackworth and Freuder, 1985]. Figure 1.1 shows the $O(n^3)$ time path-consistency algorithm by van Beek [van Beek, 1992] which uses a queue to keep track of those triples of variables that might be affected by the changes made and which have to be analysed again.

2.2 Relation Algebras based on JEPD Relations

Of particular interest are relation algebras that are based on finite sets of *jointly exhaustive and pairwise disjoint* (JEPD) relations. JEPD relations are sometimes called *atomic*, *basic*, or *base relations.* We refer to them as basic relations. Since any two entities are related by exactly one of the basic relations, they can be used to represent definite knowledge with respect to the given level of granularity. Indefinite

Algorithm: PATH-CONSISTENCY
Input: A set Θ of binary constraints over the variables $x_1, x_2, \ldots, x_n$
Output: path-consistent set equivalent to Θ, or *fail*, if inconsistency is detected

```
1. Q ← $\{(i,j,k),(k,i,j) \mid i<j, k \neq i, k \neq j\}$;
        ($i$ indicates the $i$-th variable of $\Theta$. Analogously for $j$ and $k$)
2. while Q $\neq \emptyset$ do
3. select and delete a path $(i,k,j)$ from Q;
4.      if REVISE$(i,k,j)$ then
5.          if $R_{ij} = \emptyset$ then return fail
6.              else Q ← Q $\cup\{(i,j,k),(k,i,j) \mid k \neq i, k \neq j\}$;
```

Function: REVISE(i,k,j)
Input: three variables i, k and j
Output: true, if R_{ij} is revised; false otherwise.
Side effects: R_{ij} and R_{ji} revised using the operations $\cap$ and $\circ$ over the constraints involving i, k, and j.

```
1. oldR := $R_{ij}$;
2. $R_{ij} := R_{ij} \cap (R_{ik} \circ R_{kj})$;
3. if (oldR = $R_{ij}$) then return false;
4. $R_{ji} := R_{ij}^{\smile}$;
5. return true.
```

Figure 1.1. Van Beek's PATH-CONSISTENCY algorithm.

knowledge can be specified by unions of possible basic relations. In this chapter we denote a set of basic relations with $\mathcal{B}$ and it should be clear from the context which particular set of basic relations we are referring to. If the set of relations formed by generating all unions over these basic relations is closed under composition and converse, then this set of relations is the carrier of a relation algebra. We denote the set of all relations by $2^{\mathcal{B}}$ alluding to the fact that we identify sets of relations with their unions.

For these relation algebras, the universal relation is the union over all basic relations. Converse, complement, intersection and union of relations can easily be obtained by performing the corresponding set theoretic operations. Composition of basic relations has to be computed using the semantics of the relations. Composition of unions of basic relations can be obtained by computing the union of the composition

Table 1.1. The thirteen basic relations of Allen's interval algebra

Interval Base Relation	Symbol	Pictorial Example	Endpoint Relations
x before y	$\prec$	xxx	$X^- < Y^-$, $X^- < Y^+$,
y after x	$\succ$	yyy	$X^+ < Y^-$, $X^+ < Y^+$
x meets y	m	xxxx	$X^- < Y^-$, $X^- < Y^+$,
y met-by x	$\mathsf{m}^{\smile}$	yyyy	$X^+ = Y^-$, $X^+ < Y^+$
x overlaps y	o	xxxx	$X^- < Y^-$ $X^- < Y^+$,
y overlapped-by x	$\mathsf{o}^{\smile}$	yyyy	$X^+ > Y^-$, $X^+ < Y^+$
x during y	d	xxx	$X^- > Y^-$, $X^- < Y^+$,
y includes x	$\mathsf{d}^{\smile}$	yyyyyyyy	$X^+ > Y^-$, $X^+ < Y^+$
x starts y	s	xxx	$X^- = Y^-$, $X^- < Y^+$,
y started-by x	$\mathsf{s}^{\smile}$	yyyyyyyy	$X^+ > Y^-$, $X^+ < Y^+$
x finishes y	f	xxx	$X^- > Y^-$, $X^- < Y^+$,
y finished-by x	$\mathsf{f}^{\smile}$	yyyyyyyy	$X^+ > Y^-$, $X^+ = Y^+$
x equals y	$\equiv$	xxxx yyyy	$X^- = Y^-$, $X^- < Y^+$, $X^+ > Y^-$, $X^+ = Y^+$

of the basic relations. Usually, compositions of the basic relations are pre-computed and stored in a *composition table*.

The best known example of such a relation algebra is the *Interval Algebra* introduced by Allen [Allen, 1983] which defines 13 different basic relations between convex intervals on a directed line. The basic relations and a graphical depiction are given in Table 1.1. Even though the interval algebra was introduced for temporal representation and reasoning, there is a number of spatial calculi which are derived from the interval algebra. Some of them we will mention in this chapter.

We say that a relation R is a *refinement* of a relation S if and only if $R \subseteq S$. Given, for instance, a union of relations $\{R_1, R_2, R_3\}$, then the relation $\{R_1, R_2\}$ is a refinement of the former relation. This definition carries over to constraints and to sets of constraints. Then, a set of constraints Θ' is a refinement of Θ if and only if both CSPs have the same variables and for all relations R'_{ij} constraining the pair x_i, x_j in Θ' and all relations R_{ij} constraining the same variables in Θ, we have $R'_{ij} \subseteq R_{ij}$. Θ' is said to be a *consistent refinement* of Θ if and only if Θ' is a refinement of Θ and both Θ and Θ' are consistent. A *consistent scenario* Θ_s of a set of constraints Θ is a consistent refinement of Θ where all the constraints of Θ_s are assertions of basic relations.

This chapter deals with determining consistency of binary constraint satisfaction problems that are based on JEPD relations. Let $\mathcal{B}$ be a finite set of JEPD binary relations. The *consistency problem* $\mathsf{CSPSAT}(\mathcal{S})$ for sets $\mathcal{S} \subseteq 2^{\mathcal{B}}$ over a (possibly infinite) domain $\mathcal{D}$ is defined as follows:

Instance: A set $\mathcal{V}$ of variables over a domain $\mathcal{D}$ and a finite set Θ of binary constraints $R(x_i, x_j)$, where $R \in \mathcal{S}$ and $x_i, x_j \in \mathcal{V}$.

Question: Is there an instantiation of all variables in Θ with values from $\mathcal{D}$ such that all constraints are satisfied?

In the general case CSPSAT is undecidable [Hirsch, 1999], but in many interesting cases it is possible to prove decidability or even tractability of $\mathsf{CSPSAT}(\mathcal{S})$.

If CSPSAT is decidable for a certain subset $\mathcal{S} \subseteq 2^{\mathcal{B}}$ then it is possible to decide CSPSAT for other subsets of $2^{\mathcal{B}}$ by using a non-deterministic algorithm. This is done by selecting refinements of the relations such that the refinements are contained in $\mathcal{S}$. For example, suppose $\mathcal{S} \subseteq 2^{\mathcal{B}}$ contains the relations $S_1, \ldots, S_n$, the relation $R \in 2^{\mathcal{B}}$ is not contained in $\mathcal{S}$, and $R = S_1 \cup S_3 \cup S_4$. Then, the constraint $R(x, y)$ can be processed by guessing non-deterministically one of the relations S_1, S_3, and S_4. In general, a subset $\mathcal{S} \subseteq 2^{\mathcal{B}}$ *splits* another subset $\mathcal{T} \subseteq 2^{\mathcal{B}}$ *exhaustively* if for every relation T of $\mathcal{T}$ there are refinements $S_1, \ldots, S_k \in \mathcal{S}$ such that $T = S_1 \cup \ldots \cup S_k$. If $\mathcal{S}$ splits $\mathcal{T}$ exhaustively, it is obvious that decidability of $\mathsf{CSPSAT}(\mathcal{S})$ implies decidability of $\mathsf{CSPSAT}(\mathcal{T})$. Furthermore, it implies that $\mathsf{CSPSAT}(\mathcal{T})$ can be decided in polynomial time on a non-deterministic Turing machine, if $\mathsf{CSPSAT}(\mathcal{S})$ can be decided in polynomial time.

The non-deterministic algorithm sketched above can be turned into a deterministic one by employing a backtracking scheme. The backtracking algorithm given in Figure 1.2 is a generalisation of the one proposed by Ladkin and Reinefeld [Ladkin and Reinefeld, 1992] and relies on a set $\mathcal{S}$ that splits $2^{\mathcal{B}}$ exhaustively. Note that a set $\mathcal{S}$ splits $2^{\mathcal{B}}$ exhaustively if and only if $\mathcal{S}$ contains all base relations. $\mathcal{S}$ is called the *split set*. The backtracking algorithm uses a function DECIDE which is a sound and complete decision procedure for $\mathsf{CSPSAT}(\mathcal{S})$. The (optional) procedure PATH-CONSISTENCY in line 1 is used as forward-checking and restricts the remaining search space. Nebel [Nebel, 1997] showed that this restriction preserves soundness and completeness of the algorithm – provided the split set is closed under intersection, composition, and converse. If the decision procedure DECIDE runs in polynomial time, CONSISTENCY is exponential in the number of constraints of Θ. If enforcing path-consistency is sufficient for deciding $\mathsf{CSPSAT}(\mathcal{S})$, DECIDE($\Theta$) in line 4 is not necessary and one can return `true` at this point.

The efficiency of the backtracking algorithm depends on several factors. One of them is, of course, the size of the search space which has to

Algorithm: CONSISTENCY
Input: A set Θ of binary constraints over the variables $x_1, x_2, \ldots, x_n$ and a subset $\mathcal{S} \subseteq 2^{\mathcal{B}}$ that splits $2^{\mathcal{B}}$ exhaustively and for which $\mathsf{CSPSAT}(\mathcal{S})$ is decidable.
Output: `true`, iff Θ is consistent.

```
1. PATH-CONSISTENCY($\Theta$)
2. if $\Theta$ contains the empty relation then return false
3. else choose an unprocessed constraint $R(x, y)$ and
        split $R$ into $S_1, \ldots, S_k \in \mathcal{S}$ such that $S_1 \cup \ldots \cup S_k = R$
4. if no constraint can be split then return DECIDE($\Theta$)
5. for all refinements $S_l$ $(1 \leq l \leq k)$ do
6.      replace $R(x, y)$ with $S_l(x, y)$ in $\Theta$
7.      if CONSISTENCY($\Theta$) then return true
```

Figure 1.2. Backtracking algorithm for deciding consistency.

be explored. A common way of measuring the size of the search space is the average *branching factor*, i.e., the average number of branches each node in the search space has. For the backtracking algorithm described in Figure 1.2 this depends on the average number of relations of the split set $\mathcal{S}$ into which a relation has to be split. The less splits in average the better, i.e., it is to be expected that the efficiency of the backtracking algorithm depends on the split set $\mathcal{S}$ and its branching factor. Another factor is how the search space is explored. The backtracking algorithm of Figure 1.2 offers two possibilities of applying heuristics. One is in line 3 where the next unprocessed constraint can be chosen, the other is in line 5 where the next refinement can be chosen. These two choices influence the search space and the path through the search space. Good choices should increase efficiency of the backtracking algorithm.

Other fundamental reasoning problems are the *minimal label problem* CSPMIN, the problem of finding the strongest entailed relation for each pair of variables from a given set of constraints, and the *entailment problem* CSPENT, i.e., decide whether a particular constraint is entailed by a set of constraints. As was shown for the corresponding temporal problems (see the next section), the entailment problem, the minimal label problem, and the consistency problem are equivalent under polynomial Turing reductions [Vilain et al., 1989; Golumbic and Shamir, 1993].

3. Spatial Constraint Calculi

In qualitative spatial reasoning it is common to consider a particular aspect of space such as topology, direction, or distance and to develop a system of qualitative relationships between spatial entities which cover this aspect of space to some degree and which appear to be useful from an applicational or from a cognitive perspective. If these relations are based on a set of jointly exhaustive and pairwise disjoint basic relations which is closed under several operations, it is possible to apply constraint based methods for reasoning over these relations (see Section 2). For this it is only necessary to give a composition table; either for all relations or for the basic relations plus a procedure for computing the compositions of complex relations.

The composition table should be obtained using the formal semantics of the relations. Otherwise it is not possible to verify correctness and completeness of the inferences. Formal semantics of the relations are also necessary for finding efficient reasoning algorithms which are essential for most applications. Without formal semantics it is sometimes not even possible to show that reasoning over a system of relations is decidable (e.g., the 9-intersection relations [Egenhofer, 1991] as shown by Grigni et al. [Grigni et al., 1995]).

In the following subsections we survey some important approaches to the main aspects of space, topology, direction, and distance. Instead of summarising many different approaches, we focus on those approaches which have been formally analysed.

3.1 Topology

Topological distinctions between spatial entities are a fundamental aspect of spatial knowledge. Topological distinctions are inherently qualitative which makes them particularly interesting for qualitative spatial reasoning. Although there is a large body of work on topology developed in mathematics, this is not very well suited for qualitative spatial reasoning. Mathematical research on topology is not concerned with reasoning over topological relationships and as such does not provide us with any reasonable topological calculi and reasoning mechanisms [Gotts et al., 1996].

Topological approaches to qualitative spatial reasoning usually describe relationships between spatial regions rather than points, where spatial regions are subsets of some topological space. Most existing approaches on formalising topological properties of spatial regions are based on work from Whitehead [Whitehead, 1929] and Clarke [Clarke, 1981; Clarke, 1985] who axiomatised mereotopology using a single prim-

Table 1.2. Topological interpretation of the eight base relations of RCC-8. All spatial regions are regular closed, i.e., $x = c(i(x))$ and $y = c(i(y))$. $i(\cdot)$ specifies the topological interior of a spatial region, $c(\cdot)$ the topological closure.

RCC-8 *Relation*	*Topological Constraints*
$\langle x, y\rangle \in \mathsf{DC}$	$x \cap y = \emptyset$
$\langle x, y\rangle \in \mathsf{EC}$	$i(x) \cap i(y) = \emptyset$, $x \cap y \neq \emptyset$
$\langle x, y\rangle \in \mathsf{PO}$	$i(x) \cap i(y) \neq \emptyset$, $x \not\subseteq y$, $y \not\subseteq x$
$\langle x, y\rangle \in \mathsf{TPP}$	$x \subset y$, $x \not\subseteq i(y)$
$\langle x, y\rangle \in \mathsf{TPP}^{-1}$	$y \subset x$, $y \not\subseteq i(x)$
$\langle x, y\rangle \in \mathsf{NTPP}$	$x \subset i(y)$
$\langle x, y\rangle \in \mathsf{NTPP}^{-1}$	$y \subset i(x)$
$\langle x, y\rangle \in \mathsf{EQ}$	$x = y$

itive relation, the binary connectedness relation. Some approaches also distinguish between a mereological primitive, the parthood relation, and a topological primitive, the connected relation [Borgo et al., 1996].

Using these primitive relations it is possible to define many other relations. A set of jointly exhaustive and pairwise disjoint relations which can be defined in all approaches of this kind are the eight relations $\mathsf{DC}, \mathsf{EC}, \mathsf{PO}, \mathsf{EQ}, \mathsf{TPP}, \mathsf{NTPP}, \mathsf{TPP}^{-1}, \mathsf{NTPP}^{-1}$. In the best known approach in this domain, the Region Connection Calculus by Randell, Cui, and Cohn [Randell et al., 1992], these relations are known as the RCC-8 relations. In Table 1.2 we defined the RCC-8 relations using the interior and exterior of spatial regions. Sample instances of the relations are given in Figure 1.3. The relation symbols are abbreviations of their meanings: DisConnected, Externally Connected, Partially Overlapping, EQual, Tangential Proper Part, Non-Tangential Proper Part and the converse relations of the latter two relations.

