In Defense of Large Qualitative Calculi

Jason Jingshi Li and **Jochen Renz**
Artificial Intelligence Group, School of Computer Science
The Australian National University, Canberra, ACT 0200, Australia
& Canberra Research Laboratory, National ICT Australia

Abstract

The next challenge in qualitative spatial and temporal reasoning is to develop calculi that deal with different aspects of space and time. One approach to achieve this is to combine existing calculi that cover the different aspects. This, however, can lead to calculi that have a very large number of relations and it is a matter of ongoing discussions within the research community whether such large calculi are too large to be useful. In this paper we develop a procedure for reasoning about some of the largest known calculi, the Rectangle Algebra and the Block Algebra with about 10^{661} relations. We demonstrate that reasoning over these calculi is possible and can be done efficiently in many cases. This is a clear indication that one of the main goals of the field can be achieved: highly expressive spatial and temporal representations that support efficient reasoning.

1. Introduction

Knowledge about space and time is an important part of every intelligent system. While space and time can be represented using a coordinate system, the most common approach within the AI community is a qualitative representation. A qualitative representation aims at representing spatial or temporal information in a symbolic way that is well suited for human users to recognize, to memorize, and to communicate. This is done by representing information about spatial or temporal entities using a finite and usually small number of possible relationships that adequately represent a given spatial or temporal scenario. Well known examples are the Region Connection Calculus RCC-8 (Randell, Cui, and Cohn 1992) for representing topological relationships between spatially extended entities, and the Interval Algebra (IA) (Allen 1983) for representing relationships between convex intervals on a directed line. RCC-8 distinguishes eight pairwise disjoint and mutually exhaustive relations (also called *base relations*) between extended regions, i.e., between any two regions exactly one base relation holds. IA distinguishes thirteen base relations.

In both cases, the number of base relations is small and can be easily remembered. Cognitive studies have shown that both systems of relations are naturally used by humans (Renz, Rauh, and Knauff 2000). One of the major advantages of a qualitative representation is that it allows to easily represent uncertainty and indefinite information by specifying a union of possible base relations. This gives us a total number of $2^{|\mathcal{B}|}$ relations that can be distinguished for a given set of base relations $\mathcal{B}$. For RCC-8 we get $2^8 = 256$ relations and for IA $2^{13} = 8192$ relations. While such numbers of different relations are clearly too large for humans to remember and to use effectively, the underlying set of base relations is small. Therefore, each of the 256 and 8192 relations can be easily derived, explained and understood. What is more, reasoning over information expressed with these relations can be done efficiently in most cases (Nebel 1997), despite it being an NP-complete problem.

An obvious question now is how large can a qualitative spatial or temporal calculus get without losing the advantages of a qualitative representation, i.e, how many relations can we deal with? We argue that this depends on two factors:

1. **Cognitive adequacy of the base relations**, in particular, can humans easily remember and intuitively use all of the base relations?
2. **Effective and efficient reasoning over the full calculus**, i.e., can we (efficiently) solve reasoning problems over a given calculus up to a reasonable size?

The first point, cognitive adequacy, does not only depend on the number of base relations, but also on how they are structured. For example, the Rectangle Algebra (RA) (Balbiani, Condotta, and del Cerro 1998), represents relations between rectangles on a plane by using IA relations for projections of the rectangles on the x-axis and the y-axis. RA has 169 base relations, 13 for each axis. Clearly 169 "different" relations would be way too many to remember and to effectively distinguish. But since they are based on a combination of only 13 IA base relations, all 169 RA base relations can be distinguished and enumerated relatively easily.

The second point, effective and efficient reasoning, depends on many factors. Reasoning is NP-complete for almost all calculi that have been studied in the literature. Therefore, "efficient reasoning" in this context means the possibility of solving most problem instances up to a reasonably large size very fast, "effective reasoning" means to be able to do formal reasoning at all. The reason we can get efficient solutions for these NP-hard reasoning problems is the use of tractable subsets of a calculus. These are subsets of the set of all relations for which reasoning is tractable and which can be utilized for finding efficient solutions of otherwise hard problem instances (Nebel 1997). So one factor is the ability to identify large tractable subsets of a calculus. Recent advances in the field led to the possibility of automatically identifying tractable subsets of a calculus (Renz 2007;

Renz and Li 2008). This requires computationally expensive procedures whose runtime depends on the size of the calculus. Rough estimates led to an upper bound of approximately 13-15 base relations, which seems too low for the large calculi we discuss in this paper. This leaves the possibility of manually identifying large tractable subsets. Other factors are, e.g., the ability to compute and to store the composition table of a calculus (which is required for reasoning) and the ability to store and to retrieve tractable subsets of a calculus. These become serious issues when we have 169 base relations and 2^{169} (approx. 10^{50}) relations in total, as we have for the RA, and could make formal reasoning on todays computers impossible.

