An Automatic Decomposition Method for Qualitative Spatial and Temporal Reasoning

Julien Hué and Matthias Westphal and Stefan Wölfl
Institut für Informatik
Albert-Ludwigs-Universität Freiburg
Georges-Köhler-Allee 52
79110 Freiburg, Germany
{hue,westpham,woelfl}@informatik.uni-freiburg.de

***Abstract*—Qualitative spatial and temporal reasoning is a research field that studies relational, constraint-based formalisms for representing, and reasoning about, spatial and temporal information. The standard approach for checking consistency is based on an exhaustive representation of possible configurations between three entities, the so-called composition tables. These tables, however, encode semantic background knowledge in a redundant way, which becomes a size and efficiency issue, when the composition table needs to be grounded as done in SAT encodings of problem instances. In this paper, we present a new framework that allows for decomposing composition tables into logically simpler parts, while preserving logical equivalence, e.g., the decomposition in start- and end-points for Allen's Interval Calculus. We show that finding such decompositions is an NP-complete problem and present a SAT-based method to generate decompositions. Finally, we discuss the impact of our decomposition method on SAT encodings of problem instances, and present a reasoning system built on decompositions that compares favorably with state-of-the-art solvers.**

I. Introduction

Qualitative spatial and temporal reasoning (QSTR) deals with knowledge representation formalisms that are close to human conceptualizations of space and time. Since spatial and temporal aspects are usually concerned with infinite, continuous domains, the key idea in QSTR is to only reason about qualitative descriptions of entity configurations, i.e., one employs relational schemas that allow for describing such configurations via an abstraction from concrete numerical data. On the symbolic level, relations between entities are represented by using a fixed, finite vocabulary of qualitative terms, such as "to the left of" or "in front of". Examples of such qualitative formalisms include the Point Algebra [23], Allen's Interval Algebra [1], and several Region Connection Calculi [19], [8].

The common approach in QSTR is to study constraint satisfaction problems over languages that use such qualitative relations in order to express (spatial or temporal) constraints between entities. Semantic background knowledge on the considered domain is recorded in so-called composition tables, which from a logical point of view can be regarded as inference rules. More precisely, composition tables encode knowledge about possible configurations between three entities in a form close to transitivity rules. For example, if x happened before y and y happened at the same time as z, then x must have happened before z as well. Since composition rules only restrict relations between three entities, one can apply them to all triples of variables in a qualitative problem instance in a way that is similar to the path consistency algorithm known in the context of constraint satisfaction problems on finite domains, i.e., these rules are used to refine relations that might hold between the considered entities. For a more comprehensive and detailed introduction we refer to [21].

Composition rules as listed in a composition table can therefore be directly applied in other logic programming paradigms, e.g., in boolean satisfiability (SAT) or Answer Set Programming (ASP). However, we need to restrict relations between all triples of entities, and this can often only be achieved by explicitly repeating rules from the composition table. Representing composition directly in propositional logic thus results in high space requirements (e.g., when using SAT encodings of qualitative constraint satisfaction problems [24]). One way to tackle this issue is to decompose the problem into smaller networks by using a divide-and-conquer method as presented in [13]. A drawback of this method is that it is dependent on the actual structure of the network. A complementary and more general idea would be to exploit underlying structures of a qualitative representation to replace the composition table with a smaller, simpler, but semantically equivalent rule set. Examples of this approach can be found throughout the field of qualitative reasoning such as the representation of Allen's Interval Algebra and its composition rules based on start and endpoints of intervals [16], [17].

In this paper, we present a novel method to automatically derive such representations from any composition table. The core idea is to decompose the symbolic qualitative representation into simpler relations and then the composition table into axioms (which we call compositional inferences) over these new relations. We here compute this decomposition based on a SAT program whose output provides the definition of a new representation and the corresponding compositional inferences. Furthermore, we present a new SAT encoding of the constraint satisfaction problem based on decompositions which compares favorably with the known encodings such as the one presented in [17]. We also implemented a prototype solver based on the decompositions provided. This implementation shows a good behavior when compared to the state-of-the-art solver GQR [10].

The paper is organized as follows. After a reminder about notation and basic concepts in QSTR, we provide formal definitions of the representation of a qualitative calculus and its decomposition. We then study the complexity of finding decompositions and introduce a SAT program which can compute decompositions. Then, we present decompositions of several qualitative calculi that we have obtained with our method, explain the new encoding proposal and the prototype implementation of our new solver, and report on the experimental results of our comparison.

II. Background

We briefly recall basic concepts used in qualitative constraint satisfaction.

Given a finite set of m variables $\mathcal{V} = \{v_1, \dots, v_m\}$ and 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 from $\mathcal{V}$ written as $R_i(v_{i_1}, \dots, v_{i_n})$. An assignment f is a partial function from the set of variables $\mathcal{V}$ to the set of values $\mathcal{D}$. An assignment f is said to satisfy a constraint $R_i(v_{i_1}, \dots, v_{i_n})$ iff $\langle f(v_{i_1}), \dots, f(v_{i_n})\rangle \in R_i$. A constraint network is a tuple $\langle \mathcal{V}, \mathcal{D}, \Theta\rangle$ where $\mathcal{V}$ is a set of variables, $\mathcal{D}$ the domain, and Θ a set of constraints defined on $\mathcal{D}$. The satisfiability problem is to find a *solution* of a given constraint network, i.e., an assignment of all variables to values such that all constraints in Θ are satisfied.

In QSTR, knowledge is usually represented as constraint networks, where the relations in constraints express knowledge about objects from an infinite domain. In the context of temporal reasoning, for example, we can assume $\mathcal{D}$ to be the set of rational numbers $\mathbb{Q}$, and represent that a happened before b by posing a constraint $<(a, b)$ where $<$ represents the smaller relation on $\mathbb{Q}^2$. Since the domain is infinite, we cannot express the relation as a table of valid tuples and instead we are required to introduce a symbolic representation scheme.

Definition 1 ([15]). *Let $\mathcal{D}$ be a non-empty domain. A* partition scheme *is a partition of $\mathcal{D}^2$ into a finite family $\mathcal{B}$ of non-empty binary relations (called* base *or* atomic relations*) on $\mathcal{D}$ such that $\mathcal{B}$ contains the identity relation on $\mathcal{D}^2$ and is closed under converses (i.e., $\forall B \in \mathcal{B} : \exists B' \in \mathcal{B} : B' = \{(x_2, x_1) \mid (x_1, x_2) \in B\}$).*

Moreover, qualitative information is often imprecise in the sense that we do not know exactly which base relation holds. That is, we want to express constraints of the form $B_{i_1}(x, y) \vee \dots \vee B_{i_k}(x, y)$.