What distinguishes the different approaches and what thereby influences the definable relations is the interpretation of the connectedness relation and the properties of the considered regions. Some approaches distinguish between open and closed regions [Randell and Cohn, 1989; Asher and Vieu, 1995] which allows, for instance, to define different kinds of contact. Asher and Vieu [Asher and Vieu, 1995] distinguished between strong contact (two regions have points in common) and weak contact (two regions are disjoint but their topological closures share common points). Other approaches do not make this distinction and treat regions which are open, closed, or neither equally [Randell et al., 1992]. The Region Connection Calculus [Randell et al., 1992] considers only the topological closure of regions: two regions are connected if their topological closures share a common point. Cohn et

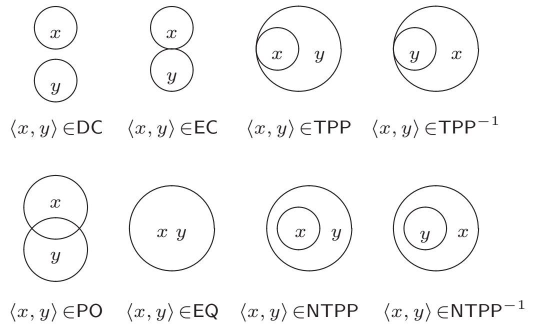
x
y
$\langle x, y\rangle \in$DC
x
y
$\langle x, y\rangle \in$EC
x y
$\langle x, y\rangle \in$TPP
y x
$\langle x, y\rangle \in \mathsf{TPP}^{-1}$
x
y
$\langle x, y\rangle \in$PO
x y
$\langle x, y\rangle \in$EQ
x y
$\langle x, y\rangle \in$NTPP
y x
$\langle x, y\rangle \in \mathsf{NTPP}^{-1}$

Figure 1.3. Two-dimensional examples for the eight basic relations of RCC-8.

al. [Cohn et al., 1997] argue that this definition is more appropriate for commonsense spatial reasoning since there is “no reason to believe that some physical objects occupy closed regions and others open”. Orthogonal to the interpretation of the connectedness relation is the distinction of what regions are considered. A very common restriction is to use only non-empty regular regions. As shown by Asher and Vieu [Asher and Vieu, 1995], models based on Clarke’s connectedness relation require all regions to be nonempty and regular. However, it is possible to specify additional properties of regions such as dimensionality, internal connectedness, i.e., whether a region consists of one-piece or of multiple pieces, or the existence of holes. The different approaches are compared in [Cohn and Varzi, 1998; Cohn and Varzi, 1999]. In particular, the RCC-8 constraint language uses non-empty, regular closed regions which are subsets of a regular connected topological space. Regions do not have to be internally connected and are allowed to have holes.

All of these approaches have in common that the relations are axiomatised and defined in first-order logic which provides them with formal semantics. The formal properties of first-order theories based on a connectedness relation were studied by Grzegorczyk [Grzegorczyk, 1951], Dornheim [Dornheim, 1998] and Pratt and Schoop [Pratt and Schoop, 1998; Schoop, 1999]. These are very expressive approaches and lead easily to undecidability of formal reasoning, i.e., logical implication in these theories is not decidable in general. When constraining oneself to less expressive languages such as constraint calculi, the computational costs are, of course, less. Constraint calculi can be regarded as the special case of first-order sentences where only existentially quantified region variables are used. Using the RCC-8 relations, we can state constraints such as, for example, $\mathsf{DC}(x_1, x_2), \mathsf{TPP}^{-1}(x_2, x_3), \{\mathsf{EC}, \mathsf{PO}\}(x_3, x_1)$. These are interpreted over the domain of regular closed regions of any regular topological space, such as, e.g., the n-dimensional Euclidean space. Now, given the composition table of RCC-8, we can use the algorithms introduced in the previous section in order to decide whether the set of constraint is consistent, or whether other constraints are logically implied. A prerequisite, however, is that a method for deciding consistency for some subset of the relation system formed by the the RCC-8 relations can be found. Bennett [Bennett, 1994] gave an encoding of the RCC-8 relations in propositional modal logic and, thus proved that reasoning over the RCC-8 constraint language is decidable. In fact, this technique can be used as a decision method for RCC-8 constraint systems.

A different approach to defining topological relations was given by Egenhofer [Egenhofer, 1991] in the area of spatial information systems. Egenhofer defined binary relations according to the 9 different intersections of the interior, exterior, and boundary of regions, hence, called *9-intersection*. Depending on the regions that are used, many different relations can be defined in this way [Egenhofer et al., 1994; Egenhofer and Franzosa, 1994]. If only two-dimensional, internally connected regular regions without holes are considered and only emptiness or nonemptiness of the intersection is taken into account, this results in the same set of eight basic relations as definable in the above described approaches. In contrast to the connection based approaches, this approach is not provided with formal semantics which makes it very difficult to study its formal properties. For instance, attempts were made to identify sound and complete algorithms for reasoning over the eight relations defined by Egenhofer [Smith and Park, 1992; Egenhofer and Sharma, 1993] while it was taken for granted that path-consistency decides consistency if only basic relations are used. As shown by [Grigni et al., 1995], this is not the case for Egenhofer's definition of the eight topological relations.

3.2 Orientation

Orientation is, like topology, very well suited for a qualitative approach. In everyday (non-technical) communication, orientation of spatial entities with respect to other spatial entities is usually given in terms of a qualitative category like "to the left of" or "northeast of" rather than using a numerical expression like "53 degrees" (which is certainly more common in technical communication like in aviation). Unlike the topological approaches we discussed in the previous section, orientation of spatial entities is a ternary relationship depending on the located object, the reference object, and the *frame of reference* which can be

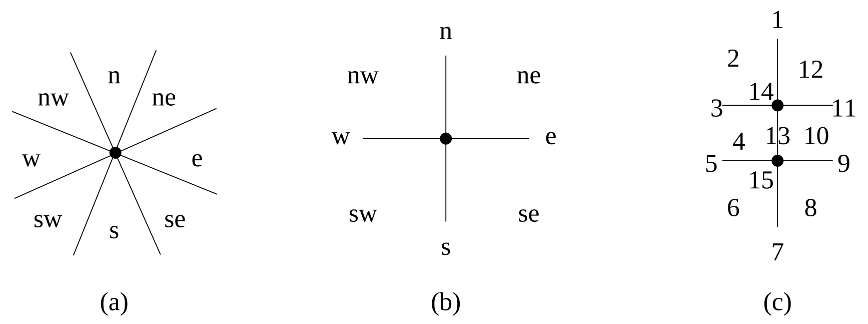

Figure 1.4. Orientation relations between points: (a) cone-based (b) projection-based (c) double-cross.

specified either by a third object or by a given direction. In the literature one distinguishes between three different kinds of frames of reference, extrinsic ("external factors impose an orientation on the reference object"), intrinsic ("the orientation is given by some inherent property of the reference object"), and deictic ("the orientation is imposed by the point of view from which the reference object is seen") [Hernàndez, 1994, p.45]. If the frame of reference is given, orientation can be expressed in terms of binary relationships with respect to the given frame of reference.

Most approaches to qualitatively dealing with orientation are based on points as the basic spatial entities and consider only two-dimensional space. Frank [Frank, 1991] suggested different methods for describing the cardinal direction of a point with respect to a reference point in a geographic space, i.e., directions are in the form of "north", "east", "south", and "west" depending on the granularity. These are, however, just labels which can be equally termed as, for instance, "front", "right", "back", and "left" in a local space. Frank distinguishes between two different methods for determining the different sectors corresponding to the single directions: the *cone-based method* and the *projection-based method* (see Figure 1.4). The projection-based approach allows us to represent the nine different relations (n, ne, e, se, s, sw, w, nw, eq) in terms of the point algebra by specifying a point algebraic relation for each of the two axes separately. This provides the projection-based approach (which is also called the *cardinal algebra* [Ligozat, 1998]) with formal semantics which were used by Ligozat [Ligozat, 1998] to study its computational properties. In particular, Ligozat found that reasoning with the cardinal algebra is NP-complete (See below in Section 4) and, further, identified a maximal tractable subset of the cardinal algebra by

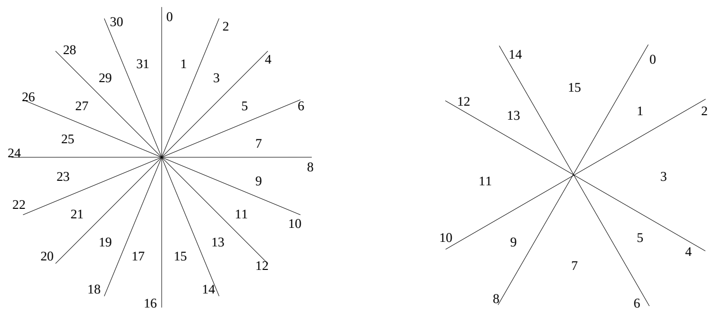

Figure 1.5. Two different Star calculi, one with 8 lines forming 33 relations and one with 4 lines forming 17 relations.

using the concept of *preconvex* relations, a method which has already been used for Allen's interval algebra [Ligozat, 1996].

A generalisation of these calculi was proposed and analysed by Renz and Mitra [Renz and Mitra, 2004]. Their calculus, the *Star calculus* (see Figure 1.5), is based on a number of n lines l_i with given angles δ_i (for arbitrary n) which define $2n$ sectors and $4n + 1$ basic relations. The number of lines and the angles of the sectors can be adopted to the given application, so the Star calculus can be used for representing and reasoning about qualitative directions of arbitrary granularity. The Star calculus has some interesting properties. For example, it can be shown that when having three or more lines it is possible to emulate a coordinate system which is due to having the lines as separate basic relations. This removes the distinction made between qualitative and quantitative representation and also means that qualitative reasoning methods like path-consistency cannot be complete for deciding consistency. Renz and Mitra therefore proposed to combine the lines and the sectors and to consider them as new basic relations. In both cases the full calculus is NP-hard while reasoning over the basic relations is tractable.

A further point-based approach was developed by Freksa [Freksa, 1992], the so-called *double-cross* calculus, which defines the direction of a located point to a reference point with respect to a perspective point. Within this approach three axes are used: one is specified by the perspective point and the reference point, the other two axes are orthogonal to the first one and are specified by the reference point and the perspective point. These axes define 15 different ternary basic relations (see Figure 1.4c). The computational properties of this calculus have been studied by Scivos and Nebel [?]. It turned out that the consistency problem is NP-hard even if only the 15 basic relations are used.

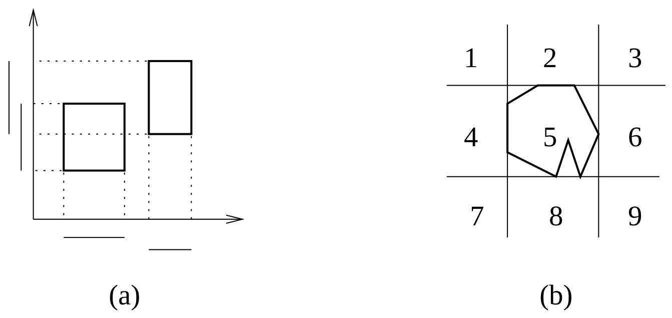

Figure 1.6. Orientation relations between extended entities: (a) rectangle algebra (b) direction-relation matrix.

Developing orientation relations between extended spatial entities is much more difficult than between points. Extended objects often have their own intrinsic directions like a natural front. Also, the direction between extended objects with complex shapes such that, for instance, their convex hulls intersect is not at all clear. Even for simpler objects there is often no agreement about which natural language expression describes their orientational relationship best. Therefore, most approaches use approximations to the spatial regions or use only spatial regions of a particular kind. One approach which has been chosen by many researchers results in restricting all regions to be rectangles whose sides are parallel to the axes determined by the frame of reference [Guesgen, 1989; Papadias and Theodoridis, 1997; Balbiani et al., 1998]. In this approach all regions can be represented by their projections to the defining axes which corresponds to having Allen's interval algebra [Allen, 1983] for each axis separately, i.e., every relation is a pair of two interval relations (see Figure 1.6a). For two-dimensional space this results in 13×13 different basic relations (also called the *rectangle algebra* [Balbiani et al., 1998]) whose formal semantics are provided by the interval algebra.

Balbiani et al. [Balbiani et al., 1998; Balbiani et al., 1999a] studied the formal properties of the rectangle algebra (and also of the '*block algebra* which is the n-dimensional extension of the interval algebra [Balbiani et al., 1999b]). NP-completeness of the algebra carries over from the interval algebra. Balbiani et al. [Balbiani et al., 1998; Balbiani et al., 1999a] identified a tractable subset of the rectangle algebra following a line of reasoning which has been introduced by Ligozat [Ligozat, 1996], namely, by considering convex and preconvex relations. Unlike for the interval algebra, the set of preconvex relations is not closed under the fundamental operations. Thus, Balbiani et al. extended the concept of preconvexity by distinguishing between weakly and strongly preconvex relations, and show that the set of strongly preconvex relations is a tractable subset of the rectangle algebra for which path-consistency is sufficient for deciding consistency. In fact the rectangle algebra represents more than just orientation between two rectangles but also their topological relations. Hence, the rectangle algebra can be regarded as an approach to combining topology and orientation. However, because of the large number of relations of the rectangle algebra (a total number of 2^{169} relations) reasoning even over a tractable subset can be very inefficient.

An interesting but less expressive approach to representing orientational relationships between extended spatial entities was introduced by Goyal and Egenhofer [Goyal and Egenhofer, pear]. Their calculus consists of a 3×3 *direction-relation matrix* which represents the 9 sectors formed by the minimal bounding axes of an extended spatial entity (see Figure 1.6b). For each sector it is possible to specify whether the located object is contained in the sector or not, or (non-qualitatively) to which degree the located object is contained in the sector.

Skiadopolous and Koubarakis [Skiadopoulos and Koubarakis, 2005] developed reasoning algorithms for this calculus and analysed its computational properties, but their algorithms are not based on constraint based methods like path-consistency.

3.3 Distance

Together with topology and orientation, distance is one of the most important aspects of space. Unlike the other two, distance is a scalar entity. Dealing with distance information is an important cognitive ability in our everyday life. In order to grab something, for instance, we must be good in judging distances. When communicating about distances, we usually use qualitative categories like “A is close to B” or qualitative distance comparatives like “A is closer to B than to C”, but also numerical values like “A is about one meter away from B”. As indicated by the above examples, one can distinguish between absolute distance relations (the distance between two spatial entities) and relative distance relations (the distance between two spatial entities as compared to the distance to a third entity). While absolute distance can be represented either qualitatively or quantitatively, relative distance is purely qualitative. When representing absolute distance in a qualitative way, this also depends on the scale of space which is used. Montello [Montello, 1993]

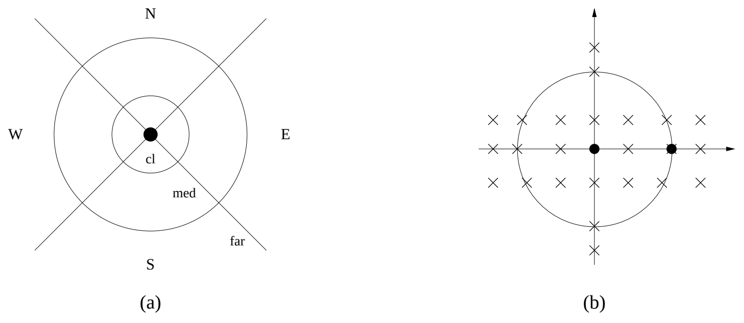

Figure 1.7. Different approaches to representing positional information: (a) absolute distances combined with cone-based orientation [Clementini et al., 1997] (b) relative distances combined with projection-based orientation. [Isli and Moratz, 1999]

suggests four different kinds of scales of space: figural space, vista space, environmental space, and geographic space.

Most approaches to qualitative distance consider points as the basic entities. Absolute distance relations are obtained, e.g., by dividing the real line into several sectors such as "very close", "close", "commensurate", "far", and "very far" depending on the chosen level of granularity [Hernàndez et al., 1995]. Relative distance can be obtained by comparing the distance to a given reference distance which results in ternary relations such as "closer than", "equidistant", or "farther than". Reasoning about qualitative distances leads to several difficulties. For instance, given a sequence of collinear points $p_1, \ldots, p_n$ such that p_i is close to p_{i+1} for every i, for which n is p_n far from p_1? Moreover, combining distance relations does not only depend on the distances itself but also on the position of the corresponding points. For instance if point B is far from A and C is far from B, then C can be very far from A if A, B, and C are aligned and if B is between A and C; or C can be close to A if the angle between AB and BC is small. Therefore, it seems advisable to study distance in combination with orientation. This combination is called *positional* information.