The question of how large can a qualitative calculus reasonably be becomes more and more important with the current direction the field is heading. It is one of the main challenges in the field of qualitative spatial and temporal reasoning to combine different calculi in order to represent more expressive spatial and temporal information. Different calculi are used to represent different aspects of space or time or information in different dimensions. The above mentioned calculi, for example, only deal with topology (RCC-8) or topology plus direction (IA). If we want to represent information about different aspects, we have to develop calculi that deal with all the required aspects simultaneously.

One approach of combining calculi is to keep the different calculi separately and to propagate information between them in order to deal with dependencies between the different aspects. A well-known method that combines calculi in that way is Bipath-consistency (Gerevini and Renz 2002). An alternative approach is to develop a new calculus whose base relations are formed as the cross-product of the original base relations. This is analogous to how the RA is derived as a cross-product of the IA with itself. As for the Rectangle Algebra, this approach, leads to a massive explosion in the number of relations, e.g., combining two calculi each with 10 base relations and 2^{10} relations in total, leads to a new calculus with 100 base relations and 2^{100} relations in total. It is commonly believed that reasoning over such gigantic calculi is not feasible in practice and that, therefore, methods such as Bipath-consistency are the preferred approach for combining calculi.

In this paper we analyze if reasoning with very large calculi is possible and if it can be done efficiently. Inspired by recent results on how spatial and temporal constraint networks can be solved very efficiently by applying divide-and-conquer methods, we present a procedure that is able to effectively reason about huge combined calculi—provided that the individual calculi satisfy a previously introduced property, the atomic network amalgamation property (Li, Huang, and Renz 2009). Our procedure first divides constraint networks into an equivalent set of smaller networks, then splits the relations of the combined calculus into relations of the original calculi by using a technique similar to bipath-consistency, and finally transforms the outcome into a propositional formula which can then be solved using an off-the-shelf SAT solver.

As an example, we use RA and its extension, the Block Algebra (BA) (Balbiani, Condotta, and del Cerro 2002) which is a combination of IA over three dimensions and consists of a massive $2^{13\times13\times13}$ (about 10^{661}) relations! In an empirical analysis we show that we can efficiently solve relatively large instances. Our procedure is superior to any existing technique and successfully demonstrates that huge calculi, that inevitably occur when combining other calculi, can be used for practical spatial and temporal reasoning.

The paper is structured as follows. In Section 2 we introduce the necessary background on qualitative calculi and different methods and techniques we use in this paper. Further details and references can be found in (Renz and Nebel 2007). In Section 3 we present a novel procedure for reasoning over large calculi such as the RA or the Block Algebra and prove its correctness. In Section 4 we present empirical evidence that our procedure is successful in efficiently solving large RA and BA instances and compare its performance to existing methods. Section 5 discusses our results.

2. Background

A qualitative spatial or temporal calculus is based on a given and usually infinite domain $\mathcal{D}$ of spatial or temporal entities and a set of pairwise disjoint and jointly exhaustive base relations $\mathcal{B}$ which is a partition of $\mathcal{D} \times \mathcal{D}$. The base relations usually distinguish a certain spatial or temporal aspect such as direction or topology on a certain level of granularity. In order to express indefinite information, we allow all relations $2^{\mathcal{B}}$ from the powerset of the set of base relations.

The well-known IA, for example, distinguishes thirteen base relations between two intervals x and y: x before(b) y, x after(bi) y, x meets(m) y, x met-by(mi) y, x overlaps(o) y, x overlapped-by(oi) y, x starts(s) y, x started-by(si) y, x finishes(f) y, x finished-by(fi) y, x during(d) y, x contains(di) y, and x equal(eq) y. The relations marked with an "i" in their abbreviation are the *converse* relations of the corresponding relations. Intervals can also be represented by using their start and end point and the relations between these points which can be one of $<, >, =$. This is called the *Point Algebra* (PA), which can only express a small subset of all possible IA relations that includes all IA base relations. Other than converse of relations, we can also use *union* ($\cup$), *intersection* ($\cap$), and *composition* ($\circ$) of relations. Composition is particularly important for reasoning as it allows us to derive previously unknown relations, for example, if xRy and ySz are known, the relation between x and z is determined by $xR \circ Sz$. If previous information xTz is given, then composition leads to $xT \cap (R \circ S)z$. This is known as the *path-consistency* operation. Composition has to be computed for every new domain and every new set of base relations and is often very difficult if not impossible to obtain when the domain is infinite. The most common reasoning problem is the *consistency problem*: Given a set Θ of constraints xRy with relations $R \in 2^{\mathcal{B}}$ and variables $x, y \in \mathcal{V}$ over a domain $\mathcal{D}$, is Θ consistent, i.e., is there an assignment of all variables of $\mathcal{V}$ with values from $\mathcal{D}$ such that all constraints in Θ are satisfied? Throughout the paper we sometimes call a set of constraints a CSP, a CSP network, or just a network. If a network contains only base relations, it is called an *atomic network* or a *scenario*.