Definition 2. *Given a partition scheme $\mathcal{B}$, a* qualitative constraint network *is a constraint network $\langle \mathcal{V}, \mathcal{D}, \Theta\rangle$, where Θ only contains constraints on binary relations that are unions of base relations from $\mathcal{B}$.*

To reason and refine such disjunctions, we consider the natural Boolean algebra defined on $2^{\mathcal{B}}$ (i.e., the elements of $\mathcal{B}$ can be conceived of as the atoms of this Boolean algebra with intersection, union, and complement as operations). On top of this Boolean algebra one can define a relation algebra-like structure if one adds a converse operation (the converse of a set of base relations is defined simply as the set of the converses of the base relations) and a composition operation that approximates the set-theoretical composition of binary relations by relations from the partition scheme. That is, we define $B \circ B' := \{B'' \in \mathcal{B} \mid \exists x_1, x_2, x_3 \in \mathcal{D} : B''(x_1, x_3) \wedge B(x_1, x_2) \wedge B'(x_2, x_3)\}$ and then extend this operation $\circ\colon \mathcal{B} \times \mathcal{B} \to 2^{\mathcal{B}}$ to an operation $\circ\colon 2^{\mathcal{B}} \times 2^{\mathcal{B}} \to 2^{\mathcal{B}}$, by setting $R \circ R' := \bigcup_{B \in R, B' \in R'} B \circ B'$.

Since this relation-algebraic structure is determined by the set of base relations as well as the converse and composition functions defined for base relations (these functions are typically presented as tables), the tuple $\mathcal{C} = (\mathcal{B}, \circ, \cdot^{\smile})$ is sufficient to specify a set of inference rules, which is usually referred to as a *qualitative calculus*. For example, an entry $B_1 \circ B_2 = \{B'_1, \dots, B'_k\}$ of the composition table can be read as the (universally quantified) rule $B_1(v_1, v_2) \wedge B_2(v_2, v_3) \to B'_1(v_1, v_3) \vee \dots \vee B'_k(v_1, v_3)$.

We will assume that qualitative constraint networks are *normalized* in that Θ contains exactly one constraint for each pair of variables v_i, v_j with $i < j$. Note that $2^{\mathcal{B}}$ is closed under intersection and converse, and that the universal relation can be simply expressed by the union of all base relations.

In general, the problem of deciding whether a qualitative constraint networks is satisfiable is undecidable [12]. Contrary to that, the problem of deciding whether a qualitative constraint network is consistent, is in NP. Note that consistency here refers to the set of inference rules of the qualitative calculus at hand. More precisely, a qualitative constraint network $\langle \mathcal{V}, \mathcal{D}, \Theta\rangle$ is said to be *inconsistent* if there exist variables $v_1, v_2 \in \mathcal{V}$ such that a constraint with the empty relation $\emptyset(v_1, v_2)$ can be derived from the network. This can be restated as in the following definition. An atomic refinement Θ' of a (normalized) constraint network Θ is a constraint network such that $\forall R_i(v_{i_1}, v_{i_2}) \in \Theta\ \exists B_i(v_{i_1}, v_{i_2}) \in \Theta' : B_i \subseteq R_i$

Definition 3. *A qualitative constraint network $\langle \mathcal{V}, \mathcal{D}, \Theta\rangle$ is* consistent *iff (i) there exists an* atomic refinement *Θ' of the (normalized) constraints Θ, and (ii) Θ' is* closed under composition*: $\forall R(v_{i_1}, v_{i_3}),\ R'(v_{i_1}, v_{i_2}),\ R''(v_{i_2}, v_{i_3}) \in \Theta' : R \subseteq R' \circ R''$.*

As every relation is a union of base relations, this decision problem is approached by searching an atomic refinement of every relation while applying composition to every triple of variables as a constraint propagation method. We can think of this as a finite constraint network on the domain $\mathcal{D}' = \mathcal{B}$, with variables $\mathcal{V}' = \{\, (v_i, v_j) \in \mathcal{V}^2 \mid i < j \,\}$, and a constraint set $\Theta' = \{R_\circ\big((v_i, v_k), (v_i, v_j), (v_j, v_k)\big) \mid 1 \le i < j < k \le |\mathcal{V}|\}$, where $R_\circ = \{\ (B_3, B_1, B_2) \mid B_3 \in (B_1 \circ B_2)\ \} \subseteq \mathcal{B}^3$ enforces inference through the composition function. Elements of this relation are called composition triads [14]. With this formulation, typical qualitative reasoners establish generalized arc consistency during search (for details see, e.g., [6], [24]). For the remainder of this paper we will only discuss the consistency problem.

An important family of qualitative calculi is the Region Connection Calculi (RCC) which comprises different formalisms for reasoning about topological relations between spatially extended objects (called *regions*) [19], [18], [7]. The

$\circ$	$DR(b,c)$	$PO(b,c)$	$EQ(b,c)$	$PP(b,c)$	$PC(b,c)$
$DR(a,b)$	$*$	DR PO PP	DR	DR PO PP	DR
$PO(a,b)$	DR PO PC	$*$	PO	PO PP	DR PO PC
$EQ(a,b)$	DR	PO	EQ	PP	PC
$PP(a,b)$	DR	DR PO PP	PP	PP	$*$
$PC(a,b)$	DR PO PC	PO PC	PC	PO EQ PP PC	PC

Fig. 1. The composition table of RCC5.

theory of Region Connection Calculi is built on a binary connectedness predicate and can be cast as a first-order theory. Different fragments can be obtained depending on further distinctions being made (e.g., distinctions between interior, bounding points, or the convex hull of regions). Regions are typically interpreted as regular open (or regular closed) sets of a topological space, and we do so here as well, when we illustrate regions by simple regions in $\mathbb{R}^2$.

1) RCC5: The simplest of the RCC is RCC5 [2] which deals with relations that can be defined in terms of parthood of simple regions or closed disks. The language of RCC5 contains five binary relations $\mathcal{B} = \{DR, EQ, PO, PP, PC\}$ with the following interpretation: $DR(a,b)$ means that a and b are completely disconnected; $EQ(a,b)$ means that a and b are identical; $PO(a,b)$ means that a and b partially overlap; $PP(a,b)$ means that a is a proper part of b; $PC(a,b)$ means that b is a proper part of a. The composition table for RCC5 can be seen in Figure 1.