One approach for developing a position calculus is by Clementini et al. [Clementini et al., 1997] who combine a cone-based orientation approach with absolute distance relations (see Figure 1.7a). Clementini et al. present different procedures for computing the composition of two positional relations (A, B) and (B, C). They consider three special cases where BC is the same, opposite, or orthogonal direction to AB. Another approach is by Isli et al. [Isli and Moratz, 1999] who propose several position calculi on various levels of granularity by combining relative distance relations with different approaches to orientation such as the projection-based approach (see Figure 1.7b) or the double-cross calculus. The computational properties of these approaches have not been studied yet.

4. Computational complexity

Since we are interested in automated reasoning with the spatial calculi described above, it is a good idea to get an understanding of how computationally demanding reasoning in these calculi is. Here computational complexity theory is the right theoretical tool.

In the field of computational complexity [Papadimitriou, 1994], computational problems are classified according to their need for resources for solving them, usually the running time and the memory consumption. This allows to compare the complexity of different problems and to design algorithms for a whole class of problems. For classifying computational problems, they are usually expressed as *decision problems*, i.e., problems that require a simple yes/no answer. Such problems can be equivalently viewed as formal languages over some alphabet Σ, which is formed by all yes-instances.

Most problems can be easily translated into an equivalent decision problem. Assume for example the problem of finding a satisfying truth assignment for a propositional formula. The corresponding decision problem is the problem SAT: given a set of variables V and a propositional formula ϕ over V in CNF, is there a satisfying truth assignment for ϕ? The complexity of a decision problem is usually measured according to the worst-case running time or memory consumption of the best possible algorithm. If we now can prove lower bounds on the runtime for the decision problem then these lower bounds apply obviously to the original problem as well.

Running time as well as memory consumption of an algorithm depends on the size n of its input, i.e., on the size of the problem instance, and can be expressed as a function $f(n)$. For classifying algorithms according to their running time, the asymptotical behaviour is more important than f itself. This is specified in terms of the *O-notation* which gives an upper bound on the running time within a constant factor [Cormen et al., 1990]. An algorithm with a running time of $O(n^3)$ or faster is usually considered to be efficient. In areas like database systems where instances have a very large size, a running time of $O(n^3)$ is too slow. In these areas efficient algorithms should have a linear running time.

4.1 Tractability and NP-completeness

There is a large number of different complexity classes that are used to categorise decision problems [Johnson, 1990]. Particularly important is the class of decision problems that can be solved in polynomial time using a deterministic algorithm. This complexity class is called P and it is considered to be the class of efficiently solvable problems. Problems in P are also called *tractable* problems, problems outside P are called *intractable* problems.

Interestingly, there exists a large class of problems for which nobody has found polynomial-time algorithms yet, but it appears equally hard to prove that no such algorithms exist. In order to capture these problems, one extends the notion of algorithm. The class of problems solvable in polynomial time using a non-deterministic algorithm is called NP, which is equivalent to specifying that a given solution of an NP problem can be verified in polynomial time using a deterministic algorithm. In the sequel, an algorithm will always be a deterministic algorithm, unless otherwise stated. It is clear that P is a subset of NP, but it is not known whether P is a proper subset of NP or whether P is equal to NP, which is called the $\mathsf{P} \overset{?}{=} \mathsf{NP}$ problem.

An important method of comparing problems is specifying a *reduction* from one problem to another. Given two problems $A, B \subseteq \Sigma^*$, problem A can be *reduced* to problem B by giving a constructive transformation $f \colon \Sigma^* \to \Sigma^*$ such that $f(x) \in B$ if and only if $x \in A$. If f can be computed in polynomial time, the reduction is a *polynomial (time) reduction*. If A is polynomially reducible to B (written as $A \leq_p B$), then any polynomial time algorithm for solving B can be used to solve A. Thus, for showing that a particular decision problem A is in P, it is sufficient to find another problem $B \in \mathsf{P}$ such that $A \leq_p B$.

A decision problem A is said to be NP-*hard* if any other problem in NP can be polynomially reduced to A. An NP-hard problem which is itself contained in NP is called NP-*complete*. NP-complete problems are the most difficult problems in NP. In fact, most of the problems for which nobody has found an efficient algorithms yet but which are resistant against proving them to be intractable fall into this class. In order to prove a decision problem A to be NP-hard, it is sufficient to find another NP-hard problem that can be polynomially reduced to A. The first problem that was identified to be NP-complete is the SAT problem [Cook, 1971]. In this work we use the following NP-complete propositional decision problems [Garey and Johnson, 1979]:

Given: A set of variables V and a propositional formula ϕ over V in CNF such that each clause of ϕ has exactly three literals.

Questions:

1 Is there a satisfying truth assignment for ϕ? (3SAT)

2 Is there a satisfying truth assignment for ϕ such that each clause has at least one true literal and at least one false literal? (NOT-ALL-EQUAL-3SAT)

3 Is there a satisfying truth assignment for ϕ such that each clause has exactly one true literal? (ONE-IN-THREE-3SAT)

Some variants of the propositional satisfiability problem are solvable in polynomial time. This includes the 2SAT problem, the propositional satisfiability problem of Krom formulae, and the HORNSAT problem, the propositional satisfiability problem of Horn formulae, which is of particular importance in this work. It is generally believed that $\mathsf{P} \neq \mathsf{NP}$, and, hence, that NP-complete problems are intractable. This is also the assumption of this work. So far, any algorithm for an NP-complete problem has at least super-polynomial running time.

4.2 Phase Transitions

Having proved a problem to be NP-complete is not the end of the computational analysis of a problem, but rather its beginning. NP-completeness is just a worst-case measure of a problem. It means that for any algorithm there exist instances which cannot be solved in polynomial time. It is possible that only one in a million instances is very hard and that the other instances can be solved easily.

There are several ways to deal with NP-complete problems. One way is to develop efficient approximation algorithms which are correct but not complete for deciding either solubility or insolubility. Another way is to use complete algorithms which require exponential time in the worst-case and to develop heuristics which solve many instances efficiently. In all cases the effectiveness of new algorithms and heuristics should be verified using a large number of instances. Since it is usually not easy to obtain a large number of real-world instances, many researchers generate instances randomly with respect to different control parameters.

Cheeseman, Kanefsky, and Taylor [Cheeseman et al., 1991] found that randomly generated instances of the NP-complete problems they studied had a very special behaviour: When ordering these instances according to a particular problem-dependent parameter, there are three different regions with respect to the solubility of the instances that occur when changing the parameter. In one region instances are soluble with a very high probability, in one region instances are insoluble with a very high probability, and in between these two regions there is a very small region

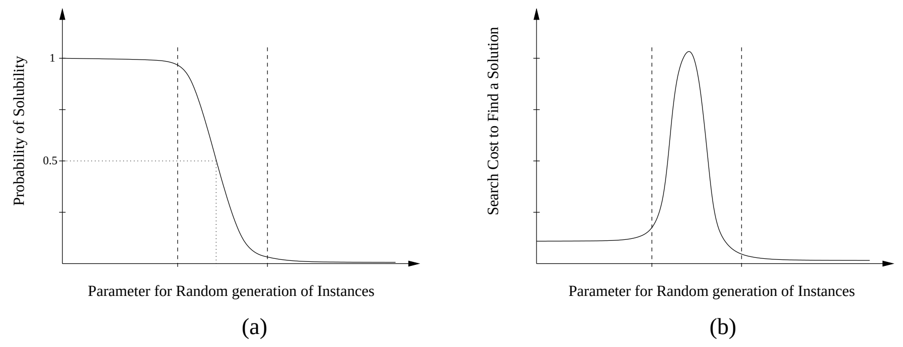

Figure 1.8. Typical phase-transition behaviour of randomly generated instances.

where the probability of solubility of these instances changes abruptly from very high to very low (see Figure 1.8(a)). Cheeseman et al. [Cheeseman et al., 1991] called this region *phase-transition.* In this region a small change of the local parameter leads to a large change in the solubility of the instances.

Cheeseman et al. [Cheeseman et al., 1991] further found that almost all hard instances are located in the phase-transition region. In general, instances in the phase transition appear to be harder than instances in the other two regions (see Figure 1.8(b)). This is because instances in soluble region are under-constrained and for this reason any search method finds a solution very fast without much backtracking. Similarly, instances in the insoluble region are over-constrained and for this reason search methods fail quite early when searching through the space of possible solutions. In some studies, however, it turned out that some under-constrained instances are particularly hard [Gent and Walsh, 1996].

The behaviour of randomly generated instances of NP-complete problems described by Cheeseman et al. was found by many researchers for many NP-complete problems, although satisfiability problems were the most studied problems. A typical parameter for satisfiability problems that causes a phase transition is the ratio of clauses-to-variables. An interesting selection of papers on the topic can be found in [Hogg et al., 1996].

4.3 How to prove NP-hardness and NP membership for spatial CSPs

In order to prove NP-hardness of a decision problem, in our case the consistency problem of a set of spatial constraints CSPSAT, it is sufficient to find another NP-hard problem that can be polynomially reduced to the problem at hand. Usually this has to be done in a different way for each new problem again and again and it is in many cases a difficult task to find a new transformation and to prove that it is a one to one transformation. The difficulty of this problem can be estimated when considering that new NP-hardness proofs often deserve a publication.

When looking at spatial CSPs over different sets of relations it is striking that they all have the same structure with different relations. One might expect that the same reduction with different parameters can be used for different sets of relations and that a general transformation scheme can be used. In this section we present a scheme which we developed for proving NP-hardness of different subsets of RCC-8 and which seems to be general in the way that the parameters of the scheme can be found by exhaustive search over possible relations, no matter what the relations are. So the transformations could essentially be identified automatically for any system of relations.

Our scheme uses a transformation from a propositional satisfiability problem to CSPSAT($\mathcal{S}$) where $\mathcal{S}$ is a subset of a system of relations $2^{\mathcal{B}}$ by constructing a set of spatial constraints Θ for every instance $\mathcal{I}$ of the propositional satisfiability problem, such that Θ is consistent if and only if $\mathcal{I}$ is a positive instance. The propositional satisfiability problems we use are 3SAT, the problem of deciding whether there is a truth assignment for a set of clauses where each clause has exactly three literals, as well as two variants of 3SAT where truth assignments of particular types are required. These variants are NOT-ALL-EQUAL-3SAT, the problem of deciding whether there is a truth assignment such that for every clause at least one literal is assigned *true* and one literal is assigned $false$, and ONE-IN-THREE-3SAT, the problem of deciding whether there is a truth assignment such that for every clause exactly one literal in every clause is assigned *true*. All three decision problems are NP-hard [Schaefer, 1978].

The different transformations have in common that every variable v of the propositional satisfiability problem is transformed to two constraints $x_v\{R_t, R_f\}y_v$ and $x_{\neg v}\{R_t, R_f\}y_{\neg v}$ corresponding to the positive and the negative literal of v, where R_t and R_f are relations of $\mathcal{S}$ with $R_t \cap R_f = \emptyset$. v is assigned *true* if and only if $x_v\{R_t\}y_v$ holds and assigned $false$ if and only if $x_v\{R_f\}y_v$ holds. Since the two literals corresponding to a variable need to have opposite assignments, we have to make sure that $x_v\{R_t\}y_v$ holds if and only if $x_{\neg v}\{R_f\}y_{\neg v}$ holds, and *vice versa*, for which additional *polarity constraints* are required. In addition, every literal occurrence l of the propositional satisfiability problem is transformed to the constraint $x_l\{R_t, R_f\}y_l$, where $x_l\{R_t\}y_l$ holds if and only if l is assigned *true*. In order to assure the correct assignment of positive and negative literal occurrences with respect to the corresponding variable, polarity constraints are required again. For instance, if the variable v is assigned *true*, i.e., $x_v\{R_t\}y_v$ holds, then $x_p\{R_t\}y_p$ must hold for every positive literal occurrence p of v, and $x_n\{R_f\}y_n$ must hold for every negative literal occurrence n of v. Further, *clause constraints* have to be added to assure that the clause requirements of the specific propositional satisfiability problem are satisfied. For example, if $\{i, j, k\}$ is a clause of an instance of ONE-IN-THREE-3SAT, then exactly one of the constraints $x_i\{R_t\}y_i$, $x_j\{R_t\}y_j$, and $x_k\{R_t\}y_k$ must hold.

According to this scheme, all we have to do in order to find a transformation is to identify relations $R_f, R_t \in \mathcal{S}$, the polarity constraints which enable to propagate the assignment of literal occurrences to other literal occurrences, and the clause constraints which ensure that properties of clauses also hold for their transformation. These constraints can be found by exhaustively assigning and testing the polarity CSP of figure 1.9(a) and based on this, the clause CSP of figure 1.9(b). If it is possible to identify the polarity and the clause constraints, then we have found a polynomial transformation from a propositional satisfiability problem to the consistency problem of $\mathcal{S}$.

The next step is to show that this transformation is a many-to-one transformation such that whenever we have a positive instance of the propositional satisfiability problem we get a positive instance of the consistency problem. Unlike finding polarity and clause constraints, this part of the NP-hardness proof cannot be automated as it depends on the domains we are using. However, since the CSP we get is very structured with only the polarity constraints and the clause constraints, this can be easy to show in many cases. It is actually an advantage that the domain we are using is infinite as it allows to treat the different polarity and clause constraints almost independently. Examples for transforming propositional satisfiability problems to CSPSAT($\mathcal{S}$) for different subsets $\mathcal{S}$ of RCC-8 can be found in [Renz and Nebel, 1999].

The next step in the complexity analysis of a given spatial calculus is to prove NP membership of CSPSAT. Recall that in order to show that a decision problem is a member of NP, we have to show that a possible solution can be checked in polynomial time. So for showing that CSPSATis in NP for a system of relations $2^{\mathcal{B}}$ over a domain $\mathcal{D}$ it is sufficient to show that CSPSAT($\mathcal{B}$) over $\mathcal{D}$ is a member of NP. While for many NP-complete problems the NP-membership proof is easier than showing NP-hardness, it is the other way around for spatial CSPs. This is due to the fact that we might have to check arbitrary spatial entities which might not even be representable in a computational framework (see e.g. [Renz, 1998]) and due to the infinity of the domain $\mathcal{D}$. Proving NP membership can be very difficult and has to be done for each system

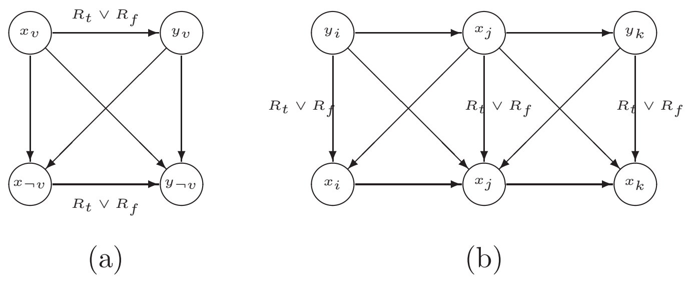

Figure 1.9. The polarity constraints (a) assure that positive and negative literals of the same variable have opposite assignments. The clause constraints (b) assure that the clause requirements of the particular 3SAT problem are satisfied.

of relations and for each domain separately. Consider for example the RCC-8 relations and Egenhofer's relations. The composition table of the relations are the same, but the domains are different. While the RCC-8 domain consists of regular subsets of a topological space, Egenhofer's domains consist of connected two-dimensional regions without holes which is much more restricted. The consequence of this is that while RCC-8 is in NP [Renz, 1998], NP membership of Egenhofer's calculus is still an open problem [Grigni et al., 1995]. Instead of proving NP membership by showing that a given solution can be verified in polynomial time, we can also give a polynomial time decision procedure for CSPSAT($\mathcal{B}$) over $\mathcal{D}$ and show that whenever the decision procedure recognises an instance Θ as consistent, there is an instantiation of all variables in Θ with values of the domain $\mathcal{D}$ such that all constraints of Θ are satisfied. Having a polynomial decision procedure is a stronger result and implies NP membership. In the next session we will look at how such decision procedures can be identified.

5. Identifying tractable subsets of spatial CSPs

Reasoning about most interesting spatial calculi is NP-hard. This, however, is often true only for the full calculus, i.e., if all relations $2^{\mathcal{B}}$ can be used. If we restrict ourselves to subsets $\mathcal{S} \subseteq 2^{\mathcal{B}}$ of the full set of relations, it might be possible that reasoning over this subset is tractable. Ideally we are interested in finding maximal tractable subsets of $2^{\mathcal{B}}$ which are those subsets which are tractable and which become NP-hard if any other relation is added. This represents the boundary between tractability and NP-hardness. Some subsets are obviously tractable such as the set of relations that contain the identity relations as a disjunct. The subsets that are most interesting are those that contain all the basic relations $\mathcal{B}$. So as a minimal requirement and as a first step we have to show that the set of basic relations is tractable.