Consistency can be approximated by using the path-consistency algorithm which applies the path-consistency operation to all triples of variables of $\mathcal{V}$ until a fixed point is reached, in which case the resulting set is called *path-consistent* or an inconsistency is derived, in which case Θ is inconsistent. Path-consistency is also called 3-consistency.

In general, a set of constraints is *k-consistent* (for $k \leq |\mathcal{V}|$) if for any consistent assignment of $k-1$ variables of Θ, there is a consistent assignment of any k-th variable such that all k variables together are assigned consistently. A set of constraints is *strong n-consistent*, if it is k-consistent for any $k \leq n = |\mathcal{V}|$. For most cases, the consistency problem of a calculus is NP-hard if all $2^{\mathcal{B}}$ relations of a calculus are permitted. If the consistency problem is tractable for a subset $\mathcal{S}$ of $2^{\mathcal{B}}$ (called a *tractable subset*) and $\mathcal{B} \subseteq \mathcal{S}$, then the consistency problem can be solved by splitting relations outside $\mathcal{S}$ into subrelations in $\mathcal{S}$ and by backtracking over these tractable subrelations. If the tractable subsets are large, then this method often leads to very efficient solutions.

Combinations of Calculi

In recent years there has been an increased interest in combining different qualitative calculi in order to obtain calculi that can deal with more than one aspect of space or time or with entities in multiple dimensions. Different approaches to combining calculi have been proposed in the literature. They can be divided into *orthogonal combinations* and *non-orthogonal combinations* and also in what we call *parallel combination* and *cross-product combination*. In an orthogonal combination, there are no interdependencies between the two calculi, i.e., for every base relation b_i of the first calculus, we can pick any base relation b_j of the second calculus and find an instantiation for x and y such that both xb_iy and xb_jy are satisfied. For a non-orthogonal combination this property is not satisfied and there are interdependencies. An example for an orthogonal combination is the RA which is a combination of the IA over two dimensions. It is clear that between two rectangles, the interval relation on the x-axis is independent of the interval relation on the y-axis and that all combinations are possible.

A parallel combination is where we keep different sets of constraints for the different calculi in parallel and propagate information between the different sets of constraints in order to preserve interdependencies. An example for this is the bipath-consistency method introduced by Gerevini and Renz (2002) which uses an interdependency table between the different sets of relations and propagates new restrictions to the other set whenever a constraint of one set is revised. A cross-product combination is one that introduces a new calculus by forming the cross product of the relations of each of the calculi. For example, for every relation b_i of the first calculus and every relation b_j of the second calculus, we generate a new relation $b_ib_j \subseteq \mathcal{D} \times \mathcal{D}$ which is the intersection of b_i and b_j. The RA is an example for a cross-product combination as there are 13x13 new base relations, all possible combinations are covered.

(Wölfl and Westphal 2009) analysed the different approaches to combining qualitative calculi and found that a cross-product combination often provides a tighter and more expressive combination that can sometimes even lead to more efficient reasoning. The main problem with the cross-product combination is that it leads to a very large number of base relations, which may be too large to handle effectively.

Propositional SAT Encoding

Previously, the fastest reasoning methods for qualitative calculi have been based on constraint satisfaction with backtracking over large tractable subsets. Recently it was found that transforming a set of constraints into a formula in propositional logic and solving this formula using off-the-shelf SAT solvers can also lead to very efficient solutions. One advantage of the SAT-based method is that it doesn't require tractable subsets and therefore may be very useful for large qualitative calculi where tractable subsets are hard to come by but also hard to store and retrieve.