Beside its definition thanks to the composition table, RCC5 can also be defined by a translation into predicate logic given in [19]. For this, let $C(a,b)$ be a formula representing connectedness meaning that a and b share a common point. In particular, $C(a,b)$ respects symmetry and reflexivity. For convenience, we denote by $P(a,b)$ the fact that a is a part of b. Formally, $P(a,b) := \forall z, C(z,a) \rightarrow C(z,b)$ and $Pi(a,b) := P(b,a)$ The different relations between objects of space can be defined by:

- $DR(a,b)$ iff $\neg P(a,b) \wedge \neg Pi(a,b) \wedge \neg C(a,b)$;
- $PP(a,b)$ iff $P(a,b) \wedge \neg Pi(a,b) \wedge C(a,b)$;
- $PC(a,b)$ iff $\neg P(a,b) \wedge Pi(a,b) \wedge C(a,b)$;
- $PO(a,b)$ iff $\neg P(a,b) \wedge \neg Pi(a,b) \wedge C(a,b)$;
- $EQ(a,b)$ iff $P(a,b) \wedge Pi(a,b) \wedge C(a,b)$.

2) RCC8: When considering the border of regions, we can split DR into DC (DisConnected) and EC (Externally Connected), PP into TPP (Tangential Proper Part) and $NTPP$ (Non-Tangential Proper Part), and the converse relation PC into $TPPI$ and $NTPPI$. Formally, one can describe these 8 relations of RCC8 as in [21]: $EQ(a,b)$ iff $P(a,b) \wedge P(b,a)$; $DC(a,b)$ iff $\neg C(a,b)$; $EC(a,b)$ iff $C(a,b) \wedge \neg\exists z(P(z,a) \wedge P(z,b))$; $PO(a,b)$ iff $\neg P(a,b) \wedge \neg P(b,a) \wedge \neg\exists z(P(z,a) \wedge P(z,b))$; $TPP(a,b)$ iff $PP(a,b) \wedge \exists z(EC(z,a) \wedge EC(z,b))$; $NTPP(a,b)$ iff $PP(a,b) \wedge \neg TPP(a,b)$; $TPPI(a,b)$ iff $TPP(b,a)$ and $NTPPI(a,b)$ iff $NTPP(b,a)$ where $PP(a,b)$ is described as $P(a,b) \wedge \neg P(b,a)$.

III. Decomposition of Qualitative Calculi

The introduced representations of RCC5 and RCC8 were directly stemming from their definition. As less inference rules lead to faster reasoning we naturally should ask the questions: Are these the smallest possible representations? And how can we obtain compositional inferences with such representations? In the following we treat base relations as predicates between two variables. The key idea we pursue is to rewrite base relation in a different predicate scheme with correct inference rules. For this we identify a new set of binary predicates $\mathcal{A}$ s.t. every base relation predicate can be represented by a conjunction of (possibly negated) predicates from $\mathcal{A}$. In a second step, the task is to generate a set of inference rules over $\mathcal{A}$ which is equivalent to the composition table.

A. Representations, Inferences, and Decompositions

In the following we assume that we are given a fixed qualitative calculus $\mathcal{C} = \langle \mathcal{B}, \circ, \cdot^{\smile} \rangle$ with base relations $\mathcal{B}$ and a set of binary predicates $\mathcal{A} = \{A_1, \ldots, A_n\}$. Within our framework, the decomposition of a qualitative calculus is represented by a set of rules which write base relation predicates as conjunctions of predicates from $\mathcal{A}$ with negation, and a set of compositional inferences based on this new representation that replicates the composition table. For the first part, we define a representation of base relations over $\mathcal{A}$. We denote as $L^i(a,b)$ either $A_i(a,b)$ or $\neg A_i(a,b)$ for some $A_i \in \mathcal{A}$.

Definition 4. *A* representation T_B *of a base relation* $B \in \mathcal{B}$ *is a conjunction* $T_B(a,b) := \bigwedge_{i=1}^{n} L_B^i(a,b)$.

We are only interested in representing all base relations in a way that allows us to distinguish them. In such cases, we consider a representation to be valid.

Definition 5. *Let* $T_B(a,b)$ *be a representation of* B *for each base relation. A* valid representation of all base relations *is a set* $\mathcal{T} = \{T_B \mid B \in \mathcal{B}\}$, *s.t. there is no* $T_{B_1}, T_{B_2} \in \mathcal{T}$ *where* $B_1 \neq B_2$ *and* $\forall 1 \leq i \leq n, L_{B_1}^i = L_{B_2}^i$.

Example 1. *We illustrate the notion of representation with the Point Algebra. The base relations of the Point Algebra are* $\mathcal{B} = \{<, =, >\}$. *They can be represented with predicates* $\mathcal{A} = \{A_1, A_2\}$ *where* $\mathcal{T} = \{T_< := A_1 \wedge \neg A_2, T_> := \neg A_1 \wedge A_2, T_= := A_1 \wedge A_2\}$

To add reasoning capabilities, we further need inference rules that mimic the information of the composition table. We use compositional inferences for this purpose.

Definition 6. *A* compositional inference *is a logic statement of the form* $\forall a,b,c \in \mathcal{V}, L^{i_1}(a,b) \wedge L^{i_2}(b,c) \rightarrow L^{i_3}(a,c)$ *with* $i_1, i_2, i_3 \in \{1, \ldots, n\}$.

Compositional inferences are used to rule out impossible configurations of three objects. It could have been written as a set of nogoods, i.e., $L^{i_1}(a,b) \wedge L^{i_2}(b,c) \wedge L^{i_3}(a,c)$, which would have been logically equivalent but there would have been a loss of efficiency in the computation (a nogood would be equivalent to 6 compositional inferences and minimal decompositions are not always of size which are multiple of 6).

For some predicate schemes $\mathcal{A}$ we will not find a valid representation. Further, for some we can find a valid representation, but no suitable set of compositional inferences. Here we are only interested in obtaining representation with sets of compositional inferences that are equivalent to the composition table.