For RCC-8, Renz and Nebel [Renz and Nebel, 1999] showed tractability of the basic relations by developing a polynomial transformation of RCC-8 constraints into SAT formulae. Those RCC-8 relations that transform into a Horn formula together form a tractable subset. Altogether 64 relations were identified in this way, among them were all the basic relations. While we could try to develop a new algorithm or a new transformation for every spatial calculus and for different subsets of them, it is highly desirable that the path-consistency algorithm (see section 2.1) can be used for deciding consistency of tractable subsets. If this is the case then consistency can be decided purely by algebraic operations on the relations without having to fall back to the infinite domains. And we have to deal with the domains only once for proving that path-consistency decides consistency. Obviously, this again depends strongly on the domains and the relations that are used and cannot be generalised. Therefore we have to find a new tractability proof for every set of basic relations over every domain. This can be very complicated as we have to deal with infinite domains.

For RCC-8, for example, the proof that the path-consistency algorithm decides consistency for the basic relations (actually for a larger set of relations) was done as follows [Renz and Nebel, 1999]. First, it was analysed how applying path-consistency can lead to an inconsistency. Then it was shown that whenever the path-consistency algorithm detects an inconsistency, positive unit resolution applied to the SAT encoding of RCC-8 produces the empty clause. In [Renz, 1998] an algorithm was presented which computes an instantiation for all variables of any consistent set of constraints over the RCC-8 basic relations. This algorithm works for Euclidean spaces in all dimensions d. For $d \geq 3$ it also works for connected regions without holes.

Once it has been shown that path-consistency decides consistency for the basic relations, it is possible to try to extend the set of relations and to identify larger tractable subsets. There are basically two general methods which can be used for extending tractability of subsets of relations to larger subsets: the closure method [Renz and Nebel, 1999] and the refinement method [Renz, 1999]. We will describe these methods in the following section. In particular the refinement method seems very powerful and will be presented in more detail.

5.1 Closure of sets of relations

Given a system of relations $2^{\mathcal{B}}$, the number of subsets $\mathcal{S} \subseteq 2^{\mathcal{B}}$ that we might have to analyse for a computational analysis is huge, namely, $2^{(2^{|\mathcal{B}|})}$. This number can be slightly reduced if only those subsets are considered that contain all the basic relations and possibly the universal relation. Fortunately, we can reduce the number of subsets further by noting that the computational complexity associated with an arbitrary subset $\mathcal{S}$ is identical to the complexity associated with the closure of this subset under composition, intersection, and converse, denoted by $\widehat{\mathcal{S}}$ – an observation that was first used in determining a maximal tractable subset of Allen's interval calculus [Nebel and Bürckert, 1995, Theorem 14]. Renz and Nebel [Renz and Nebel, 1999] proved this for arbitrary systems of relations and came up with the following theorem.

Theorem 1.1 *Let $\mathcal{C}$ be a set of binary relations that is closed under composition, intersection, and converse. Then for any subset $\mathcal{S} \subseteq \mathcal{C}$ that contains the universal relation, the problem* $\mathsf{CSPSAT}(\widehat{\mathcal{S}})$ *can be polynomially reduced to* $\mathsf{CSPSAT}(\mathcal{S})$.

Note that Theorem 1.1 holds only if there exists an infinite supply of fresh variables; this is not always the case (e.g., bounded variable problems which are studied in logic and model theory). Another requirement of Theorem 1.1 is the possibility to specify more than one constraint for each pair of variables. Otherwise the identity relation must be contained in $\mathcal{S}$. The following corollary specifies how Theorem 1.1 will be used.

Corollary 1.2 *Let $\mathcal{S}$ be a subset of $2^{\mathcal{B}}$.*

1 $\mathsf{CSPSAT}(\widehat{\mathcal{S}}) \in \mathsf{P}$ *if and only if* $\mathsf{CSPSAT}(\mathcal{S}) \in \mathsf{P}$.

2 $\mathsf{CSPSAT}(\mathcal{S})$ *is* NP*-hard if and only if* $\mathsf{CSPSAT}(\widehat{\mathcal{S}})$ *is* NP*-hard.*

The first statement of Corollary 1.2 can be used to increase the number of elements of tractable subsets of CSPSAT considerably. With the second statement of Corollary 1.2 NP-hardness proofs of CSPSAT can be used to exclude certain relations from being in any tractable subset of CSPSAT. In any case, we will have to analyse only those subsets that are closed under composition, converse and intersection.

The computational analysis of RCC-8 shows how powerful this method is. The closure of the set of 64 relations that transform to Horn formulae consists of 148 relations (called $\widehat{\mathcal{H}}_8$) and turns out to be a maximal tractable subset of RCC-8. Furthermore, it has been shown [Renz and Nebel, 1999] that path-consistency decides $\mathsf{CSPSAT}(\widehat{\mathcal{H}}_8)$.

5.2 The refinement method

In this subsection we present a general method for proving tractability of reasoning over disjunctions of a JEPD set $\mathcal{B}$ of binary relations over a domain $\mathcal{D}$ which are atoms of a relation algebra, i.e., a method for proving tractability of $\mathsf{CSPSAT}(\mathcal{S})$ for sets $\mathcal{S} \subseteq 2^{\mathcal{B}}$ (see Section 2.2). In order to do so, this method requires a subset $\mathcal{T}$ of $2^{\mathcal{B}}$ for which path-consistency is already known to decide $\mathsf{CSPSAT}(\mathcal{T})$. Then the method checks whether it is possible to refine every constraint involving a relation in $\mathcal{S}$ according to a particular refinement scheme to a constraint involving a relation in $\mathcal{T}$ without changing consistency. The following definition will be central for this method.

Definition 1.3 (reduction by refinement)
Let $\mathcal{S}, \mathcal{T} \subseteq 2^{\mathcal{B}}$. $\mathcal{S}$ can be reduced by refinement *to $\mathcal{T}$, if the following two conditions are satisfied:*

1 *for every relation $S \in \mathcal{S}$ there is a relation $T_S \in \mathcal{T}$ with $T_S \subseteq S$,*

2 *every path-consistent set Θ of constraints over $\mathcal{S}$ can be refined to a set Θ' of constraints over $\mathcal{T}$ by replacing $x_iSx_j \in \Theta$ with $x_iT_Sx_j \in \Theta'$ for $i < j$, such that enforcing path-consistency to Θ' does not result in an inconsistency.*

Note that in the above definition constraints x_iSx_j are refined only for $i < j$. This is no restriction, as by enforcing path-consistency the converse constraint $x_jS^{\smile}x_i$ will also be refined. Rather it offers the possibility of refining, e.g., converse relations to other than converse subrelations, i.e., if, for instance, R is refined to r, $R^{\smile}$ can be refined to a relation other than $r^{\smile}$. This property of a set of relations can be used to derive its tractability.

Lemma 1.4 *If path-consistency decides $\mathsf{CSPSAT}(\mathcal{T})$ for a set $\mathcal{T} \subseteq 2^{\mathcal{B}}$, and $\mathcal{S}$ can be reduced by refinement to $\mathcal{T}$, then path-consistency decides $\mathsf{CSPSAT}(\mathcal{S})$.*

Proof. Let Θ be a path-consistent set of constraints over $\mathcal{S}$. Since $\mathcal{S}$ can be reduced by refinement to $\mathcal{T}$, there is by definition a set Θ' of constraints over $\mathcal{T}$ which is a refinement of Θ such that enforcing path-consistency to Θ' does not result in an inconsistency. Path-consistency decides $\mathsf{CSPSAT}(\mathcal{T})$, so Θ' is consistent, and, hence, Θ is also consistent. ■

Since path-consistency can be enforced in cubic time, it is sufficient for proving tractability of $\mathsf{CSPSAT}(\mathcal{S})$ to show that $\mathcal{S}$ can be reduced by refinement to a set $\mathcal{T}$ for which path-consistency decides $\mathsf{CSPSAT}(\mathcal{T})$. Note that for refining a constraint xSy ($S \in \mathcal{S}$) to a constraint xT_Sy ($T_S \in \mathcal{T}$), it is not required that T_S is also contained in $\mathcal{S}$. Thus, with respect to common relations the two sets $\mathcal{S}$ and $\mathcal{T}$ are independent of each other. This is in contrast to Theorem 1.1 which states that the tractability of a set of relations implies the tractability of its closure.

We will now present a method for showing that a set of relations $\mathcal{S} \subseteq 2^{\mathcal{B}}$ can be reduced by refinement to another set $\mathcal{T} \subseteq 2^{\mathcal{B}}$. In order to manage the different refinements, a *refinement matrix* is introduced that contains for every relation $S \in \mathcal{S}$ all specified refinements.

Definition 1.5 (refinement matrix)
A refinement matrix *M of $\mathcal{S}$ has $|\mathcal{S}| \times 2^{|\mathcal{B}|}$ Boolean entries such that for $S \in \mathcal{S}$, $R \in 2^{\mathcal{B}}$, $M[S][R] = true$ only if $R \subseteq S$.*

For example, if we want to build a refinement matrix which states that that the relation $\{\mathsf{DC}, \mathsf{EC}, \mathsf{PO}, \mathsf{TPP}\}$ can be refined to the relations $\{\mathsf{DC}, \mathsf{TPP}\}$ and $\{\mathsf{DC}\}$, then we set $M[\{\mathsf{DC}, \mathsf{EC}, \mathsf{PO}, \mathsf{TPP}\}][R]$ is *true* only for $R = \{\mathsf{DC}, \mathsf{TPP}\}$ and for $R = \{\mathsf{DC}\}$ and $false$ for all other relations $R \in 2^{\mathcal{B}}$. M is called the *basic refinement matrix* if $M[S][R] = true$ if and only if $S = R$.

Renz [Renz, 1999] proposes the algorithm Check-Refinements (see Figure 1.10) which takes as input a set of relations $\mathcal{S}$ and a refinement matrix M of $\mathcal{S}$. The algorithm uses triples of relations $T = (R_{12}, R_{23}, R_{13})$ which represent sets of constraints $\{xR_{12}y, yR_{23}z, xR_{13}z\}$ for some variables x, y, z. It computes all possible path-consistent triples of relations R_{12}, R_{23}, R_{13} of $\mathcal{S}$ (step 4), and enforces path-consistency (using a standard procedure Path-Consistency) to every refinement $R'_{12}, R'_{23}, R'_{13}$ for which $M[R_{ij}][R'_{ij}] = true$ for all $i, j \in \{1, 2, 3\}, i < j$ (steps 5,6). If one of these refinements results in the empty relation, the algorithm returns `fail` (step 7). Otherwise, the resulting relations $R''_{12}, R''_{23}, R''_{13}$ are added to M by setting $M[R_{ij}][R''_{ij}] = true$ for all $i, j \in \{1, 2, 3\}, i < j$ (step 8). This is repeated until M has reached a fixed point (step 9), i.e., enforcing path-consistency on any possible refinement does not result in new relations anymore. If no inconsistency is detected in this process, the algorithm returns `succeed`.

A similar algorithm, Get-Refinements, returns the revised refinement matrix if Check-Refinements returns `succeed` and the basic refinement matrix if Check-Refinements returns `fail`. Since $\mathcal{B}$ is a finite set of relations, M can be changed only a finite number of times, so both algorithms always terminate. If $n = |2^{\mathcal{A}}|$ is the total number of relations, then there are at most n^3 possible triples of relations in step 4, at most n^3 possible refinements of each triple in step 5, and at most n^2

Algorithm: Check-Refinements
Input: A set $\mathcal{S}$ and a refinement matrix M of $\mathcal{S}$.
Output: `fail` if the refinements specified in M can make a path-consistent triple of constraints over $\mathcal{S}$ inconsistent; `succeed` otherwise.

```
1. changes ← $true$
2. $while$ changes $do$
3.     $oldM \leftarrow M$
4.     $for\ every$ path-consistent triple
       $T = (R_{12}, R_{23}, R_{13})$ of relations over $\mathcal{S}$ $do$
5.         $for\ every$ refinement $T' = (R'_{12}, R'_{23}, R'_{13})$ of $T$
           with $oldM[R_{12}][R'_{12}] = oldM[R_{23}][R'_{23}] =$
           $oldM[R_{13}][R'_{13}] = true$ $do$
6.             $T'' \leftarrow$ Path-Consistency($T'$)
7.             $if$ $T'' = (R''_{12}, R''_{23}, R''_{13})$ contains the empty
               relation $then\ return$ fail
8.             $else\ do$ $M[R_{12}][R''_{12}] \leftarrow true$,
                       $M[R_{23}][R''_{23}] \leftarrow true$,
                       $M[R_{13}][R''_{13}] \leftarrow true$
9.     $if$ $M = oldM$ $then$ changes ← $false$
10. $return$ succeed
```

Figure 1.10. Algorithm Check-Refinements.

iterations of the *while* loop. Thus, a rough estimation of the worst-case running time of both algorithms leads to $O(n^8)$.

Lemma 1.6 *Let Θ be a path-consistent set of constraints over $\mathcal{S}$ and M a refinement matrix of $\mathcal{S}$. For every refinement Θ' of Θ with $x_i R' x_j \in \Theta'$ only if $x_i R x_j \in \Theta$, $i < j$, and $M[R][R'] = true$, the following holds: if* Check-Refinements($\mathcal{S}, M$) *returns* `succeed`, *enforcing path-consistency to Θ' does not result in an inconsistency.*

If Check-Refinements returns `succeed` and Get-Refinements returns M', we have pre-computed all possible refinements of every path-consistent triple of variables as given in the refinement matrix M'. Thus, applying these refinements to a path-consistent set of constraints can never result in an inconsistency when enforcing path-consistency.

Theorem 1.7 *Let $\mathcal{S}, \mathcal{T} \subseteq 2^{\mathcal{B}}$, and let M be a refinement matrix of $\mathcal{S}$. Let M' be the refinement matrix returned by* Get-Refinements($\mathcal{S}, M$). *If for every $S \in \mathcal{S}$ there is a $T_S \in \mathcal{T}$ with $M'[S][T_S] = true$, then $\mathcal{S}$ can be reduced by refinement to $\mathcal{T}$.*

By Lemma 1.4 and Theorem 1.7 we have that the procedures CHECK-REFINEMENTS and GET-REFINEMENTS can be used to prove tractability for sets of relations.

COROLLARY 1.8 *Let $\mathcal{S}, \mathcal{T} \subseteq 2^{\mathcal{B}}$ be two sets such that path-consistency decides* $\mathsf{CSPSAT}(\mathcal{T})$*, and let M be a refinement matrix of $\mathcal{S}$.* GET-REFINEMENTS$(\mathcal{S}, M)$ *returns M'. If for every $S \in \mathcal{S}$ there is a $T_S \in \mathcal{T}$ with $M'[S][T_S] = true$, then path-consistency decides* $\mathsf{CSPSAT}(\mathcal{S})$.

If a suitable refinement matrix can be found, CHECK-REFINEMENTS can be used to immediately verify that reasoning over the given set of relations is tractable. One problem with this method is that the algorithms, though polynomial, are not very efficient. Especially for large sets of relations the algorithms are very slow. Fortunately, the algorithms are used for determining tractability of reasoning over sets of relations and not for the reasoning process itself. Renz [Renz, 2002] proposed a faster version of the algorithm which uses a refinement array instead of a refinement matrix which reduces the runtime of the algorithm to $O(n^4 \log n)$.

In the following subsection we show how the refinement method can be applied to RCC-8 for proving certain subsets to be tractable. For RCC-8 it will lead to a complete analysis of tractability by identifying all three maximal tractable subsets.

5.3 Applying the refinement method

The refinement method requires for any input set of relations $\mathcal{S} \subseteq 2^{\mathcal{B}}$ a subset $\mathcal{T} \subseteq 2^{\mathcal{B}}$ for which path-consistency is known to decide consistency and a refinement strategy $\mathcal{S} \Rightarrow \mathcal{T}$. Assuming that a set $\mathcal{T}$ is known, the main tasks are to find a candidate set $\mathcal{S}$ and a refinement strategy, i.e., we have to find for every relation of $\mathcal{S}$ a relation of $\mathcal{T}$ and apply the refinement algorithm using the different refinement strategies.