We describe here the point-based propositional encoding of the IA as developed by (Pham, Thornton, and Sattar 2008) that transforms a set of constraints Θ over the IA into a set of clauses, i.e., into a propositional satisfiability problem (SAT) in conjunctive normal form. The point-based encoding describes the relations M between any two intervals l and m in terms of the PA relations between the four endpoints of the interval l_-, l_+, m_-, m_+. Let M_{lm} denote the interval relation between interval l and m, D_{ij} denote the PA relation between points i and j, x_{ij}^w denote the propositional variable which is true *iff* the PA relation w is true between points i and j. We also say $\mu(r)$ is the PA representation of the IA relation r. Given the above definitions, four sets of clauses are introduced to describe the problem:

1. ALO: $\bigvee_{v \in D_{ij}} x_{ij}^v$
2. AMO: $\bigwedge_{u,v \in D_{ij}} \neg x_{ij}^u \vee \neg x_{ij}^v$ for $u \neq v$
3. SUP: $\bigwedge_{u \in D_{ik}, v \in D_{kj}} \neg x_{ik}^u \vee \neg x_{kj}^v \vee x_{ij}^{w_1} \vee \ldots \vee x_{ij}^{w_m}$
   where $\{w_1, \ldots, w_m\} = D_{ij} \cap u \circ v$
4. FOR: $\bigwedge_{r \notin M_{lm}} \neg x_{l-m-}^u \vee \neg x_{l-m+}^v \vee \neg x_{l+m-}^y \vee \neg x_{l+m+}^z$
   where $\mu(r) = (u, v, y, z)$

The first set of clauses (at-least-one) ensure there are at least one PA relation between any two endpoints, the second (at-most-one) ensures that there are at most one PA relation between any two endpoints. The two set of clauses made sure that the final solution is an atomic network. The third set of clauses (support) encode the composition constraint that enforces path-consistency. The forth (forbidden) guarantee that the PA network does not permit any *spurious relations*, i.e., relations between intervals l and m in the point-based encoding that are not part of the original CSP.

The Divide and Conquer Approach

While the SAT encoding of the IA provides a reasonable performance, it is not outstanding compared to the best constraint-based approaches such as the GQR solver (Westphal and Wölfl 2009). However, by utilising a new divide-and-conquer approach (Li, Huang, and Renz 2009), that divides a large constraint network into a number of smaller networks, and combine this with the SAT-based encoding, leads to a significant speed-up. It has been shown that this partitioning approach significantly reduces the size of the SAT encoding and the solving time for IA networks. We give more details on this method in the next section.

3. A Hybrid Procedure for RA and BA

We now propose a novel procedure for deciding consistency of RA and BA networks. The procedure utilizes the advantages of a cross-product combination as well as those

of a parallel combination of the IA with itself. Since RA and BA are formed by a cross-product combination, we can use a previously introduced divide-and-conquer method provided that RA and BA satisfy some required properties. This method partitions a network into an equivalent set of smaller networks. We will then split up the cross-product relations into different independent sets of relations for each dimension plus additional interdependency conditions as if we were doing a parallel combination. The resulting constraints are then encoded into a propositional SAT formula, while solution-finding can be delegated to state-of-the-art SAT solver Minisat (Eén and Sörensson 2003).

Divide and Conquer

We first partition the given CSP network by applying the divide-and-conquer approach as described in (Li, Huang, and Renz 2009). The technique partitions the CSP network into a number of smaller networks sharing overlapping parts. Li, Huang, and Renz proved that the large network is consistent if and only if all the smaller networks are consistent, provided that certain conditions are satisfied by the underlying calculus. In order to show that the divide-and-conquer approach is applicable to RA and BA, we must first show that these conditions are satisfied:

1. Path-consistency decides consistency for atomic networks: given a path-consistent network where all the labels are base relations, then there exists an instantiation of all variables such that all constraints are satisfied.
2. *Atomic Network Amalgamation Property (aNAP)*: given two atomic networks with an overlapping part, they can be amalgamated into a large network and no existing constraints are modified by applying path-consistency.

For the first condition, it follows from the work in (Balbiani, Condotta, and del Cerro 1998) that path-consistent atomic networks of RA and BA are strong-n-consistent. This result coupled with the theorem 1 in (Li, Huang, and Renz 2009) entails that BA and RA both possesses Atomic Network Amalgamation Property.

Point-Based Representation

After the network has been appropriately partitioned, we encode the smaller networks into a propositional satisfiability problem. We propose a point-based encoding for BA that is an extension of the point-based encoding for IA (Pham, Thornton, and Sattar 2008). For RA, we simply reduce the point-based constraint to one less dimension.