Definition 7. *Let $\mathcal{T}$ be a valid base relation representation. A set of compositional inferences $\mathcal{N}$ is: (i) sound, if for any $L^{i_1}(a,b) \wedge L^{i_2}(b,c) \rightarrow L^{i_3}(a,c) \in \mathcal{N}$ there is no composition triad (B_3, B_1, B_2) with conjuncts L^{i_1} in T_{B_1}, L^{i_2} in T_{B_2} and $\neg L^{i_3}$ in T_{B_3}; (ii) complete, if for any triple (B_3, B_1, B_2) that is not a composition triad, there are conjuncts L^{i_1} in T_{B_1}, L^{i_2} in T_{B_2} and $\neg L^{i_3}$ in T_{B_3}, s.t. $L^{i_1}(a,b) \wedge L^{i_2}(b,c) \rightarrow \neg L^{i_3}(a,c)$ is in $\mathcal{N}$.*

By abuse, we will sometimes denote $A_i(a,b)$ by $A^i(a,b)$.

Example 2. *We continue Example 1, where $\mathcal{T}$ represented the Point Algebra. We can define a set of compositional inferences over elements of $\mathcal{A}$. Namely: $A_2(a,b) \wedge A_2(b,c) \rightarrow A_2(a,c)$ meaning that $a \geq b$ and $b \geq c$ entails $a \geq c$, $A_1(a,b) \wedge A_1(b,c) \rightarrow A_1(a,c)$ meaning that $a \leq b$ and $b \leq c$ entails $a \leq c$, and also $A_1(a,b) \wedge \neg A_2(b,c) \rightarrow \neg A_2(a,c)$, $A_2(a,b) \wedge \neg A_1(b,c) \rightarrow \neg A_1(a,c)$, $\neg A_1(a,b) \wedge A_2(b,c) \rightarrow \neg A_1(a,c)$ and $\neg A_2(a,b) \wedge A_1(b,c) \rightarrow \neg A_2(a,c)$.*

Finally, with these concepts we can define decompositions of a qualitative calculus.

Definition 8. *A* decomposition *of a qualitative calculus is a tuple $\langle \mathcal{T}, \mathcal{N} \rangle$, where $\mathcal{T}$ is a valid representation of all base relations and $\mathcal{N}$ is a complete and sound set of compositional inferences.*

Further, for any qualitative calculus there always exists a decomposition.

Proposition 1. *Let $\mathcal{C}$ be a qualitative calculus, then there exist a valid representation $\mathcal{T}$ and a complete and sound set of compositional inferences $\mathcal{N}$ for $\mathcal{C}$.*

Proof: It is easy to construct a valid representation by choosing $|\mathcal{A}| = |\mathcal{B}|$ where $T_{B_i} = (\bigwedge_{j \neq i} \neg A_j) \wedge A_i$. We then construct compositional inferences from those triples of base relations that are not composition triads. For instance, if $B_{i_3} \notin (B_{i_1} \circ B_{i_2})$ then $A_{i_1}(a,b) \wedge A_{i_2}(b,c) \rightarrow \neg A_{i_3}(a,c)$ is an inference. ∎

We will now address the question of a minimal representation and decomposition by means of automatic computations whose solutions are possible decompositions that correctly represent the qualitative calculus.

B. Computing Optimal Decompositions

A reasoning process based on decomposition would be very dependent on the size of this decomposition. The most important factor is of course the number of literals used as it dictates the size of the search space. The other important factor is the number of decomposition inferences as the fewer, the faster will be the propagation.

As minimality is important for reasoning efficiency, we considered several different approaches for finding decompositions: brute force search, SAT, and ASP. The SAT approach turned out to be the most practical one. We will give some theoretical results first, then detail the SAT encoding before showing some of the running times of these different approaches, and explaining the obtained decomposition.

Proposition 2. *The decision problem whether there exists a decomposition of a calculus $\mathcal{C}$ with k literals and n compositional inferences is NP-complete.*

Proof: The NP-membership is easy to verify. One can verify the sound and completeness of a set of compositional inference thanks to an ASP program which is positive-order consistent and of polynomial size, which means it is a polynomial operation. It is possible to prove the NP-hardness thanks to a reduction from the Hitting Sets problem [20]. The proof is in two steps. The lack of space prevents us for going into too much details.

Firstly, consider a problem we can call Disconnected Minimal Hitting Sets defined the following way: Given a set $A = \{a_1, \ldots, a_n\}$, a collection $B_1, B_2, \ldots, B_m$ of subsets of A, a collection $C_1, C_2, \ldots, C_m$ of subsets of A, and a number k. There exists a Hitting Set $H \subseteq A$ such that the number of C_i with $C_i \cap H \neq \emptyset$ is equal to k and $H \cup B_i \neq \emptyset, 1 \leq i \leq m$.

It is similar to the minimal Hitting Sets problem except for the value to minimize which is not the number of elements of H but the number of C_i where elements of H appear. It is very easy to see that Minimal Hitting Sets is just a special case of Disconnected Minimal Hitting Sets and that it is thus NP-hard.

Secondly, we can formalize the problem of finding a minimal decomposition for a calculus as a Disconnected Minimal Hitting Sets. We denote by $R = \{r_1, \ldots, r_m\}$ the possible representations for a literals. The set A is the set of compositional inferences built over the r_i. The B_i are the elements of the composition table. The C_j are the r_i. With this encoding, we try to find a set which covers every entry of the composition table while minimizing the number of compositional inferences used to do so. ∎

1) Decompositions as a SAT problem: We encode the decision problem for the fixed number of literals k. Firstly, we require solutions to be valid representations. To this end we assert that two relations are not represented the same way. This verification is made in two steps: firstly if two relations B_{j_1} and B_{j_2} are represented the same way for one literal A_i, then the atom $j_1\_j_2\_id\_on\_lit\_i$ is entailed. Second, we assert for any pair of base relations at least one literal of their representation is different.

$$\forall i \in 1..|\mathcal{A}|, \forall j_1, j_2 \in 1..|\mathcal{B}|,$$
$$\begin{cases} \neg A\_i\_j_1 \vee \neg A\_i\_j_2 \vee j_1\_j_2\_id\_on\_lit\_i \\ A\_i\_j_1 \vee A\_i\_j_2 \vee j_1\_j_2\_id\_on\_lit\_i \\ \neg j_1\_j_2\_id\_on\_lit\_1 \vee \cdots \vee \neg j_1\_j_2\_id\_on\_lit\_|\mathcal{A}| \end{cases}$$

Finally, to restrict solutions to decompositions we assert the existence of correct compositional inferences. Here, firstly a compositional inference x (denoted by ci_x) is discarded if it covers a possible entry in the composition table.

$$\begin{array}{l}\forall ci\_x = \langle L_{i_1}, L_{i_2}, L_{i_3}\rangle \text{ and } \forall j_1, j_2, j_3 \text{ s.t. } B_{j_3} \in B_{j_1} \circ B_{j_2}, \\ \qquad \{ \; \neg L\_i_1\_j_1 \vee \neg L\_i_2\_j_2 \vee \neg L\_i_3\_j_3 \vee \neg ci\_x\end{array}$$

where L_i is either A_i or $\neg A_i$.