Candidate sets $\mathcal{S}_i$ can be found by using the closure method and the known NP-hard relations. In [Renz, 1999], Renz identified candidate sets for RCC-8 by computing the largest subsets of RCC-8 that contain the basic relations, the universal relation, are closed under the operators and do not contain any of the known NP-hard relations, i.e., the relations that can be used for the NP-hardness proofs. This resulted in only three candidate sets (which are called $\mathcal{C}_8$, $\mathcal{Q}_8$ and the already known maximal tractable subset $\widehat{\mathcal{H}}_8$) which can be tested using the refinement method, provided that a refinement strategy can be found.

One way of finding a refinement strategy is to use a greedy method of extending partial refinement strategies by first refining only one or a few relations, fill the refinement matrix/array using the refinement algorithm and if no inconsistency occurs add some more refinements. This can be repeated until a working refinement strategy can be found or until it is shown that no refinement strategy exists. It might also be possible that a particular refinement strategy, the *identity refinement strategy*, is applicable. The identity refinement strategy refines each relation $R \in \mathcal{S}$ to the relation $R' = R \setminus ID$, where $ID \in \mathcal{B}$ is the identity relation. Renz [Renz, 1999] observed that all relations that are contained in the two candidate sets for RCC-8 which are not contained in $\widehat{\mathcal{H}}_8$ can be refined to relations of $\widehat{\mathcal{H}}_8$ by removing the identity relation. It turned out that applying the refinement algorithm to the candidate sets of RCC-8 leads to refinement matrices that contain a basic relation for each relation of the candidate sets. This shows that each of these candidate sets can be refined to the set of basic relations and, therefore, that the candidate sets are tractable and can be decided by the path-consistency algorithm. Renz [Renz, 1999] also applied the identity refinement matrix to the known maximal tractable subset ORD-Horn of the interval algebra [Nebel and Bürckert, 1995] and it turns out that the refinement method also works for the interval algebra.

Now we have all the tools for identifying (maximal) tractable subsets of a system of spatial relations. In the next subsection we show how these sets can be used for finding fast solutions to intractable CSPSATinstances.

6. Practical Efficiency of Reasoning Methods

In the previous section we described how to find tractable subsets of the usually NP-hard spatial calculi. For most of the tractable subsets path-consistency or even simpler methods are sufficient for deciding consistency, so except for very large instances or for calculi over a large set of relations, there are usually no efficiency problems when considering instances that contain only relations of a tractable subset. Efficiency problems occur, however, if we go outside the tractable subsets and enter the NP-hard territory. As we will see later, instances of an NP-hard problem can often be solved very fast in practice. basically four reasons for this. The first one is that the interleaved applications of path-consistency during the backtracking search is often very powerful and already eliminates many labels that cannot lead to a solution. The second reason is that large tractable subsets reduce the size of the backtracking search tree by several orders of magnitude. This results from the possibility of splitting relations into tractable sub-relations instead of splitting them into all contained basic relations. The third reason is that different heuristics and strategies can be applied for solving hard instances. Often it is the case that there is a heuristic for which a hard instances turns out to be easy. So the more heuristics and strategies are available the higher is the likelihood that one of them can solve an instance fast. The last reason is the observation that most instances outside the phase-transition region are in almost all cases very easy to solve. In the following we will discuss these points in more detail and show results from an empirical investigation of the practical efficiency of RCC-8.

6.1 Generating Test Instances

In order to test the practical efficiency of reasoning algorithms, it is necessary to generate a large number of test instances. Ideally these should be real instances of existing applications. If such an application is not available, instances have to be generated systematically or randomly. Since many instances are easy to solve, it is important to try to generate instances that are as hard as possible. When randomly generating instances, there is usually a parameter that produces a phase-transition of the probability of satisfiability of the generated instances, i.e., when increasing the value of the parameter, the probability changes from almost 1 to almost 0 (or vice versa) within a very small range of the parameter (see Figure 1.8). Almost all instances outside the phase-transition region are very easy to solve while the phase-transition region contains most hard instances [Cheeseman et al., 1991]. The most useful instances for empirical study of reasoning algorithms can therefore be found in and around the phase-transition region, which has to be empirically determined.

For randomly generated RCC-8 instances it turned out that one phase-transition is induced by the degree d of nodes, i.e., how many edges for each node of the constraint graph are randomly instantiated on average [**?**]. The phase-transition turns out to be around $d = 10$. Another way of generating hard instances is to randomly generate instances that contain only relations that are outside the tractable subsets. This, however, is mainly for testing the behaviour of the algorithms in extreme cases and is not very representative for practical purposes for which it might better to analyse a uniform distribution of the relations. Another important factor when generating random instances is to make sure that the instances are not trivially flawed [Achlioptas et al., 1997], i.e., the probability that small inconsistent sub-CSPs, such as inconsistent triples, are contained in the instances should not be high and should not determine the phase-transition.

The following empirical results are taken from [?] and show how randomly generated RCC-8 instances can be solved very efficiently. The random RCC-8 instances were generated according to the model $A(n, d, l)$, where n is the number of variables, d the average degree and l the average number of base relations per relation. The relations were selected among all RCC-8 relations, for RCC-8 $l = 4.0$ means that all relations are selected with equal probability.

6.2 Testing Algorithms

When testing algorithms on the generated instances, several properties are interesting and should be observed. One is of course the time it takes to solve the instances, but it is also important to compare the number of nodes of the backtracking search space that were visited while solving instances. This value is important for comparing algorithms on different machines as the run-time differs from machine to machine and also depends on other factors such as the load and the available memory of the machine used for the test. Instead of using only the average values (runtime, visited nodes, etc.) we also look at different percentiles, i.e., we order the values and look at the values of the elements at position 50%, 70%, 90%, or 99%. Since we are dealing with an NP-complete problem for which some instances take a very long time to solve, taking the average only would be too erratic. In the following we mainly look at 99% percentile instances as these give a good indication of the performance for the hardest among the instances.

6.3 Effect of using large tractable subsets

A very important factor in obtaining more efficient solutions to instances of an NP-hard spatial reasoning problem is the use of large tractable subsets of the NP-hard set of relations. The backtracking algorithms split each constraint into sub-constraints that contain only relations of a tractable subset where each split spans a new subtree of the search space. Using large tractable subsets makes it possible to split the constraints into fewer sub-constraints, thus reducing the number of subtrees and the size of the search space. This can be measured in terms of the average branching factor of a search tree. For RCC-8, using the set of basic relations for splitting the constraints leads to an average branching factor of $b = 4$ which corresponds in this case to the average number of basic relations in each of the 256 RCC-8 relations. For the maximal tractable subsets $\widehat{\mathcal{H}}_8$, $\mathcal{Q}_8$, and $\mathcal{C}_8$, the average branching factors are $b = 1.4375$, $b = 1.516$, and $b = 1.523$, respectively. The average size of the search spaces can be computed as $b^{(n^2-n)/2}$. As can be seen in

Table 1.3. Average size of the search space depending on the number of variables and the branching factor of the split set.

#regions	$\mathcal{B}(4.0)$	$\widehat{\mathcal{B}}(2.5)$	$\widehat{\mathcal{H}}_8$ (1.4375)
5	10^6	9537	37
7	4.4×10^{12}	2.3×10^8	2040
10	1.2×10^{27}	8.1×10^{17}	10^7
20	2.5×10^{114}	4.1×10^{75}	8.8×10^{29}

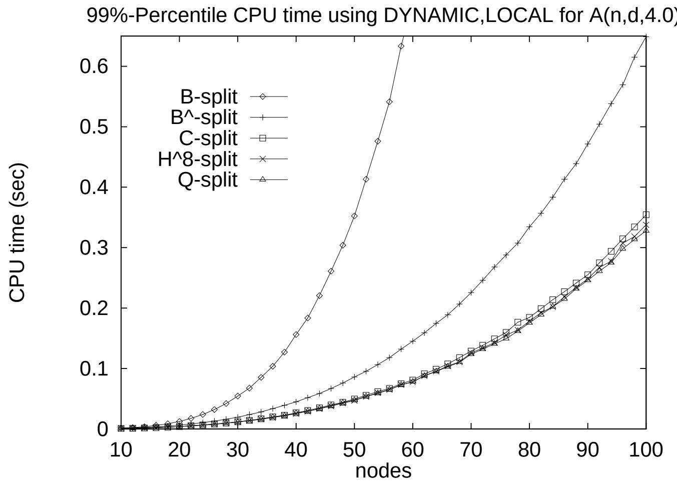

Figure 1.11. 99% percentile running times for solving RCC-8 instances of the phase-transition region using different tractable subsets ($d = 8.0$ to $d = 10.0$, 2,500 instances per data point).

Table 6.3 this results in considerably smaller search spaces. This however is not fully reflected in the empirical results because of the effect of the interleaved applications of the path-consistency algorithm at each node of the search tree which eliminates inconsistent relations from the constraints and has a similar effect of reducing the search space. Both methods together, path-consistency and large tractable subsets, already lead to quite impressive results for solving randomly generated RCC-8 instances. In Figure 1.11 we see the 99% percentile running times for solving instances of the phase-transition region using different tractable subsets. The maximal tractable subset lead to considerably faster solutions but not as much faster as suggested by table 6.3

6.4 Effect of different heuristics

Another factor for obtaining faster solutions is to use different heuristics for choosing the path through the search space. There are two positions in the backtracking algorithms where a heuristic choice can be made. One is the order in which the constraints are selected, the other choice is the order of the sub-relations when splitting a constraint. For both choices we can apply different heuristics which influence the search space and the path through the search space. It is clear that the choice of the heuristics has more effect on consistent instances. In order to determine that an instance is consistent, it is sufficient to find one path from the root of the search tree to a consistent leaf. So if the perfect heuristic choice is made at all nodes, any consistent instance can be solved without backtracking. For inconsistent instances, all possible leafs of the search tree must be inconsistent, so the fastest way to determine inconsistency is when this can be detected early on in the search tree. We chose two different heuristics for the ordering of constraints and two for the ordering of sub-relations [Nebel, 1997].

static/dynamic: Constraints are processed according to a heuristic evaluation of their constrainedness which is determined *statically* before the backtracking starts or *dynamically* during the search.

local/global: The evaluation of the constrainedness is based on a *local* heuristic weight criterion or on a *global* heuristic criterion [van Beek and Manchak, 1996].

In Figure 1.12 the 99% percentiles are shown for the different combinations of heuristics, the second column of Table 1.4 shows the number of hard instances for each combinations. Hard instances are considered to be those that cannot be solved by using 10.000 visited nodes in the search space. It can be seen that although some combinations are better than others, they are all quite successful and the differences are not enormous. Their real advantage is described in the following section.

6.5 Effects of combining different strategies

We denote as a strategy a choice of tractable subset for splitting, a heuristic for constraint selection and a heuristic for sub-relation ordering.
As described in the previous section, every consistent instance can be solved without backtracking if the right heuristic choice is made at each node. Therefore it is not surprising that some instances can be solved faster by one strategy while other instances are solved faster by other strategies. This means that it might be possible to solve more instances efficiently by combining different strategies than by each strategy alone.

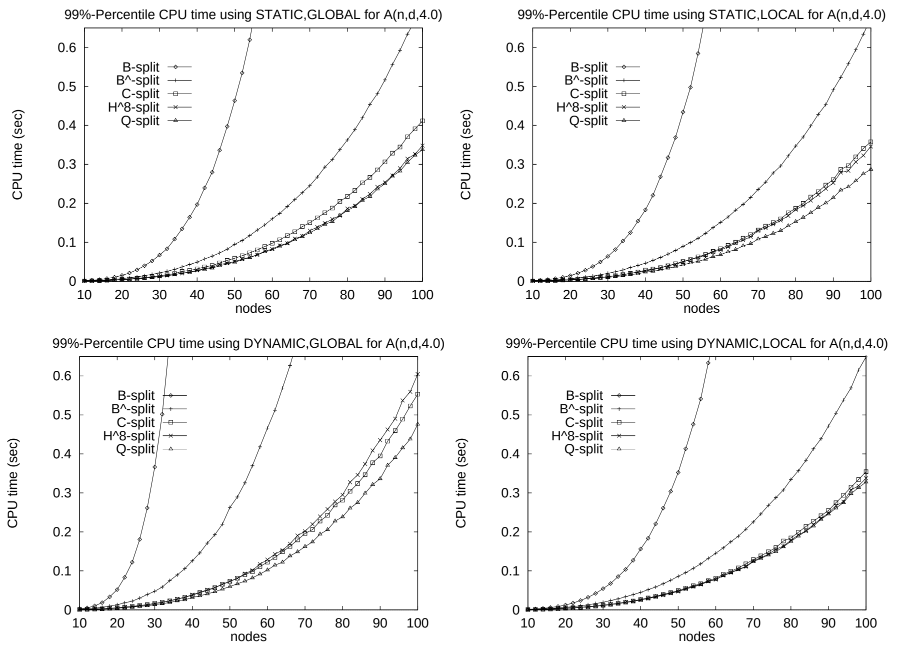

Figure 1.12. Percentile 99% CPU time of the different heuristics for solving $A(n, d, 4.0)$ ($d = 8.0$ to $d = 10.0$, 2,500 instances per data point).

We tested this hypothesis by running all strategies on the set of all hard instances identified in the experiment described above. It turns out that almost all of these hard instances can be solved by at least one strategy (see Table 1.4). We also looked at which strategy gives the first response, i.e., which strategy solves each instance fastest, which is shown in the same table. In most cases, the first response comes very fast, usually with less than 300 visited nodes in the search space (see Figure 1.13). It is surprising that the inconsistent instances can be solved particularly fast which shows a clear advantage of the method of combining different strategies to random methods with restarts. Random methods are actually completely useless for inconsistent instances because these methods are not complete. In order to push our methods even further, we also looked at how well the different strategies complement each other, and tried to find the combination of strategies which solves the instances with the least accumulated number of nodes. It turns out that four strategies ($\widehat{\mathcal{H}}_8$/*static*/*global*, $\widehat{\mathcal{H}}_8$/*dynamic*/*local*, $\mathcal{C}_8$/*dynamic*/*local*, $\widehat{\mathcal{B}}$/*static*/*local*) complement each other particularly well. By combining these four strategies, almost all instances in the phase transition region

Table 1.4. The second column shows the number of hard instances for each heuristic, there are 788 hard instances in total. Column three shows the percentage of solved hard instances for each heuristic and column four the percentage of first response when orthogonally running all heuristics. Note that sometimes different heuristics are equally fast. Therefore the sum is more than 100%.

		$A(n,d,4.0)$	
Heuristics	# Hard Instances	Solved Instances	1. Response
$\widehat{\mathcal{H}}_8$/sta/loc	64	91.88%	19.80%
$\widehat{\mathcal{H}}_8$/sta/glo	42	94.67%	12.56%
$\widehat{\mathcal{H}}_8$/dyn/loc	52	93.40%	24.37%
$\widehat{\mathcal{H}}_8$/dyn/glo	100	87.31%	13.58%
$\mathcal{C}_8$/sta/loc	81	89.72%	6.35%
$\mathcal{C}_8$/sta/glo	58	92.64%	5.20%
$\mathcal{C}_8$/dyn/loc	78	90.10%	5.96%
$\mathcal{C}_8$/dyn/glo	108	86.63%	6.60%
$\mathcal{Q}_8$/sta/loc	81	89.72%	9.77%
$\mathcal{Q}_8$/sta/glo	54	93.15%	12.06%
$\mathcal{Q}_8$/dyn/loc	74	90.61%	10.15%
$\mathcal{Q}_8$/dyn/glo	104	86.80%	12.82%
$\widehat{\mathcal{B}}$/sta/loc	68	91.37%	1.40%
$\widehat{\mathcal{B}}$/sta/glo	89	88.71%	1.27%
$\widehat{\mathcal{B}}$/dyn/loc	70	91.12%	0.89%
$\widehat{\mathcal{B}}$/dyn/glo	162	79.44%	0.89%
$\mathcal{B}$/sta/loc	163	79.31%	0.51%
$\mathcal{B}$/sta/glo	222	71.83%	0.25%
$\mathcal{B}$/dyn/loc	209	73.48%	0.51%
$\mathcal{B}$/dyn/glo	(303)	–	0.13%
combined	788	99.87%	

can be solved by restricting the combined number of visited nodes to a value which is linear in the size of the instances. We tested this for CSPs up to a size of 500 variables, i.e., CSPs with about 25.000 relations. At that point the increased run-time of the interleaved path-consistency computations turned out to be the limiting factor.