As any block can be uniquely represented by two points in 3D space, we refer to these representative points of a block p as $p1, p2$. Any BA base-relation r between two blocks p and q can be transformed into a collection of PA relation along the x, y and z axis between the representative points of p and q. We define this transformation as $\mu(r) = (v_{x\_p1q1}, v_{x\_p1q2}, v_{x\_p2q1}, v_{x\_p2q2}, v_{y\_p1q1}, v_{y\_p2q2}, v_{y\_p2q1}, v_{y\_p2q2}, z_{y\_p1q1}, v_{z\_p1q2}, v_{z\_p2q1}, v_{z\_p2q2})$, where for example, v_{x_p1q1} denotes the PA relation between points $p1$ and $q1$ along the x axis.

Definition 1. *Given a BA network Θ with n nodes and its corresponding PA networks P_x, P_y and P_z (each with 2n points, $P_{[x|y|z]\_1}, \ldots, P_{[x|y|z]\_2n}$), the corresponding point-based CSP of Θ is a triple (X,D,C) where*

- $X = \{X_{x\_ij}, X_{y\_ij}, X_{z\_ij} | i, j \in [1 \ldots 2n], i < j\}$. *These are the variables of the CSP, where for example, X_{x_ij} represents a PA relation between two points i and j on the x-axis.*
- $D = \{D_{x\_ij}, D_{y\_ij}, D_{z\_ij}, \}$ *where each D_{x_ij} is the set of domain values of X_{x_ij}, which is the set of point relations between P_{x_i} and P_{x_j} on the x-axis. Likewise with D_{y_ij} for the y-axis and D_{z_ij} for the z-axis.*
- C *is a set of the following constraints, corresponding to the PA constraints on the x, y and z axes and the forbidden constraints to rule out spurious BA relations permitted by PA constraints alone:*
  - $\bigwedge_{u \in D_{x\_ik}, v \in D_{x\_kj}} X_{x\_ik} = u \wedge X_{x\_kj} = v \Rightarrow X_{x\_ij} \in D_{x\_ij} \cap (u \circ v)$
  - $\bigwedge_{u \in D_{y\_ik}, v \in D_{y\_kj}} X_{y\_ik} = u \wedge X_{y\_kj} = v \Rightarrow X_{y\_ij} \in D_{y\_ij} \cap (u \circ v)$
  - $\bigwedge_{u \in D_{z\_ik}, v \in D_{z\_kj}} X_{z\_ik} = u \wedge X_{z\_kj} = v \Rightarrow X_{z\_ij} \in D_{z\_ij} \cap (u \circ v)$
  - $\bigwedge_{r \notin M_{pq}} \mu(r) \neq (X_{x\_p1q1}, X_{x\_p1q2}, X_{x\_p2q1}, X_{x\_p2q2}, X_{y\_p1q1}, X_{y\_p1q2}, X_{y\_p2q1}, X_{y\_p2q2} X_{z\_p1q1}, X_{z\_p1q2}, X_{z\_p2q1}, X_{z\_p2q2})$.

Theorem 1. *Let Θ be a BA network and Φ be the corresponding point-based CSP defined in Definition 1, then Θ is consistent iff Φ is consistent.*

Proof. ($\Rightarrow$) Let Θ' be a consistent scenario of Θ. As the labels of Θ' consist of only base relations, it corresponds to a point-based CSP Φ' that is a consistent scenario of Φ. ($\Leftarrow$) Let Φ' be an instantiation of Φ satisfying C. Let the inverse of $\mu(r)$ be $\mu^{-1}(X_{x\_p1q1}, X_{x\_p1q2}, X_{x\_p2q1}, X_{x\_p2q2}, X_{y\_p1q1}, X_{y\_p1q2}, X_{y\_p2q1}, X_{y\_p2q2}, X_{z\_p1q1}, X_{z\_p1q2}, X_{z\_p2q1}, X_{z\_p2q2})$, such that it maps to the PA atomic relations of four end points over three dimensions to exactly one BA relation. So we construct a BA network Θ' from Φ' by labeling each edge (l,m) with the corresponding BA atomic relation $\mu^{-1}(X_{x\_p1q1}, X_{x\_p1q2}, X_{x\_p2q1}, X_{x\_p2q2}, X_{y\_p1q1}, X_{y\_p1q2}, X_{y\_p2q1}, X_{y\_p2q2}, X_{z\_p1q1}, X_{z\_p1q2}, X_{z\_p2q1}, X_{z\_p2q2})$. Therefore, Θ' is a consistent scenario of Θ as Φ' satisfies all constraints of C □