Secondly, it is necessary to check that there exists at least one compositional inference applicable for every forbidden entry of the composition table. The next rule says that a compositional inference is not applicable if it does not exist. The last ones state that a compositional inference is not applicable on an entry, if it does not cover it.

$$\begin{array}{l}\forall j_1, j_2, j_3 \text{ s.t. } B_{j_3} \notin B_{j_1} \circ B_{j_2}, \\ \forall ci\_x = \langle T_1.L_{i_1}, T_2.L_{i_2}, T_3.L_{i_3}\rangle, \\ \qquad \left\{ \begin{array}{r} ci\_x \vee \neg ci\_x\_ao\_j_1\_j_2\_j_3 \\ L\_i_1\_j_1 \vee \neg ci\_x\_ao\_j_1\_j_2\_j_3 \\ L\_i_2\_j_2 \vee \neg ci\_x\_ao\_j_1\_j_2\_j_3 \\ L\_i_3\_j_3 \vee \neg ci\_x\_ao\_j_1\_j_2\_j_3 \end{array} \right.\end{array}$$

Finally, at least one rule should be applicable on every forbidden entry of the composition table:

$$\begin{array}{l}\forall j_1, j_2, j_3 \text{ s.t. } B_{j_3} \notin B_{j_1} \circ B_{j_2}, \\ \qquad \{ \; ci\_1\_ao\_j_1\_j_2\_j_3 \vee \cdots \vee ci\_|\mathcal{N}(C)|\_ao\_j_1\_j_2\_j_3\end{array}$$

Nevertheless, this program only computes one solution with k literals while not minimizing the number of compositional inferences. To overcome this, we implemented a counting method as introduced in [22] for the number of (true) compositional inferences ci_x and a number comparison with a given fixed bound n. We then launched the SAT program several times, first lowering the bound k to find the smallest decomposition in terms of literals and then lowering the allowed number of compositional inferences n until the lowest bound for this was reached as well. Thus the SAT program can be used to find optimal decompositions that minimize both k and n.

2) Time necessary to prove the existence of a solution: We compared the different computational approaches both in terms of running times and memory usage. The tests were run on an Intel Quad-Core Xeon 2.66 GHz with 4 GB of memory for each core and 4 MB of secondary cache. The Brute Force algorithm has been implemented in C++. The SAT solver used is Minisat Version 1:2.2.1-2 [9]. The ASP solver used is clingo 3.0.3 [11]. The ASP program and the SAT program are based on the idea described in Section III-B1.

Figure 2 gives the running times necessary to prove the existence of a solution given a fixed number of literals for the decomposition (here irrespective of the number of compositional inferences).

Method	3	4	5	6	7	8
Brute Force	12.9	111.5	1539.3	2611.5	4291.8	8.2
SAT	0.9	3.8	24.1	24.0	35.6	50.4
ASP	0.4	2.2	16.2	58.4	×	×

Fig. 2. Running times (sec) for the RCC8 calculus w.r.t the number of literals

Calculus	$\lvert\mathcal{B}\rvert$	$\lvert\mathcal{A}\rvert$	$\lvert\mathcal{N}\rvert$
Point Algebra	3	2	6
Point Branching	4	3	9
RCC5	5	3	12
RCC8	8	6	36
OCC	8	6	20
Cardinal Direction	9	4	12
Allen	13	8	48

Fig. 3. Size of the optimal decompositions for several qualitative calculi

$$\begin{array}{rcccc}
\neg P(a,b) & \wedge & Pi(b,c) & \rightarrow & \neg P(a,c) \\
\neg Pi(a,b) & \wedge & P(b,c) & \rightarrow & \neg Pi(a,c) \\
Pi(a,b) & \wedge & \neg P(b,c) & \rightarrow & \neg P(a,c) \\
P(a,b) & \wedge & \neg Pi(b,c) & \rightarrow & \neg Pi(a,c) \\
Pi(a,b) & \wedge & Pi(b,c) & \rightarrow & Pi(a,c) \\
P(a,b) & \wedge & P(b,c) & \rightarrow & P(a,c) \\
C(a,b) & \wedge & Pi(b,c) & \rightarrow & C(a,c) \\
\neg C(a,b) & \wedge & P(b,c) & \rightarrow & \neg C(a,c) \\
C(a,b) & \wedge & \neg C(b,c) & \rightarrow & \neg Pi(a,c) \\
Pi(a,b) & \wedge & \neg C(b,c) & \rightarrow & \neg C(a,c) \\
P(a,b) & \wedge & C(b,c) & \rightarrow & C(a,c) \\
\neg C(a,b) & \wedge & C(b,c) & \rightarrow & \neg P(a,c)
\end{array}$$

Fig. 4. Set of inferences for RCC5 decomposition

Even though the Brute Force algorithm is using far less memory than the others, it has running times which makes it unusable in practice. On the other end, the ASP programs obtained good running times but its memory usage makes it fail for the computation of size 7 and 8. Finally, only the SAT program offers both a good running time and a memory usage that allows computations of larger calculi, e.g., Allen's Interval Calculus. It nevertheless took more than a week to compute the optimal Allen decomposition.

3) Results obtained: The optimal decomposition (both in terms of literals and compositional inferences) obtained are given in the Figure 3. Due to lack of space, all decompositions cannot be detailed or discussed here.

Most of the results can be explained thanks to formalisms already known in the literature. For RCC5, the decomposition is identical to the translation into predicate logic presented in [19] and summarized in the background section with the inferences presented in Figure 4.

For the RCC8 calculus, the representation is identical to the description in terms of P, Pi, NTP, $NTPi$, DC and DR as given in [3]. 36 compositional inferences are necessary for covering the composition table.

The decomposition of the Cardinal Direction calculus is the same as the two-dimensional decomposition of the Point Algebra. The Allen minimal decomposition is the one using the decomposition of Point Algebra on start and endpoints given in [16] and represented in Figure 5.

These decompositions confirm the correctness of our results.