6.6 Discussion

We have seen that even though spatial reasoning with RCC-8 is an NP-complete problem, we were able to solve almost all of the hardest instances identified in our experiments in reasonable time. This is only possible through the use of the maximal tractable subsets that we identified by a theoretical analysis of the reasoning problem. For RCC-8

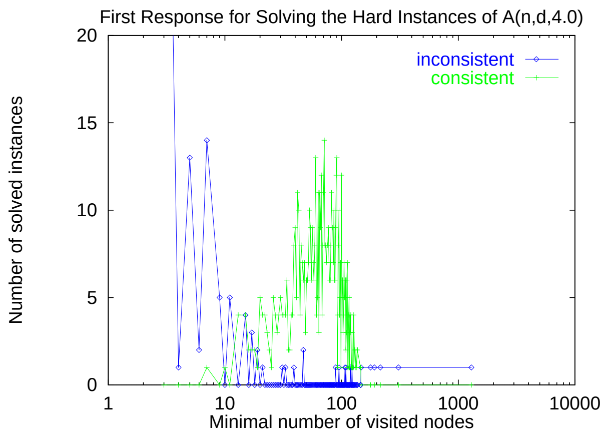

Figure 1.13. Fastest solution of the hard instances when running all heuristics in parallel.

this turns out to work particularly well, which is due to the existence of three different maximal tractable subsets for RCC-8 but also because RCC-8 is a rather small algebra. Empirical studies for Allen's interval algebra [Allen, 1983] which has 13 basic relations but only one maximal tractable subset which contains all basic relations, show that reasoning is still much more efficient in practice when using the maximal tractable subclass than without [Nebel, 1997], but the overall practical efficiency was not as impressive as for RCC-8. Nevertheless, identifying maximal tractable subsets that contain all basic relations is an essential part if more efficient solutions to an NP-complete spatial or temporal reasoning problem are to be found. Although attempts have been made to identify maximal tractable subsets that do not contain all basic relations [Krokhin et al., 2003], these subsets cannot be used for obtaining more efficient solutions to the general NP-complete problem as it is not possible to split each relation into members of these maximal tractable subset. Another important finding is that combining different strategies leads to much better results than trying to optimise one strategy. In that respect we can conclude that the more strategies the better. This includes analysing different heuristics as well as using different tractable subsets which even includes using subsets of tractable subsets.

7. Combination of Spatial Calculi

There has been a large amount of research on qualitative spatial calculi, a fraction of it has been described in this chapter, and more in other chapters of this book. The usefulness of these research efforts, however, largely depends on how well this research can make its way into practical applications. Without a doubt, space is one of the fundamental aspects of our daily life and of our physical world, and therefore qualitative spatial representation and reasoning should be an essential part in many applications. It is remarkable, however, that up to now there are relatively few real applications. One reason for this lack of applications is that research has mainly focused on understanding and analysing single, isolated aspects of space, like distance, direction, or topology. The spatial calculi we presented so far all fall into this "single aspect" category. On the other hand, almost all possible applications require different aspects of space and not only topology or only direction. Future research on qualitative spatial representation and reasoning should focus strongly on combining different aspects of space, on developing and analysing spatial calculi over different aspects of space, and on methods for dealing with these calculi. In this chapter we will present some promising first attempts in this direction.

The first approach is by Gerevini and Renz [Gerevini and Renz, 2002] who combine topology and size information and who introduce several modifications of the existing constraint algorithms for dealing with different kinds of constraints. A second approach is by Renz [Renz, 2001] who combines directional and topological information for one-dimensional intervals by adding direction of intervals to the interval algebra.

7.1 Different ways of combining multiple aspects of space

Constraint-based approaches in principle support the use of different kinds of constraints if they work on the same domains. This is relatively straightforward if finite domains are used where the constraint algorithms manipulate the domains of the variables. For qualitative spatial reasoning where infinite domains are used and constraint algorithms work on relation-symbols instead of restricting domains, this turns out to be a difficult problem. The reason for this is that relations over one aspect are not independent of relations of another aspect. For example if the distance between two objects is large, they cannot overlap. If one object is contained in the other one it must be smaller. These are two simple examples which show that topology is neither independent from direction nor from size. These restrictions and dependencies must be enforced on the relational level and must therefore be analysed when developing a combined calculus and must be precomputed like a composition table.

One way of developing a calculus for multiple aspects of space is to take the relations for each aspect, for example two sets of basic relations $\mathcal{R} = \{R_1, \ldots, R_n\}$ and $\mathcal{S} = \{S_1, \ldots, S_m\}$, and form new relations as the cross product $\mathcal{R} \times \mathcal{S}$. Some of the new relations will be empty and can be removed. The advantage of this approach is that the dependencies of the different aspects are implicitly encoded in the new composition table and that all the existing reasoning algorithms can be used. The disadvantage is the large number of relations that result from this approach (which makes reasoning and also analysing the combined calculus very time and space consuming). An example for this approach is by Pujari et al. [Pujari et al., 1999] in the area of temporal reasoning where the interval algebra is combined with relative durations of intervals. Another example is by Renz [Renz, 2001] who added direction of intervals to the interval algebra.

An alternative approach is to treat the different aspects separately and to develop new reasoning algorithms for combining different sets of constraints and their dependencies. Different aspects and different granularities can then be added in a modular way without having an explosion in the number of relations. One problem here is how to keep track of the interactions between the different sets of relations and their interactions, i.e, how can relations of different sets be composed, intersected etc. This approach will be further discussed in the following section where we take the combination of topology and qualitative size as an example.

In any case, it is essential that different aspects can only be combined when they use the same underlying spatial entities, such as points or regular regions.

7.2 Combining topological and size information

When having two sets of basic relations $\mathcal{A}$ and $\mathcal{B}$ over a domain $\mathcal{D}$ where both of them split $\mathcal{D} \times \mathcal{D}$ exhaustively, it is clear that the relations of the two sets taken together are not pairwise disjoint and, hence, cannot be independent of each other. Instead of looking at the intersections of all the relations and to treat them as a new set of JEPD relations, we will present in this section methods for taking the sets separately and propagating their interactions. For this it is necessary to first look at all the interactions that can possibly occur. As an example we take the work by Gerevini and Renz [Gerevini and Renz, 2002] who combined RCC-8 with qualitative size relations. Given a set V of spatial region

Table 1.5. Interdependencies of basic RCC-8 relations (r) and basic $\mathcal{QS}$ relations (s).

r		$Sizerel(r)$	r		$Sizerel(r)$	s		$Toprel(s)$
TPP	$\models$	$<$	DC	$\models$	$?$	$=$	$\models$	DC, EC, PO, EQ
NTPP	$\models$	$<$	EC	$\models$	$?$	$>$	$\models$	DC, EC, PO, TPP^{-1}, NTPP^{-1}
TPP^{-1}	$\models$	$>$	PO	$\models$	$?$	$<$	$\models$	DC, EC, PO, TPP, NTPP
NTPP^{-1}	$\models$	$>$	EQ	$\models$	$=$			

variables, a set of $\mathcal{QS}$-constraints over V is a set of constraints of the form $size(x)\ S\ size(y)$, where $S \in \mathcal{QS}$, $size(x)$ is the size of the region x, $size(y)$ is the size of the region y, and $x, y \in V$. $\mathcal{QS} = \{<, >, =, \leq, \geq, \neq, <=>\}$. Their interactions are rather simple and are mainly due to the fact that regions which are contained in other regions must be smaller than the containing region. All interactions can be found in Table 1.5. $Sizerel(r)$ is the qualitative size relation entailed by an RCC-8 relation r, while $Toprel(s)$ is the RCC-8 relation entailed by a qualitative size relation.

Now we consider pairs of relations as new relations, i.e., we consider constraints of the form xRy where $R = \langle R_a, R_b \rangle$ and $R_a \in \mathcal{A}$ and $R_b \in \mathcal{B}$. This is equivalent to having two sets of constraints Θ and Σ over the same set of variables and the same domain where Θ contains the RCC-8 constraints and Σ the qualitative size constraints. If both sets are independently consistent, it is clear that the two sets taken together are not necessarily consistent too as their interactions must be considered.

A natural method for deciding the consistency of a set of RCC-8 constraints and a set of $\mathcal{QS}$-constraints, would be to first extend each set of constraints with the constraints entailed by the other set, and then independently check the consistency of the extended sets by using a path-consistency algorithm. However, as the example below shows, this method is not complete for $\widehat{\mathcal{H}}_8$ constraints.

Another possibility, would be to compute the strongest entailed relations (minimal relations) between each pair of variables before propagating constraints from one set to the other. However, this method has the disadvantage that it is computationally expensive, as the best known algorithm for computing the minimal network of a set of constraints over either $\widehat{\mathcal{H}}_8$, $\mathcal{C}_8$ or $\mathcal{Q}_8$ requires $O(n^5)$ time.

Finally, a third method could be based on iteratively using path-consistency as a preprocessing technique and then propagating the information from one set to the other. A similar method is used by Ladkin and Kautz to combine qualitative and metric constraints in the context of temporal reasoning [Kautz and Ladkin, 1991]. Note that imposing path-consistency is sufficient for consistency checking of a set of constraints over $\widehat{\mathcal{H}}_8$, $\mathcal{C}_8$, $\mathcal{Q}_8$, and $\mathcal{QS}$, but is incomplete for computing the minimal relations [van Beek, 1992; Renz and Nebel, 1999]. The following example shows that the information would need to be propagated more than once, and furthermore it is not clear whether in general this method would be complete for detecting inconsistency.

EXAMPLE 1.9 *Consider the set* Θ *formed by the following* $\widehat{\mathcal{H}}_8$ *constraints*

$x_0\{\mathsf{TPP},\mathsf{EQ}\}x_2$, $x_1\{\mathsf{TPP},\mathsf{EQ},\mathsf{PO}\}x_0$, $x_1\{\mathsf{TPP},\mathsf{EQ}\}x_2$, $x_4\{\mathsf{TPP},\mathsf{EQ}\}x_3$,

and the set Σ *formed by the of following* $\mathcal{QS}$*-constraints*

$$size(x_0) < size(x_2),\ size(x_3) \leq size(x_1),\ size(x_2) \leq size(x_4).$$

We have that Θ and Σ are independently consistent, but their union is not consistent. Moreover, the following propagation scheme does not detect the inconsistency: (a) enforce path-consistency to Σ and Θ independently; (b) extend Σ with the size constraints entailed by the constraints in Θ; (c) extend Θ with the topological constraints entailed by the constraints in Σ; (d) enforce path-consistency to Θ and Σ again. In order to detect that $\Theta \cup \Sigma$ is inconsistent, we need an additional propagation of constraints from the topological set to the size set.

Instead of directly analysing the complexity and completeness of the propagation scheme illustrated in the previous example, Gerevini and Renz [Gerevini and Renz, 2002] proposed a new method for dealing with combined topological and qualitative size constraints. In particular, they propose an $O(n^3)$ time and $O(n^2)$ space algorithm, BIPATH-CONSISTENCY, for imposing path-consistency to a set of constraints in RCC-8 $\cup$ $\mathcal{QS}$. BIPATH-CONSISTENCY solves CSPSAT for any input set Θ of topological constraints in either $\widehat{\mathcal{H}}_8$, $\mathcal{C}_8$ or $\mathcal{Q}_8$, combined with any set of size constraints in $\mathcal{QS}$ involving the variables of Θ. Thus, despite this framework is more expressive than a purely topological one over the same set of relations (and therefore has a larger potential applicability), the problem of deciding consistency can be solved without additional worst-case cost.

BIPATH-CONSISTENCY is a modification of Vilain and Kautz' path-consistency algorithm [Vilain and Kautz, 1986; Vilain et al., 1989] as described by Bessière [Bessière, 1996], which in turn is a slight modification of Allen's algorithm [Allen, 1983]. The main novelty of the algorithm is that BIPATH-CONSISTENCY operates on a graph of *pairs* of constraints. The vertices of the graph are constraint variables, which in our context correspond to spatial regions. Each edge of the graph is

```
Algorithm: BIPATH-CONSISTENCY
Input: A set $\Theta$ of RCC-8 constraints, and a set $\Sigma$ of $\mathcal{QS}$-constraints over
the variables $x_1, x_2, \ldots, x_n$ of $\Theta$.
Output: fail, if $\Sigma \cup \Theta$ is not consistent; path-consistent sets equivalent
to $\Sigma$ and $\Theta$, otherwise.

1. Q $\leftarrow \{(i,j) \mid i < j\}$;   ($i/j$ indicates the $i$-th/$j$-th variable of $\Theta$.)
2. while Q $\neq \emptyset$ do
3. select and delete an arc $(i,j)$ from Q;
4. for $k \neq i, k \neq j$ $(k \in \{1...n\})$ do
5.     if BIREVISION$(i,j,k)$ then
6.         if $R_{ik} = \emptyset$ then return fail
7.            else add $(i,k)$ to Q;
8.     if BIREVISION$(k,i,j)$ then
9.         if $R_{kj} = \emptyset$ then return fail
10.           else add $(k,j)$ to Q.
```

```
Function: BIREVISION$(i,k,j)$
Input: three region variables $i$, $k$ and $j$
Output: true, if $R_{ij}$ is revised; false otherwise.
Side effects: $R_{ij}$ and $R_{ji}$ revised using the operations $\cap$ and $\circ$ over the
              constraints involving $i$, $k$, and $j$.

1. if one of the following cases hold, then return false:
   (a) $Toprel(s_{ik}) \cap t_{ik} = U_t$ and $Sizerel(t_{ik}) \cap s_{ik} = U_s$,
   (b) $Toprel(s_{kj}) \cap t_{kj} = U_t$ and $Sizerel(t_{kj}) \cap s_{kj} = U_s$
2. oldt := $t_{ij}$; olds := $s_{ij}$;
3. $t_{ij} := (t_{ij} \cap Toprel(s_{ij})) \cap ((t_{ik} \cap Toprel(s_{ik})) \circ (t_{kj} \cap Toprel(s_{kj})))$;
4. $s_{ij} := (s_{ij} \cap Sizerel(t_{ij})) \cap ((s_{ik} \cap Sizerel(t_{ik})) \circ (s_{kj} \cap Sizerel(t_{kj})))$;
5. if $s_{ij} \neq$ olds then $t_{ij} := (t_{ij} \cap Toprel(s_{ij}))$;
6. if (oldt $= t_{ij}$) and (olds $= s_{ij}$) then return false;
7. $t_{ji} := Converse(t_{ij})$; $s_{ji} := Converse(s_{ij})$;
8. return true.
```

Figure 1.14. BIPATH-CONSISTENCY.

labelled by a pair of relations formed by a topological relation in RCC-8 and a size relation in $\mathcal{QS}$. The function BIREVISION(i,k,j) has the same role as the function REVISE used in path consistency algorithms for constraint networks (e.g., [Mackworth, 1977]). The main difference is that BIREVISION(i, k, j) considers pairs of (possibly interdependent) constraints, instead of single constraints.

A formal description of BIPATH-CONSISTENCY is given in Figure 1.14, where R_{ij} is a pair formed by a relation t_{ij} in RCC-8 and a relation s_{ij} in $\mathcal{QS}$; $R_{ij} = \emptyset$ when $t_{ij} = \emptyset$ or $s_{ij} = \emptyset$; U_t indicates the universal relation in RCC-8 and U_s the universal relation in $\mathcal{QS}$.

Gerevini and Renz [Gerevini and Renz, 2002] prove soundness and completeness of BIPATH-CONSISTENCY for the maximal tractable subsets of RCC-8 combined with qualitative size relations.