Encoding to Propositional Satisfiability

To encode a BA CSP as a propositional satisfiability problem, we first encode the projection of the problem in IA on three different axis. The IA CSP on each of the axis is encoded into SAT using the previous point-based encoding scheme. In addition, to fully encode the set of constraints C in Definition 1, we need to add the following set of clauses for every relation M between blocks p and q:

- $\bigwedge_{r \notin M_{pq}} (\neg x^{u1}_{x\_p1q1} \vee \neg x^{v1}_{x\_p1q2} \vee \neg x^{y1}_{x\_p2q1} \vee \neg x^{z1}_{x\_p2q2} \neg x^{u2}_{y\_p1q1} \vee \neg x^{v2}_{y\_p1q2} \vee \neg x^{y2}_{y\_p2q1} \vee \neg x^{z2}_{y\_p2q2} \neg x^{u3}_{z\_p1q1} \vee \neg x^{v3}_{z\_p1q2} \vee \neg x^{y3}_{z\_p2q1} \vee \neg x^{z3}_{z\_p2q2})$

  where $\mu(r) = (u1, v1, y1, z1, u2, v2, y2, z2, u3, v3, y3, z3)$

This set of clauses are designed to rule out the possible BA relations that are described by the set of PA relations but not part of the original problem description. In practice they can be simplified into fewer and shorter clauses.

Example

We will proceed to introduce an example to our approach. As we are dealing with large calculus and the problem can be enormously complex, we will deal with extremely simple scenarios to illustrate our approach.

We begin with a simple BA CSP with 4 nodes. The constraints are described as follows:

1. $N_1(<,<,<) \vee (>,>,>)N_2$
2. $N_1(<,s,s)N_3$
3. $N_2(di,>,>)N_3$
4. $N_3(<,d,f) \vee (<,f,fi) \vee (o,d,di) \vee (>,d,fi)N_4$

First we apply the divide-and-conquer technique. It partitions the network into two smaller networks: (N_1, N_2, N_3) and $(N_3, N4)$. We do not have to encode the latter as it is trivially consistent and overlaps only one node to the former.

Now we proceed to encode the constraints 1 to 3. We first convert the constraints into the point-based constraints between two points that would uniquely identify the block. E.g. the constraint between two points in N_1 is:

- $(N_{1s} <_x N_{1e}) \wedge (N_{1s} <_y N_{1e}) \wedge (N_{1s} <_z N_{1e})$

where $(N_{1s} <_x N_{1e})$ denotes the "starting" point of N_1 is positioned less than the "ending" point of N_1 along the x-axis. Likewise the constraint for points in N_2 and N_3. To encode the relation between representative points of N_1 and N_2 on the x-axis, the following constraints are introduced:

- $(N_{1s} <_x \vee >_x N_{2s}) \wedge (N_{1s} <_x \vee >_x N_{2e})$
- $(N_{1e} <_x \vee >_x N_{2s}) \wedge (N_{1e} <_x \vee >_x N_{2e})$

We introduce similar constraints for the y and z axis. However, this is not sufficient to describe the BA relation $(<,<,<) \vee (>,>,>)$, as those constraints allow for spurious BA relations such as $(<,>,<)$. To rule out such spurious relations, we must also describe the interdependencies between the axis. The "forbidden" constraints between N_{1s} and N_{2s} can be simplified as the following:

- $\neg(N_{1s} <_x N_{2s} \wedge N_{1s} >_y N_{2s})$
- $\neg(N_{1s} <_x N_{2s} \wedge N_{1s} >_z N_{2s})$
- $\neg(N_{1s} <_y N_{2s} \wedge N_{1s} >_z N_{2s})$

Similar constraints are introduced for the pairs (N_1s, N_2e), (N_1e, N_2s) and (N_1e, N_2e). This completes the encoding for the first constraint in the original CSP. Constraint 2 and 3 are encoded in a similar way and that completes the transformation.

4. Empirical Evaluation

We now test the effectiveness of our procedure on randomly generated instances of the RA and the BA. For the RA, we can compare our results with GQR, the current state-of-the-art constraint solver, while the BA cannot be solved by any existing solver we are aware of. It should be noted that for these large calculi, the total number of relations is much larger than the number of relations that are used in the randomly generated instances. We used Intel Core2Duo 2.4 GHz processor with 2GB of RAM in all our experiments.