IV. Implementation and Evaluation

To evaluate the benefit of decompositions over composition tables, we experimented with encodings of qualitative satisfiability problems in SAT. For this, we here propose a way of

relation	A_1	A_2	A_3	A_4	A_5	A_6	A_7	A_8
meaning	$X^- < Y^-$	$X^- < Y^+$	$X^+ < Y^-$	$X^+ < Y^+$	$X^- > Y^-$	$X^- > Y^+$	$X^+ > Y^-$	$X^+ > Y^+$
d	$\times$	o	$\times$	o	o	$\times$	o	$\times$
di	o	o	$\times$	$\times$	$\times$	$\times$	o	o
o	o	o	$\times$	o	$\times$	$\times$	o	$\times$
oi	$\times$	o	$\times$	$\times$	o	$\times$	o	o
m	o	o	$\times$	o	$\times$	$\times$	$\times$	$\times$
mi	$\times$	$\times$	$\times$	$\times$	o	$\times$	o	o
f	$\times$	o	$\times$	$\times$	o	$\times$	o	$\times$
fi	o	o	$\times$	$\times$	$\times$	$\times$	o	$\times$
s	$\times$	o	$\times$	o	$\times$	$\times$	o	$\times$
si	$\times$	o	$\times$	$\times$	$\times$	$\times$	o	o
$<$	o	o	o	o	$\times$	$\times$	$\times$	$\times$
$>$	$\times$	$\times$	$\times$	$\times$	o	o	o	o
$=$	$\times$	o	$\times$	$\times$	$\times$	$\times$	o	$\times$

Fig. 5. Representation of the minimal decomposition for Allen's Interval Algebra

implementing SAT encodings thanks to decompositions. We will compare the performance achieved by this encodings with a state-of-the-art encoding given in [17]. These encodings are described in the next section.

We also implemented a prototype solver based on a DPLL procedure in order to evaluate how good such a solver would be. Results of the prototype are comparable with those of the GQR solver. Both system description and compared running times are given next.

A. Propositional SAT Encoding

We use two different but related encodings of qualitative CSP instances into propositional SAT instances in conjunctive normal form. The first encoding based on [17], grounds the composition table and thus can be applied to any qualitative calculus. The second encoding applies our decomposition technique by only grounding the composition inferences of a decomposition instead of the entire composition table. We briefly describe both procedures in the following and assume that the input is a normalized qualitative constraint network with l variables and constraints $\{R_{ij} \subseteq 2^{\mathcal{B}} \mid 1 \leq i < j \leq l\}$.

In the first encoding, each constraint $R_{ij} = \{B_1, \ldots, B_m\}$ is translated into a clause $B_{1_{ij}} \vee \ldots \vee B_{m_{ij}}$. We further add clauses $\bigwedge_{B,B' \in R_{ij}, B \neq B'} (\neg B_{ij} \vee \neg B'_{ij})$ to ensure that only one base relation holds between variables i and j. To ground the composition table, for every triple $1 \leq i < j < k \leq l$ and for all $B_{ij} \in R_{ij}$, $B_{jk} \in R_{jk}$ we introduce clauses $(B_{ij} \wedge B_{jk}) \rightarrow (B'_{1_{ik}} \vee \cdots \vee B'_{m_{ik}})$ where $\{B'_{1_{ik}}, \ldots, B'_{m_{ik}}\} = R_{ik} \cap (R_{ij} \circ R_{jk})$. We refer to this as the *support* encoding. For details see [17], [24].

The second encoding utilizes forbidden clauses. This is similar to another encoding approach by Pham et al. [17], which is based on composition, while we consider forbidden clauses generated from the calculus decomposition (which also allows for negated predicates). Given a decomposition into predicates $\mathcal{A} = \{A_1, \ldots, A_n\}$, for each pair $i, j, i < j$ we have a forbidden clause $\neg(L^1_{ij} \wedge \ldots \wedge L^n_{ij})$, if $\forall B \in R_{ij} : L^1_{ij} \wedge \ldots \wedge L^n_{ij} \neq T_B$. This encodes the input by ruling out any assignment of propositional atoms that does not correspond to an input relation. Next, we extensionally represent compositional inferences by instantiating them for triples of entities. For every triple $1 \leq i < j < k \leq l$ and every compositional inference in N, we have $\neg(L^1_{ik} \wedge \ldots \wedge L^n_{ik} \wedge L^1_{ij} \ldots \wedge L^n_{ij} \wedge L^1_{jk} \ldots \wedge L^n_{jk})$. We refer to this as the *decomposition* encoding.

We note here that both encodings suffer from a cubic blow-up due to the extensional representation of composition (or compositional inference), and thus the size of instances becomes a crucial issue as described in [24]. Nevertheless, the decomposition-based encoding results in much smaller instance. For example, if we consider RCC8 instances with 200 variables and average degree 8.5, the classical encoding of the problem has around 155.000 boolean variables and 81 million clauses while the new encoding only has 119.000 variables and 48 million clauses.

B. Prototype Implementation

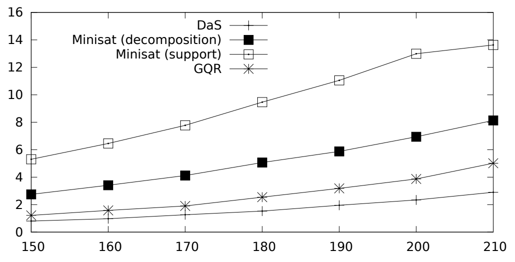
The main algorithm in our implementation is a classic DPLL procedure using the weighted domain over degree heuristic [5], in which compositional inferences are used as a propagation procedure. It has been written in C++ and for now does not include tractable subclasses. From now on, we will refer to our implementation as DaS (for *Decompose-and-Solve*).

C. Experimental study

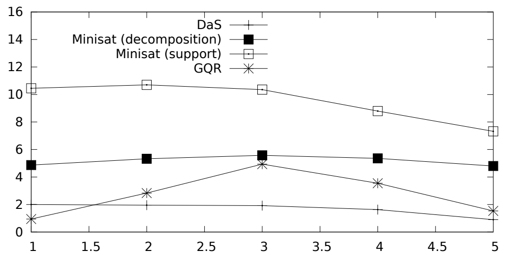
We now present the results of an experiment comparing SAT encodings and also the DaS solver and GQR. Namely, we compare DaS to the qualitative CSP solver GQR [10], version 1418, The SAT solver used is Minisat [9], version 2.2.0, applied on the presented propositional SAT encodings.

The tests were run on an Intel Quad-Core Xeon 2.66 GHz with 4 GB of memory for each core and 4 MB of secondary cache. The test protocol is as follows: since we are interested in two criteria (the number of entities in the problem and the average degree of the graph), we created for every pair of size and degree 200 random normalized qualitative constraint networks according to the A-model used in [24]. That is, a random graph with given size and degree is generated, and random relations are assigned to its edges. The tests have been conducted for RCC5 and RCC8.