THEOREM 1.10 *Given a set Θ of constraints in either $\widehat{\mathcal{H}}_8$, $\mathcal{C}_8$ or $\mathcal{Q}_8$, and a set Σ of constraints in $\mathcal{QS}$ involving variables in Θ, consistency of $\Theta \cup \Sigma$ can be decided using the* BIPATH-CONSISTENCY *algorithm in $O(n^3)$ time and $O(n^2)$ space, where n is the number of variables involved in Θ and Σ.*

Using the BIPATH-CONSISTENCY algorithm combined sets of constraints can be solved in cubic time just like the normal path-consistency algorithm for each of the two sets alone, i.e., they can be solved without additional worst-case cost. Soundness and completeness of BIPATH-CONSISTENCY do not hold automatically and has to be proved for each combination of different relations anew. Sometimes, however, the computational properties of combined calculi can be more favourable than both of them alone if the interactions with the other type of relations refines relations that make deciding consistency NP-hard to relations for which it is tractable. As can be seen in Table 1.5, whenever we have a definite qualitative size constraint $size(x) S size(y)$ with $S \in \{<, >, =\}$ and this constraint is combined with an RCC-8 constraint xRy resulting in $xR'y$, then R' will contain basic relations of at most one of the sets $\{\mathsf{TPP}, \mathsf{NTPP}\}, \{\mathsf{TPP}^{-1}, \mathsf{NTPP}^{-1}\}, \{\mathsf{EQ}\}$ and possibly some relations of the set $\{\mathsf{DC}, \mathsf{EC}, \mathsf{PO}\}$. In other words, and relation of $R \in \text{RCC-8} \setminus \widehat{\mathcal{H}}_8$ will be refined to a relation $R' \in \widehat{\mathcal{H}}_8$ and therefore it is possible that sets of constraints for which it is NP-hard to decide consistency become tractable after adding constraints over a different set of relations.

The opposite is of course also possible, that combining two calculi results in a calculus that has a higher complexity than both alone. This is the case for another combination that Gerevini and Renz analysed, namely, combining RCC-8 with metric size information. They considered metric size constraints of different kinds, metric relative size constraints $size(x) R \alpha \cdot size(y)$ where α is a positive rational number, size difference constraints $size(x) - size(y) \in I$ where I is a continuous interval of rational numbers, or domain size constraints $size(x) \in I$. The main difference of combining RCC-8 with metric size information as compared

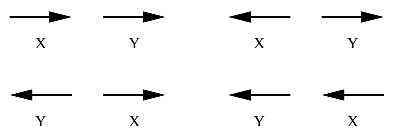

Figure 1.15. Four structurally different instantiations of the relation "x behind y" with directed intervals.

to qualitative size information is that it is possible to express that a set of regions completely fills another region. Combining RCC-8 with any of these metric size calculi, which are all independently tractable, leads to NP-hardness even when combined with only the RCC-8 basic relations. Without the PO relation, i.e., considering only the 7 other RCC-8 basic relations, the combination is tractable though.

This was an example of how different calculi can be combined. Future research effort within qualitative spatial representation and reasoning should deal with modularising different aspects and different granularities in a similar way that topology and size was combined here, studying their interactions and developing algorithms for reasoning about combined calculi. Then different applications could use the modules that are needed for the particular application, the interactions between the different modules and combine them using algorithms like Bipath-consistency. If possible, these combinations should have favourable computational properties and should enable efficient solutions.

7.3 Combining topological and directional information for intervals

In this section we give an example for a combination of two aspects of space that relies upon forming new relations out of two given sets of relations. One of the two aspects of space we are looking at is the interval algebra (IA) [Allen, 1983] which was originally defined for temporal reasoning. However, for applications that can require only a one-dimensional spatial representation, it makes sense to use the interval algebra. Some possible applications are from the area of traffic management. Roads and railway lines can be regarded as one-dimensional routes, but also air and sea traffic mainly operates on given routes. A single route can be represented as a one-dimensional space and the vehicles on a route as intervals. The main difference of vehicles on a route to the interval algebra is that vehicles and also routes have a direction. We therefore have to extend the interval algebra by adding direction in order to use it for traffic applications. Direction in a one-dimensional space is quite simple as there are only two directions, front and back, or same direction and different direction.

A straightforward way for dealing with directed intervals would be to add additional constraints on the direction of intervals to constraints over the Interval Algebra and treat the two types of constraints separately while propagating information from one type to the other (similar to what has been done in [Gerevini and Renz, 1998].) We say that an interval has *positive direction* if it has the same direction as the underlying line and *negative direction* otherwise. So possible direction constraints could be unary constraints like "x has positive/negative direction" or binary constraints like "x and y have the same/opposite direction". This approach, however, is not possible since the Interval Algebra loses its property of being a relation algebra when permitting directed intervals. This can be easily seen when considering the "behind" relation of Figure 1.15. The actual converse of "x behind y" is a subset of "y is behind or in front of x" which cannot be expressed within the Interval Algebra. If using "y is behind or in front of x" as the converse of "x behind y", whose converse is again "x is behind or in front of y", then applying the converse operation ($\cdot^{\smile}$) twice leads to a different relation than the original relation. This is a contradiction to one of the requirements of relation algebras ($R^{\smile\smile} = R$) [Ladkin and Maddux, 1994]. This contradiction does not occur when we refine the "behind" relation into two disjoint sub-relations "behind$_=$" and "behind$_{\neq}$" where the subscript indicates that both intervals have the same ($=$) or opposite ($\neq$) direction. The converse of both relations is "in-front-of$_=$" and "behind$_{\neq}$", respectively. Applying the converse operation again leads to the original relations.

Since a relation algebra must be closed under composition, intersection, and converse, we have to make the same distinction also for all other IA relations. This leads us to the definition of the directed intervals algebra (DIA). It consists of the 26 basic relations given in Table 1.6, which result from refining each IA relation into two sub-relations specifying either same or opposite direction of the involved intervals, and of all possible unions of the basic relations. This gives a total number of 2^{26} DIA relations. Converse relations are given in the same table entry. If a converse relation is not explicitly given, the corresponding relation is its own converse. We denote the set of 26 DIA basic relations as $\mathcal{B}$. Then DIA $= 2^{\mathcal{B}}$. Complex relations which are the union of more than one basic relation $R_1, \ldots, R_k$ are written as $\{R_1, \ldots, R_k\}$. The union of all basic relations, the universal relation, is denoted $\{*\}$.

A DIA basic relation $R = I_d$ consist of two parts, the interval part I which is a spatial interpretation of the Interval Algebra and the direction part d which gives the mutual direction of both intervals, either $=$ or $\neq$. If a complex relation R consist of basic relations with the same direction part d, we can combine the interval parts and write $R = \{I^1, \ldots, I^k\}_d$

Table 1.6. The 26 basic relations of the directed intervals algebra.

Directed Intervals Basic Relation	Sym-bol	Pictorial Example
x behind$_=$ y	b$_=$	-x->
y in-front-of$_=$ x	f$_=$	-y->
x behind$_\neq$ y	b$_\neq$	<-x- -y->
x in-front-of$_\neq$ y	f$_\neq$	-x-> <-y-
x meets-from-behind$_=$ y	mb$_=$	–x–>
y meets-in-the-front$_=$ x	mf$_=$	–y–>
x meets-from-behind$_\neq$ y	mb$_\neq$	<–x– –y–>
x meets-in-the-front$_\neq$ y	mf$_\neq$	–x–> <–y–
x overlaps-from-behind$_=$ y	ob$_=$	—x—>
y overlaps-in-the-front$_=$ x	of$_=$	—y—>
x overlaps-from-behind$_\neq$ y	ob$_\neq$	<—x— —y—>
x overlaps-in-the-front$_\neq$ y	of$_\neq$	—x—> <—y—
x contained-in$_=$ y	c$_=$	–x–>
y extends$_=$ x	e$_=$	—y—>
x contained-in$_\neq$ y	c$_\neq$	<–x–
y extends$_\neq$ x	e$_\neq$	—y—>
x contained-in-the-back-of$_=$ y	cb$_=$	–x–>
y extends-the-front-of$_=$ x	ef$_=$	—y—>
x contained-in-the-back-of$_\neq$ y	cb$_\neq$	<–x–
y extends-the-back-of$_\neq$ x	eb$_\neq$	—y—>
x contained-in-the-front-of$_=$ y	cf$_=$	–x–>
y extends-the-back-of$_=$ x	eb$_=$	—y—>
x contained-in-the-front-of$_\neq$ y	cf$_\neq$	<–x–
y extends-the-front-of$_\neq$ x	ef$_\neq$	—y—>
x equals$_=$ y	eq$_=$	—x—> —y—>
x equals$_\neq$ y	eq$_\neq$	—x—> <—y—

instead of $R = \{I_d^1, \ldots, I_d^k\}$. We write R_e (resp. R_n) in order to refer to the union of the interval parts of every sub-relation of a complex relation R where the direction part is $\{=\}$ (resp. $\{\neq\}$.) In this way, every DIA relation R can be written as $R = \{R_e\}_= \cup \{R_n\}_\neq$. DIA$_I$ denotes the set of 2^{13} possible interval parts of DIA relations.

Table 1.7. IA basic relations R, their reverses R^r, and their spatial interpretations dia(R).

R	$\prec$	$\succ$	m	mi	o	oi	s	si	d	di	f	fi	$\equiv$
R^r	$\succ$	$\prec$	mi	m	oi	o	f	fi	d	di	s	si	$\equiv$
dia(R)	b	f	mb	mf	ob	of	cb	ef	c	e	cf	eb	eq

It is important to note that the spatial interpretation of the Interval Algebra was chosen in a way that the interval part of a relation xI_dy only depends on the direction of y and not on the direction of x. Therefore, if the direction of x is reversed, written as $\overline{x}$, then only the direction part changes, i.e., $xI_dy = \overline{x}I_{\neg d}y$. This would not be the case in a straightforward spatial interpretation of the original temporal relations. For instance, IA relations like "x started-by y" or "x finished-by y" depend on the direction of x. Instead, we interpret these relations spatially as "x extends-the-front/back-of y" and "x contained-in-the-front/back-of y". This interpretation is independent of the direction of x. When all intervals have the same direction, both interpretations are equivalent. In order to transform the spatial and the temporal interval relations (independent of the direction of the intervals) into each other, we introduce two mutually inverse functions dia : IA $\mapsto$ DIA$_I$ and ia : DIA$_I$ $\mapsto$ IA, i.e., dia(ia(R)) $= R$ and ia(dia(R)) $= R$. The mapping is given in Table 1.7.

All relations of the directed intervals algebra are invariant with respect to the direction of the underlying line, i.e., when reversing the direction of the line, all relations remain the same. This is obviously not the case for the Interval Algebra, e.g., if x is before y and one reverses the direction of the time line, then x is after y. In order to transform DIA relations into the corresponding IA relations and *vice versa*, we introduce a unary *reverse* operator ($\cdot^r$) on relations R such that R^r specifies the relation which results from R when reversing the direction of the underlying line. For all relations $R \in$ DIA we have that $R^r = R$. For IA relations, the reverse relation is given in Table 1.7. The reverse of a complex relation is the union of the reverses of the involved basic relations. The reverse of the composition ($\circ$) of two relations is equivalent to the composition of the reverses of the two involved relations, i.e., $(R \circ S)^r = R^r \circ S^r$. Applying the reverse operator twice results in the original relation, i.e., $R^{rr} = R$. Using the reverse operator we can also specify what happens with a relation xI_dy if only the direction of y is changed. Then the topological relation of the intervals stays the same, but the order changes, i.e., "front" becomes "behind"/"back" and *vice* *versa*. The mutual direction also changes. This can be expressed in the following way: $xI_d y = x\,\mathsf{dia}(\mathsf{ia}(I)^r)_{\neg d}\,\overline{y}$.

We now have all requirements for computing the composition ($\circ_d$) of DIA relations using composition of IA relations (denoted here by $\circ_{ia}$) as specified by Allen [Allen, 1983].

Theorem 1.11 *Let R_p, S_q be DIA basic relations.*

1 If $q = \{=\}$, then $R_p \circ_d S_q = \mathsf{dia}(\mathsf{ia}(R) \circ_{ia} \mathsf{ia}(S))_p$

2 If $q = \{\neq\}$, then $R_p \circ_d S_q = \mathsf{dia}(\mathsf{ia}(R)^r \circ_{ia} \mathsf{ia}(S))_{\neg p}$

The composition of complex relations is as usual the union of the composition of the contained basic relations. It follows from the closedness of the Interval Algebra that DIA is closed under composition, intersection, converse, and reverse.

We have now obtained a new set of relations together with the necessary operations that cover the combination of the original two aspects. As opposed to the combination of topology and size, we can use the standard backtracking and path-consistency algorithms for reasoning about these relations. However, since we now have 26 basic relations, and 2^{26} possible subsets, it is much more difficult to analyse computational properties for these relations than it is for the interval algebra. Fortunately, it is partly possible to use the complexity results of the interval algebra in order to find complexity results for the directed interval algebra. NP-hardness of the directed intervals algebra follows immediately from NP-hardness of the interval algebra. But also tractable subsets can be identified by exploiting results for the interval algebra. Among them, most importantly, the set of DIA basic relations was shown to be tractable. In the proof of this result it is not important that all non-universal relations are basic relations, only that all non-universal relations consist of DIA basic relations with the same direction part. Therefore, tractability for the basic relations can be extend to the following result.

Theorem 1.12 *Let $\mathcal{S}$ be a tractable subset of the Interval Algebra which is closed under the reverse operator. Then $\mathcal{S}^{\pm} = \{\mathsf{dia}(R)_{=} | R \in \mathcal{S}\} \cup \{\mathsf{dia}(R)_{\neq} | R \in \mathcal{S}\} \cup \{*\}$ is a tractable subset of the directed intervals algebra.*

Renz [Renz, 2001] showed that ORD-Horn, the only maximal tractable subset of the interval algebra that contains all basic relations is closed under the reverse operator. Therefore, the above theorem also applies to ORD-Horn. What's more, it was shown that path-consistency decides consistency for $\mathcal{H}^{\pm}$, the set of DIA relations which results from ORD-Horn (see Theorem 1.12).

THEOREM 1.13 *Path-consistency decides* $\mathsf{CSPSAT}(\mathcal{H}^{\pm})$.

Apart from these initial results the complexity of the directed intervals algebra is open and maximal tractable subsets have not yet been identified.

8. Conclusions

In this chapter we have shown how spatial information can be represented using constraint calculi. This is a natural way of using qualitative spatial calculi and allows us to use methods and techniques developed for constraint satisfaction problems. We introduced qualitative spatial calculi, constraint satisfaction methods for reasoning over these calculi and presented methods for analysing the computational properties of spatial calculi. These methods are very general and can be used for different kinds of spatial calculi. We showed how a computational analysis can lead to very efficient reasoning even though the reasoning problems are NP-hard.

Using the presented methods we can specify a roadmap for how spatial calculi should be analysed in order to find efficient reasoning methods:

1 Define a useful set of basic relations $\mathcal{B}$ over a certain aspect of space, on a certain level of granularity and over a certain spatial domain.

2 Formally compute the composition table.

3 Try to find an NP-hardness proof for $\mathsf{CSPSAT}(2^{\mathcal{B}})$ using the method of polarity and clause constraints.

4 Try to show that $\mathsf{CSPSAT}(\mathcal{B})$ is tractable and that path-consistency decides consistency.

5 Identify larger tractable sets by applying the closure and the refinement methods. If possible identify maximal tractable subsets.

6 Using an empirical analysis, identify the combination of strategies that is most effective in solving instances $\mathsf{CSPSAT}(2^{\mathcal{B}})$.

These methods work for calculi based on a single aspect of space. For most practical applications, however, it is necessary to combine more than one aspect of space. In this chapter we presented some example of how relations can be combined. One way is to form new relations that cover the combined aspects, another way is to treat the relations separately and to keep track of and propagate their interactions like it is done by Gerevini and Renz' bipath-consistency algorithm.

It is one of the main challenges of qualitative spatial reasoning to provide general methods for combining different calculi and for analysing the computational properties of combined calculi. What we presented here is only a small step in this direction and lots of future research is required. The goal of such an analysis could be a toolbox of calculi for different spatial and also temporal aspects on different granularities and efficient algorithms for their combinations. Each application could then pick the required sets of relations and the most efficient algorithms for combining these relations.

2SAT —021 3SAT —021 9-intersection —013

Index

References

Achlioptas, D., Kirousis, L., Kranakis, E., Krizanc, D., Molloy, M., and Stamatiou, Y. (1997). Random constraint satisfaction: a more accurate picture. In *3rd Conference on the Principles and Practice of Constraint Programming (CP'97)*, volume 1330 of *Lecture Notes in Computer Science*, pages 107–120. Springer Verlag.