Rectangle Algebra

In the first experiment we benchmarked our solver on the RA. We randomly generated instances varying in size from 20 to 50 nodes, the average degree (connections per node) from 5 to the maximum degree, and the average label size (base relations per relation) from 10 to 160. For each setting we generated 5 instances, a total of 2,240 instances. We set a time limit for each instance of 1 hour.

The first observation we make is that there is a phase-transition where randomly generated instances change form mostly consistent to mostly inconsistent. However, unlike for previously evaluated calculi the phase transition does not depend on the average degree but on the average label size. One reason for this might be that calculi with a small number of base relations (such as RCC-8 or IA) do not allow us to vary the average label size too much as there are only 8 resp. 13 different labels. Therefore, it wasn't possible previously to discover this fact. For the RA we have 169 labels to choose from which allows us to greatly vary the average label size. As expected, the hardest instances are found around the phase transition region, most of them cannot be solved within the time limit. For the average degree, it seems that the higher the degree, the harder the instances. Note that for space reasons, we only show the runtimes and not the phase transitions. However, the phase transition regions lie within the regions of the largest runtimes (see Fig.3).

When comparing the performance of our solver with GQR, it turns out that our solver is considerably faster and that it can solve a number of instances that GQR cannot solve. We compared the average and median runtime on those instances that both solvers solved. In addition, we plot the failure rate for both solvers (see Fig. 2).

Block Algebra

In the second experiment we benchmarked our solver on the BA. We randomly generated instances of size 10 and 20 nodes and varied the average degree from 5 to the maximum degree, and the average label size from 10 to 2160. For each setting we generated 5 instances, a total of 1290 instances with a time limit of 1 hour per instance. Again, the phase transition and the location of the hard instances depends on the average label size and not the average degree, and the higher the average degree the harder the instances. It turns out that all instances that are not in the phase transition region can be easily solved (see Fig. 1).

Summary and Discussion

Large qualitative calculi inevitably occur when we combine different calculi in order to get more expressive formalisms that cover different aspects of space or time. There has been some discussion in the research community whether such large calculi are desirable and feasible or whether they defeat the purpose of a qualitative representation that makes

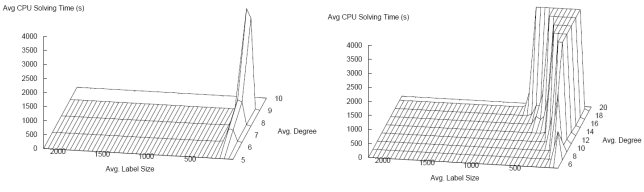

Figure 1: Avg CPU Time for 10 and 20 Nodes in BA

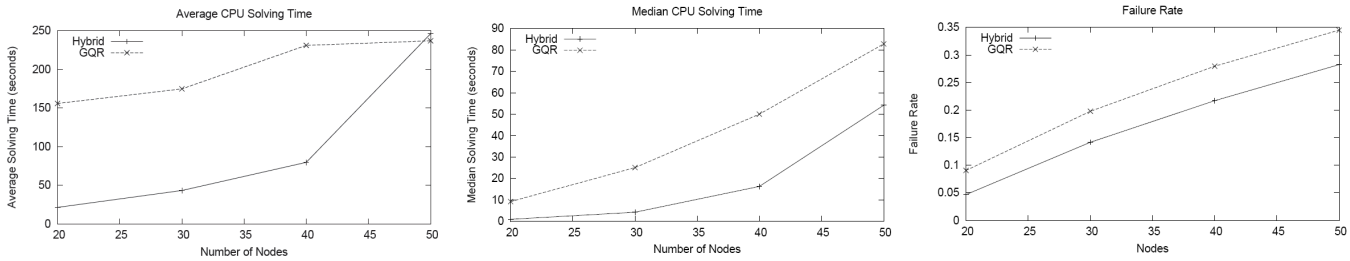

Figure 2: Comparing Average, Mean CPU Time and Failure Rate for RA Networks between Hybrid Solver and GQR

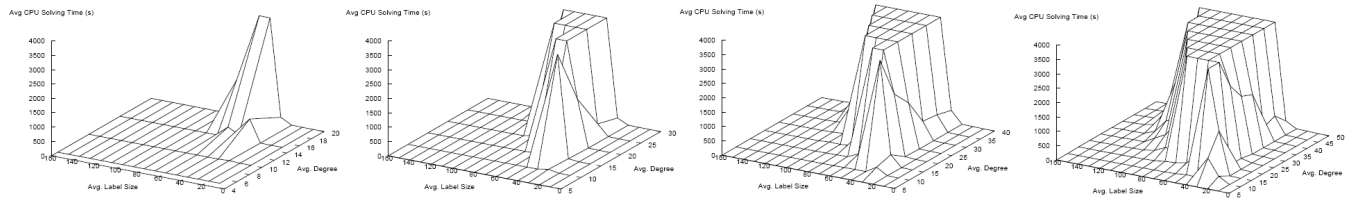

Figure 3: Hybrid solver CPU Time for RA networks of size 20, 30, 40 and 50, over avg. label size and avg. degree.