Concerning the SAT encodings, the time necessary for the translation into the SAT format are not included. The objective here is to evaluate the overall value of the approach and as DaS does not yet incorporate tractable subclasses, GQR is

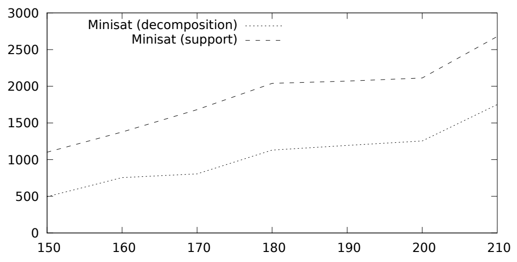


(a) Runtimes for RCC5 w.r.t. size.

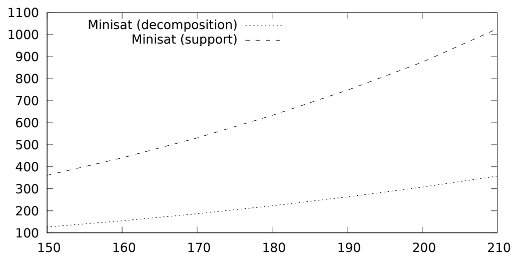


(b) Runtimes for RCC5 w.r.t. degree.

Fig. 6. Runtimes (seconds) for network sizes (150-210) and degree (1-5)



(a) Memory usage of Minisat for RCC5.



(b) Size of CNF for RCC5.

Fig. 7. Memory usage (Mb) for RCC5 SAT encodings of networks with sizes (150-210) and degree (1-5)

configured not to use tractable subclasses. Further, no time limit has been set.

We can see in Figure 6(a) and Figure 6(b) that the decomposition-based SAT encoding is clearly superior to the support-based SAT encoding. Further, we can see that DaS has an overall better performance than GQR.

For the RCC5 calculus, we obtain a very impressive reduction of the size of the CNF encodings shown in Figure 7(b) and Figure 7(a). The new encodings is 34% of the support SAT encodings on average. The memory usage for Minisat is also improved being more than halved for some instances. This gain in memory usage may be the main explanation for the gain in running times, this is confirmed by the lesser gain for RCC8.

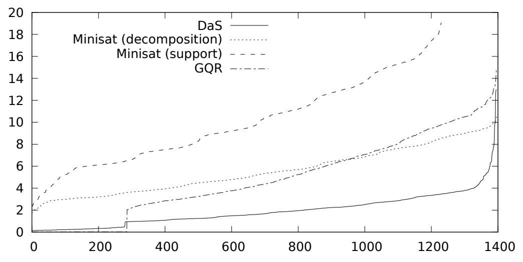
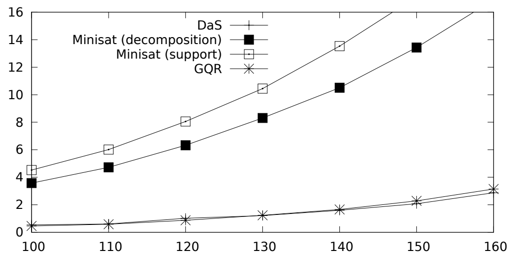
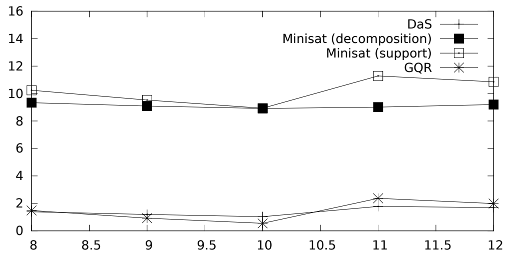


Fig. 8. Number of solved RCC5 instances with degree 3 (1400 in total) over time (seconds)

For DaS and GQR the running times are highly dependent on whether the instance is satisfiable (SAT) or unsatisfiable (UNSAT). We illustrate this for problem instances with a degree of 3, as this is the hardest spot. Figure 8 shows the number of solved instances over time. GQR performs better on UNSAT instances, which are the data points close to a runtime of 0 seconds. This behaviour can also be seen for DaS, but it is less distinctive here. For the rest of the instances (which are SAT), DaS is clearly faster. This difference does not exist between the two SAT encodings.



(a) Runtimes for RCC8 w.r.t. size.



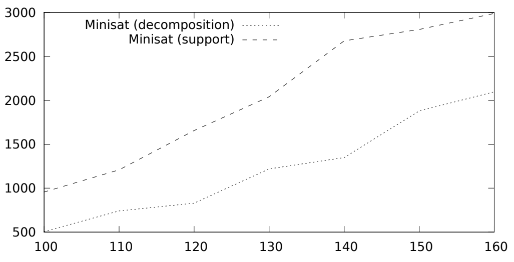
(b) Runtimes for RCC8 w.r.t. degree.

Fig. 9. Runtimes (seconds) for network sizes (100-160) and degree (8-12)

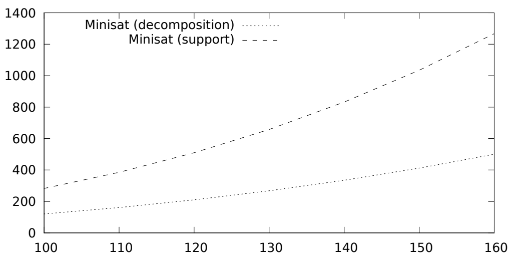
The results for RCC8 are shown in Figure 9(a) and Figure 9(b). DaS and GQR reach comparable running times for RCC8. The decomposition-based SAT encoding still performs slightly better than the support-based SAT encoding even if the difference is less clear than in the case of RCC5.

We can see in Figure 10(b) and Figure 10(a) that the gain in memory usage is still very good (40% of the size of the support SAT encodings on average) but is slightly inferior to the RCC5 calculus. This may be one explanation of the differences in running times.

Overall, even if the improvement depends on the considered calculus, the decomposition seems to allow faster reasoning



(a) Memory usage of Minisat for RCC8.



(b) Size of CNF for RCC8.

Fig. 10. Memory usage (Mb) for RCC8 SAT encodings of networks with sizes (100-160) and degree (8-12)

compared to utilizing composition tables.