Allen, JamesF. (1983). Maintaining knowledge about temporal intervals. *Communications of the ACM*, 26(11):832–843.

Asher, Nicholas and Vieu, Laure (1995). Towards a geometry of common sense: A semantics and a complete axiomatization of mereotopology. In *Proceedings of the 14th International Joint Conference on Artificial Intelligence*, pages 846–852, Montreal, Canada.

Balbiani, Philippe, Condotta, Jean-François, and Farinasdel Cerro, Luis (1998). A model for reasoning about bidimensional temporal relations. In Cohn, A.G., Schubert, L., and Shapiro, S.C., editors, *Principles of Knowledge Representation and Reasoning: Proceedings of the 6th International Conference*, pages 124–130, Trento, Italy.

Balbiani, Philippe, Condotta, Jean-François, and Farinasdel Cerro, Luis (1999a). A new tractable subclass of the rectangle algebra. In *Proceedings of the 16th International Joint Conference on Artificial Intelligence*, pages 442–447, Stockholm, Sweden.

Balbiani, Philippe, Condotta, Jean-François, and Farinasdel Cerro, Luis (1999b). A tractable subclass of the block algebra: constraint propagation and preconvex relations. In *Proccedings of the 9th Portuguese Conference on Artificial Intelligence*, pages 75–89.

Bennett, Brandon (1994). Spatial reasoning with propositional logic. In Doyle, J., Sandewall, E., and Torasso, P., editors, *Principles of Knowledge Representation and Reasoning: Proceedings of the 4th International Conference*, pages 51–62, Bonn, Germany. Morgan Kaufmann.

Bessière, Christian (1996). A simple way to improve path-consistency in Interval Algebra networks. In *Proceedings of the 13th National Conference of the American Association for Artificial Intelligence*, pages 375–380, Portland, OR. MIT Press.

Biacino, Loredana and Gerla, Giangiacomo (1991). Connection structures. *Notre Dame Journal of Formal Logic*, 32(2):242–247.

Borgo, Stefano, Guarino, Nicola, and Masolo, Claudio (1996). A pointless theory of space based on strong connection and congruence. In Aiello, L.C., Doyle, J., and Shapiro, S.C., editors, *Principles of Knowledge Representation and Reasoning: Proceedings of the 5th International Conference*, pages 220–229, Cambridge, MA. Morgan Kaufmann.

Cheeseman, Peter, Kanefsky, Bob, and Taylor, WilliamM. (1991). Where the *really* hard problems are. In *Proceedings of the 12th International Joint Conference on Artificial Intelligence*, pages 331–337, Sydney, Australia. Morgan Kaufmann.

Clarke, BowmanL. (1981). A calculus of individuals based on connection. *Notre Dame Journal of Formal Logic*, 22(3):204–218.

Clarke, BowmanL. (1985). Individuals and points. *Notre Dame Journal of Formal Logic*, 26(1):61–75.

Clementini, Eliseo, diFelice, Paolino, and Hernandez, Daniel (1997). Qualitative representation of positional information. *Artificial Intelligence*, 95(2):317–356.

Cohn, AnthonyG., Bennett, Brandon, Gooday, John, and Gotts, NicholasM. (1997). Representing and reasoning with qualitative spatial relations about regions. In Stock, O., editor, *Spatial and Temporal Reasoning*, pages 97–134. Kluwer, Dordrecht, Holland.

Cohn, AnthonyG. and Varzi, AchilleC. (1998). Connection relations in mereotopology. In *Proceedings of the 13th European Conference on Artificial Intelligence*, pages 150–154, Amsterdam, The Netherlands. Wiley.

Cohn, AnthonyG. and Varzi, AchilleC. (1999). Modes of connection. In *Spatial Information Theory: Cognitive and Computational Foundations of Geographic Information Science*, pages 299–314.

Cook, StephenA. (1971). The complexity of theorem-proving procedures. In *Proc. 3rd Ann. ACM Symp. on Theory of Computing*, pages 151–158, New York. Association for Computing Machinery.

Cormen, ThomasH., Leiserson, CharlesE., and Rivest, RonaldL. (1990). *Introduction to Algorithms*. MIT Press.

Dornheim, Christoph (1998). Undecidability of plane polygonal mereotopology. In Cohn, A.G., Schubert, L., and Shapiro, S.C., editors, *Principles of Knowledge Representation and Reasoning: Proceedings of the 6th International Conference*, Trento, Italy.

Egenhofer, MaxJ. (1991). Reasoning about binary topological relations. In Günther, O. and Schek, H.-J., 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.

Egenhofer, MaxJ., Clementini, Eliseo, and Felice, PaolinoDi (1994). Topological relations between regions with holes. *International Journal of Geographical Information Systems*, 8(2):129–144.

Egenhofer, MaxJ. and Franzosa, RobertD. (1994). On the equivalence of topological relations. *International Journal of Geographical Information Systems*, 8(6):133–152.

Egenhofer, MaxJ. and Sharma, Jayant (1993). Assessing the consistency of complete and incomplete topological information. *Geographical Systems*, 1(1):47–68.

Forbus, KennethD., Nielsen, Paul, and Faltings, Boi (1987). Qualitative kinematics: A framework. In McDermott, J., editor, *Proceedings of the 10th International Joint Conference on Artificial Intelligence*, Milan, Italy. Morgan Kaufmann.

Frank, AndrewU. (1991). Qualitative spatial reasoning about cardinal directions. In *Proceedings of the 7th Austrian Conference on Artificial Intelligence*, pages 157–167.

Freksa, Christian (1992). Using orientation information for qualitative spatial reasoning. In A.U.Frank, I.Campari, U.Formentini, editor, *Theories and Methods of Spatio-Temporal Reasoning in Geographic Space*, volume 639 of *Lecture Notes in Computer Science*. Springer-Verlag, Berlin, Heidelberg, New York.

Galton, Antony (1999). Mereotopology of discrete space. In Freksa, C. and Mark, D.M., editors, *Spatial information theory: Cognitive and computational foundations of geographic information science*, volume 1661 of *Lecture Notes in Computer Science*, pages 251–266, Berlin, Heidelberg, New York. Springer Verlag.

Garey, MichaelR. and Johnson, DavidS. (1979). *Computers and Intractability—A Guide to the Theory of NP-Completeness*. Freeman, San Francisco, CA.

Gent, IanP. and Walsh, Toby (1996). The satisfiability constraint gap. *Artificial Intelligence*, 81(1–2):59–80.

Gerevini, Alfonso and Renz, Jochen (1998). Combining topological and qualitative size constraints for spatial reasoning. In *Proceedings of the 4th International Conference on Principles and Practice of Constraint Programming*, Pisa, Italy.

Gerevini, Alfonso and Renz, Jochen (2002). Combining topological and size information for spatial reasoning. *Artificial Intelligence*, 137(1-2):1–42.

Golumbic, MartinC. and Shamir, Ron (1993). Complexity and algorithms for reasoning about time: A graph-theoretic approach. *Journal of the Association for Computing Machinery*, 40(5):1128–1133.

Gotts, NicholasM. (1996). Using the RCC formalism to describe the topology of spherical regions. Technical Report 96-24, University of Leeds, School of Computer Studies.

Gotts, NicholasM., Gooday, JohnM., and Cohn, AnthonyG. (1996). A connection based approach to commonsense topological description and reasoning. *The Monist*, 79(1):51–75.

Goyal, Roop and Egenhofer, MaxJ. (to appear). Cardinal directions between extended spatial objects. *IEEE Transactions on Knowledge and Data Engineering*.

Grigni, Michelangelo, Papadias, Dimitris, and Papadimitriou, Christos (1995). Topological inference. In *Proceedings of the 14th International Joint Conference on Artificial Intelligence*, pages 901–906, Montreal, Canada.

Grzegorczyk, Andrzej (1951). Undecidability of some topological theories. *Fundamenta Mathematicae*, 38:137–152.

Guesgen, Hans (1989). Spatial reasoning based on Allen's temporal logic. Technical Report TR-89-049, ICSI, Berkeley, CA.

Hernàndez, Daniel (1994). *Qualitative Representation of Spatial Knowledge*, volume 804 of *Lecture Notes in Artificial Intelligence*. Springer-Verlag, Berlin, Heidelberg, New York.

Hernàndez, Daniel, Clementini, Eliseo, and diFelice, Paolino (1995). Qualitative distances. In *Spatial Information Theory: A Theoretical basis for GIS*, volume 988 of *Lecture Notes in Computer Science*, pages 45–58, Berlin, Heidelberg, New York. Springer-Verlag.

Hirsch, Robin (1999). A finite relation algebra with undecidable network satisfaction problem. *Bulletin of the IGPL*.

Hogg, Tad, Huberman, BernardoA., and (eds), Colin P.Williams (1996). Special volume on frontiers in problem solving: Phase transistions and complexity. *Artificial Intelligence*, 81(1–2).

Isli, Amar and Moratz, Reinhard (1999). Qualitative spatial representation and reasoning: Algebraic models for relative position. Technical Report 284, Universität Hamburg, Fachbereich Informatik.

Johnson, DavidS. (1990). A catalog of complexity classes. In van Leeuwen, J., editor, *Handbook of Theoretical Computer Science, Vol.A*, pages 67–161. MIT Press.

Kautz, HenryA. and Ladkin, PeterB. (1991). Integrating metric and qualitative temporal reasoning. In *Proceedings of the 9th National Conference of the American Association for Artificial Intelligence*, pages 241–246, Anaheim, CA. MIT Press.

Krokhin, AndreiA., Jeavons, Peter, and Jonsson, Peter (2003). Reasoning about temporal relations: The tractable subalgebras of allen's interval algebra. *Journal of the ACM*, 50(5):591–640.

Ladkin, PeterB. and Maddux, Roger (1994). On binary constraint problems. *Journal of the Association for Computing Machinery*, 41(3):435–469.

Ladkin, PeterB. and Reinefeld, Alexander (1992). Effective solution of qualitative interval constraint problems. *Artificial Intelligence*, 57(1):105–124.

Ligozat, Gerard (1996). A new proof of tractability for Ord-Horn relations. In *Proceedings of the 13th National Conference of the American Association for Artificial Intelligence*, pages 715–720, Portland, OR. MIT Press.

Ligozat, Gerard (1998). Reasoning about cardinal directions. *Journal of Visual Languages and Computing*, 9:23–44.

Mackworth, AlanK. (1977). Consistency in networks of relations. *Artificial Intelligence*, 8:99–118.

Mackworth, AlanK. and Freuder, EugeneC. (1985). The complexity of some polynomial network consistency algorithms for constraint satisfaction problems. *Artificial Intelligence*, 25:65–73.

Montanari, Ugo (1974). Networks of constraints: fundamental properties and applications to picture processing. *Information Science*, 7:95–132.

Montello, DanielR. (1993). Scale and multiple psychologies of space. In Frank, A.U. and Campari, I., editors, *Spatial information Theory: A theoretical basis for GIS*, volume 716 of *Lecture Notes in Computer Science*, pages 312–321, Berlin, Heidelberg, New York. Springer Verlag.

Nebel, Bernhard (1997). Solving hard qualitative temporal reasoning problems: Evaluating the efficiency of using the ORD-Horn class. *CONSTRAINTS*, 3(1):175–190.

Nebel, Bernhard and Bürckert, Hans-Jürgen (1995). Reasoning about temporal relations: A maximal tractable subclass of Allen's interval algebra. *Journal of the Association for Computing Machinery*, 42(1):43–66.

Papadias, Dimitris and Theodoridis, Yannis (1997). Spatial relations, minimum bounding rectangles, and spatial data structures. *International Journal of Geographic Information Systems*, 11(2):111–138.

Papadimitriou, ChristosH. (1994). *Computational Complexity*. Addison-Wesley, Reading, MA.

Piaget, Jean and Inhelder, Bärbel (1948). *La représentation de l'espace chez l'enfant*. Presses universitaires de France, Paris, France.

Pratt, Ian and Schoop, Dominik (1998). A complete axiom system for polygonal mereotopology of the real plane. *Journal of Philosophical Logic*, 27:621–658.

Pujari, ArunK., Kumari, G.Vijaya, and Sattar, Abdul (1999). INDU: An interval and duration network. In *Australian Joint Conference on Artificial Intelligence*, pages 291–303.

Randell, DavidA. and Cohn, AnthonyG. (1989). Modelling topological and metrical properties in physical processes. In Brachman, R., Levesque, H.J., and Reiter, R., editors, *Principles of Knowledge Representation and Reasoning: Proceedings of the 1st International Conference*, pages 55–66, Toronto, ON. Morgan Kaufmann.

Randell, DavidA., Cui, Zhan, and Cohn, AnthonyG. (1992). A spatial logic based on regions and connection. In Nebel, B., Swartout, W., and Rich, C., editors, *Principles of Knowledge Representation and Reasoning: Proceedings of the 3rd International Conference*, pages 165–176, Cambridge, MA. Morgan Kaufmann.

Renz, Jochen (1998). A canonical model of the Region Connection Calculus. In Cohn, A.G., Schubert, L., and Shapiro, S.C., editors, *Principles of Knowledge Representation and Reasoning: Proceedings of the 6th International Conference*, pages 330 – 341, Trento, Italy.

Renz, Jochen (1999). Maximal tractable fragments of the Region Connection Calculus: A complete analysis. In *Proceedings of the 16th International Joint Conference on Artificial Intelligence*, pages 448–454, Stockholm, Sweden.

Renz, Jochen (2001). A spatial odyssey of the interval algebra: 1.directed intervals. In *Proceedings of the 17th International Joint Conference on Artificial Intelligence*, pages 51–56, Seattle, WA.

Renz, Jochen (2002). *Qualitative Spatial Reasoning with Topological Information*, volume 2293 of *Lecture Notes in Artificial Intelligence*. Springer Verlag, Berlin, Heidelberg, New York.

Renz, Jochen and Mitra, Debasis (2004). Qualitative direction calculi with arbitrary granularity. In *PRICAI 2004: Trends in Artificial Intelligence, 8th Pacific Rim International Conference on Artificial Intelligence*, pages 65–74.

Renz, Jochen and Nebel, Bernhard (1999). On the complexity of qualitative spatial reasoning: A maximal tractable fragment of the Region Connection Calculus. *Artificial Intelligence*, 108(1-2):69–123.

Renz, Jochen and Nebel, Bernhard (2001). Efficient methods for qualitative spatial reasoning. *Journal of Artificial Intelligence Research*, 15:289–318.

Schaefer, ThomasJ. (1978). The complexity of satisfiability problems. In *Proc. 10th Ann. ACM Symp. on Theory of Computing*, pages 216–226, New York. Association for Computing Machinery.

Schoop, Dominik (1999). *A model-theoretic approach to mereotopology.* PhD thesis, Faculty of Science and Engineering, University of Manchester.

Scivos, Alexander and Nebel, Bernhard (2001). Double-crossing: Decidability and computational complexity of a qualitative calculus for navigation. In *Spatial Information Theory: Foundations of Geographic Information Science*.

Skiadopoulos, Spiros and Koubarakis, Manolis (2005). On the consistency of cardinal direction constraints. *Artificial Intelligence*, 163(1):91–135.

Smith, TerenceR. and Park, KeithK. (1992). Algebraic approach to spatial reasoning. *International Journal of Geographic Information Systems*, 6(3):177–192.

Tarski, Alfred (1941). On the calculus of relations. *Journal of Symbolic Logic*, 6:73–89.

van Beek, Peter (1992). Reasoning about qualitative temporal information. *Artificial Intelligence*, 58(1-3):297–321.

van Beek, Peter and Manchak, DennisW. (1996). The design and experimental analysis of algorithms for temporal reasoning. *Journal of Artificial Intelligence Research*, 4:1–18.

Vilain, MarcB. and Kautz, HenryA. (1986). Constraint propagation algorithms for temporal reasoning. In *Proceedings of the 5th National Conference of the American Association for Artificial Intelligence*, pages 377–382, Philadelphia, PA.

Vilain, MarcB., Kautz, HenryA., and van Beek, Peter (1989). Constraint propagation algorithms for temporal reasoning: A revised report. In Weld, D.S. and deKleer, J., editors, *Readings in Qualitative Reasoning about Physical Systems*, pages 373–381. Morgan Kaufmann, San Mateo, CA.

Whitehead, AlfredN. (1929). *Process and Reality.* The MacMillan Company, New York.