only a small number of distinctions. We have argued that large calculi are useful provided that they are (1) based on a reasonably small and easy to understand and to remember set of relations, and (2) that effective and efficient reasoning is possible. We can assume that the first condition is met whenever a large calculus is an obvious combination of smaller calculi that satisfy the condition. Therefore, the main criterion for combined calculi is whether reasoning can be done effectively and efficiently. One problem for large calculi is that tractable subsets, which have previously been the key element for efficient solutions to the NP-hard reasoning problems, may not only be hard or impossible to find but also to store and retrieve. Hence we have explored an alternative solution method which is based on transformation of a set of spatial or temporal constraints into a propositional formula and which does not rely on tractable subsets. We proposed a novel procedure that utilizes a recent method on dividing networks into a number of equivalent smaller subnetworks, and then develop a transformation of the smaller networks into an equivalent propositional formula. We prove that both steps can be applied and produce the correct result.

We analysed some very large calculi, the Rectangle Algebra and the Block Algebra and evaluated how well our procedure can decide consistency of randomly generated instances. As for previous studies, the randomly generated instances show a phase transition behaviour with the hardest instances around the phase transition region. Interestingly, we found that the phase-transition region does actually not depend on average degree but on average label size, something which can only be detected for large calculi. It turned out that our procedure can easily solve most instances, while those instances around the phase transition region couldn't be solved in reasonable time. For the Rectangle Algebra, our procedure performs better and solves more instances than other methods, while for the Block Algebra, our procedure is so far the only method that can produce a solution at all.

Given the huge number of relations of these algebras, our results are a clear indication that large calculi can be practically useful and that efficient reasoning is possible for these calculi. It is unsurprising that instances in the phase transition are very hard, since even for much smaller calculi these instances are often too hard to solve. To the contrary, these instances are actually useful as they provide easy to generate benchmarks for modern SAT solvers.

References

Allen, J. F. 1983. Maintaining knowledge about temporal intervals. *Commun. ACM* 26(11):832–843.

Balbiani, P.; Condotta, J.-F.; del Cerro, L. F. 1998. A model for reasoning about bidimensional temporal relations. *KR*, 124–130.

Balbiani, P.; Condotta, J.-F.; and del Cerro, L. F. 2002. Tractability results in the block algebra. *J. Log. Comput.* 12(5):885–909.

N. Eén and N. Sörensson. An extensible SAT-solver. *SAT'03*, 502–518, 2003.

Gerevini, A., and Renz, J. 2002. Combining topological and size information for spatial reasoning. *Artif. Intell.* 137(1-2):1–42.

Li, J. J.; Huang, J.; and Renz, J. 2009. A divide-and-conquer approach for solving interval algebra networks. *IJCAI*, 572–577.

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

Pham, D. N.; Thornton, J.; and Sattar, A. 2008. Modelling and solving temporal reasoning as propositional satisfiability. *Artif. Intell.* 172(15):1752–1782.

Randell, D. A.; Cui, Z.; and Cohn, A. G. 1992. A spatial logic based on regions and connection. *KR*, 165–176.

Renz, J., and Nebel, B. 2007. Qualitative Spatial Reasoning using Constraint Calculi. in Aiello, Pratt-Hartmann, van Benthem (eds) *Handbook of Spatial Logics*, Springer, 161-215.

Renz, J. 2007. Qualitative Spatial and Temporal Reasoning: Efficient Algorithms for Everyone. *IJCAI*, 526–531.

Renz, J., and Li, J. J. 2008. Automated complexity proofs for qualitative spatial and temporal calculi. *KR*, 715–723.

Renz, J.; Rauh, R.; and Knauff, M. 2000. Towards cognitive adequacy of topological spatial relations. *Spatial Cognition*, *LNCS* 1849, 184–197. Springer.

Westphal, M., and Wölfl, S. 2009. Qualitative csp, finite csp, and sat: Comparing methods for qualitative constraint-based reasoning. *IJCAI*, 628–633.

Wölfl, S., and Westphal, M. 2009. On combinations of binary qualitative constraint calculi. *IJCAI*, 967–973.