V. Conclusion and perspectives

In this paper, we proposed a framework for automatically decomposing Qualitative Spatial and Temporal Reasoning formalisms into representations with equivalent inference rules. Such decompositions can be automatically computed from the composition table of the calculus thanks to a SAT program we presented. We discussed the complexity of the problem at hand and provided experimental results to underline its hardness. We then applied it to several QSTR calculi and obtained new minimal definitions for standard calculi such as RCC5, RCC8 and Allen's Interval Calculus. Moreover, we showed that the proposed decompositions considerably improve propositional logic encodings in both instance size and runtime of the Minisat solver. Further, we also presented a prototype solver based on the decomposition technique in order to evaluate the efficiency gains of the new representation. Our results show that this solver compares favorably with respect to the state-of-the-art GQR solver.

Future work will concern approximation techniques for decompositions in order to deal with larger formalisms such as RCC23 or OPRA which currently are beyond the reach of our programs. Another research question is how tractable subclasses, such as Horn theories [16] and classes built on k-consistency [4], can be integrated into our decomposition approach.

Acknowledgment

We thank the anonymous reviewers for their constructive comments. This work was supported by DFG (Transregional Collaborative Research Center *SFB/TR 8 Spatial Cognition*, project R4-[LogoSpace]).

References

[1] James F. Allen. Maintaining knowledge about temporal intervals. *Communications of the ACM*, 26(11):832–843, 1983.

[2] Brandon Bennett. Spatial reasoning with propositional logics. In Jon Doyle, Erik Sandewall, and Pietro Torasso, editors, *KR*, pages 51–62. Morgan Kaufmann, 1994.

[3] Brandon Bennett. *Logical Representations for Automated Reasoning about Spatial Relationships*. PhD thesis, School of Computing, The University of Leeds, 1997.

[4] Manuel Bodirsky and Hubie Chen. Qualitative temporal and spatial reasoning revisited. *Journal of Logic and Computation*, 19(6):1359–1383, 2009.

[5] Frédéric Boussemart, Fred Hemery, Christophe Lecoutre, and Lakhdar Sais. Boosting systematic search by weighting constraints. In Ramon López de Mántaras and Lorenza Saitta, editors, *ECAI*, pages 146–150. IOS Press, 2004.

[6] Sebastian Brand. Relation variables in qualitative spatial reasoning. In Susanne Biundo, Thom W. Frühwirth, and Günther Palm, editors, *KI*, volume 3238 of *Lecture Notes in Computer Science*, pages 337–350. Springer, 2004.

[7] Anthony G. Cohn and Shyamanta M. Hazarika. Qualitative spatial representation and reasoning: An overview. *Fundamenta Informaticae*, 46(1-2):1–29, 2001.

[8] Ivo Düntsch, Hui Wang, and Stephen McCloskey. Relation algebras in qualitative spatial reasoning. *Fundamenta Informaticae*, 39:229–248, 1999.

[9] Niklas Eén and Niklas Sörensson. An extensible SAT-solver. In *Proc. of SAT 2003*, volume 2919 of *Lecture Notes in Computer Science*, pages 502–518. Springer, 2003.

[10] Zeno Gantner, Matthias Westphal, and Stefan Wölfl. GQR - A fast reasoner for binary qualitative constraint calculi. Technical Report WS-08-11, AAAI Workshop on Spatial and Temporal Reasoning, 2008.

[11] Martin Gebser, Benjamin Kaufmann, André Neumann, and Torsten Schaub. clasp: A conflict-driven answer set solver. In Chitta Baral, Gerhard Brewka, and John S. Schlipf, editors, *LPNMR*, volume 4483 of *Lecture Notes in Computer Science*. Springer, 2007.

[12] Robin Hirsch. A finite relation algebra with undecidable network satisfaction problem. *Bulletin of the IGPL*, 7:547–554, 1999.

[13] Jason Jingshi Li and Jochen Renz. In defense of large qualitative calculi. In Maria Fox and David Poole, editors, *AAAI*. AAAI Press, 2010.

[14] Sanjiang Li and Mingsheng Ying. Extensionality of the RCC8 composition table. *Fundamenta Informaticae*, 55(3-4):363–385, 2003.

[15] Gérard Ligozat and Jochen Renz. What is a qualitative calculus? A general framework. In Chengqi Zhang, Hans W. Guesgen, and Wai-Kiang Yeap, editors, *PRICAI*, volume 3157 of *Lecture Notes in Computer Science*, pages 53–64. Springer, 2004.

[16] Bernhard Nebel and Hans-Jürgen Bürckert. Reasoning about temporal relations: A maximal tractable subclass of Allen's Interval Algebra. *Journal of the ACM*, 42(1):43–66, 1995.

[17] Duc Nghia Pham, John Thornton, and Abdul Sattar. Towards an efficient SAT encoding for temporal reasoning. In Frédéric Benhamou, editor, *CP*, volume 4204 of *Lecture Notes in Computer Science*, pages 421–436. Springer, 2006.

[18] D.A. Randell and A. G. Cohn. Modelling topological and metrical properties in physical processes. In Raymond Reiter Ronald J. Brachman, Hector J. Levesque, editor, *KR*, pages 55–66. Morgan Kaufmann, 1989.

[19] David A. Randell, Zhan Cui, and Anthony G. Cohn. A spatial logic based on regions and connection. In William R. Swartout Bernhard Nebel, Charles Rich, editor, *KR*, pages 165–176. Morgan Kaufmann, 1992.

[20] Raymond Reiter. A theory of diagnosis from first principles. *Artificial Intelligence*, 32(1):57–95, 1987.

[21] Jochen Renz and Bernhard Nebel. Qualitative spatial reasoning using constraint calculi. In *Handbook of Spatial Logics*, pages 161–215. Springer Verlag, Berlin, 2007.

[22] Carsten Sinz. Towards an optimal cnf encoding of boolean cardinality constraints. In Peter van Beek, editor, *CP 2005 (LNCS 3709)*, volume 3709 of *Lecture Notes in Computer Science*, pages 827–831. Springer Berlin / Heidelberg, 2005.

[23] Marc Vilain, Henry Kautz, and Peter van Beek. Constraint propagation algorithms for temporal reasoning. In *Readings in Qualitative Reasoning about Physical Systems*, pages 377–382. Morgan Kaufmann, 1986.

[24] Matthias Westphal and Stefan Wölfl. Qualitative CSP, finite CSP, and SAT: Comparing methods for qualitative constraint-based reasoning. In Craig Boutilier, editor, *IJCAI*, pages 628–633. AAAI Press, 2009.