Efficient Methods for
Qualitative Spatial and Temporal
Reasoning

Jason Jingshi Li

A thesis submitted for the degree of
Doctor of Philosophy at
The Australian National University

August 2010
To my wife Helen and son Lachlan:

I love you to pieces, to you I dedicate this thesis.
Acknowledgements

First and foremost I would like to thank my supervisor Jochen Renz for his excellent supervision. It has been a great pleasure to work with him throughout the tenure of my PhD candidacy. One can hardly ask for a better supervisor.

I would also like to give my thanks to:

My other supervisory panel members Jinbo Huang and Jussi Rintanen, for their great guidance and support over the last three years.

My collaborators over the past three years: Norman Foo, Tomasz Kowalski, Rex Kwok, Sanjiang Li, Pontus Stenetorp, Matthias Westphal and Stefan Wölfl for the numerous fruitful discussions, advices and productive hours working together.

Tony Flynn, Patrik Haslum and John Lloyd, for their invaluable words of wisdom.

Debdeep Banerjee, for the intellectual companionship, wild discussions and happy times together.

Michelle Moravec, the administrator at Computer Science Laboratory at the Australian National University, for making the life at the department easier for everyone, and reminding us that the university exists to serve its students.

The Australian National University and National ICT Australia (NICTA) for the stimulating intellectual environment, excellent student services, and generous financial support that made this work possible.

Lastly, I would like to thank my family, for all their understanding and support throughout the years. Especially to my wife Helen, for all the happiness we shared, journey we travelled, and love we cherished.
Acknowledgements

I turn any profession I would like to pursue on our continent. I would like to
acknowledge the contributions of those who have
provided assistance. I am grateful for the efforts of the
various organizations that have supported this project.

I would like to thank the following individuals and
groups for their contributions:

[List of acknowledgments]

I would also like to acknowledge the financial
support provided by [granting agency].

Finally, I would like to express my gratitude to all
those who have helped make this project possible.
Remarks

Parts of this thesis have already been published in international conference proceedings, or in preparation to be submitted to academic journals. These papers are results of collaboration with other authors. While it is impossible to exactly dissect each of these papers according to which co-author had which idea, the papers are largely based on my ideas and contributions unless otherwise stated. This is clearly demonstrated by the non-alphabetical order of authors in all these papers.

Abstract

Qualitative aspects of spatial or temporal information such as the distance between points, duration between events, and topology between regions can be modelled by a qualitative calculus. It represent the relations between various entities in space or time by a set of base relations, and ambiguity or incomplete knowledge can be expressed as a disjunction of different possible base relations. This thesis investigates the reasoning problem of such qualitative spatial or temporal calculi, where the task is to determine whether a given set of spatial or temporal information is consistent. The main challenge is to deal with the large number of possible relations between the spatial or temporal entities, and to find the solution in an efficient manner.

This thesis approaches this challenge from several directions. We first investigate if there are some relations in a qualitative calculus which could form constraint networks that make the reasoning problem difficult. This would help us to determine whether tractable reasoning is actually possible.

Secondly, we investigate algebraic properties of a qualitative calculus. We present a condition of the qualitative calculus that ensures no inconsistencies can be introduced when combining two constraint networks that share information in common.

Thirdly, we reverse this process to decompose a constraint network into many smaller components to decide consistency. We evaluate our results using a well known qualitative temporal calculus, Allen’s Interval Algebra, and show that this leads to a significant improvement to current state of the art.

Finally, we apply the network decomposition approach to large qualitative calculi such as Rectangle Algebra and Block Algebra. The latter was previously considered too large for any efficient reasoning. We show that our approach works well for most instances of these calculi, and that efficient reasoning with these highly expressive spatial
calculi is indeed possible.
# Contents

Acknowledgements vii  
Remarks ix  
Abstract xi  

## 1 Intelligence, Knowledge, Space and Time

1.1 Qualitative Spatial and Temporal Reasoning 1  
1.2 Reasoning About Constraints 3  
1.2.1 Tractable Reasoning 5  
1.2.2 Efficient Reasoning 6  
1.3 About this thesis 7  
1.3.1 Thesis Organization 8  

## 2 Background

2.1 A Qualitative Calculus 11  
2.1.1 Domain 13  
2.1.2 Algebra 13  
2.1.3 Weak Representations 14  
2.2 Constraint Satisfaction 16  
2.2.1 Constraint Networks in QSTR 17  
2.2.2 Algebraic-Closure / Path-Consistency 18  
2.2.3 Methods for Consistency 19  
2.2.3.1 Constraint-based approach 20  
2.2.3.2 SAT-based approach 21
2.3 Identifying Tractable Subsets ........................................... 22

3 Automated Complexity Proofs for Qualitative Calculi ................. 25
  3.1 Computational complexity ........................................... 26
    3.1.1 Existing methods for proving complexity ....................... 27
    3.1.2 Novelty of our approach ...................................... 30
  3.2 A General Procedure for Proving NP-hardness ...................... 31
    3.2.1 Step 1: Finding polarity constraints ......................... 33
    3.2.2 Improvements in finding polarity constraints ................ 36
    3.2.3 Step 2: Finding clause constraints .......................... 37
    3.2.4 Step 3: Applying closure .................................... 42
  3.3 Empirical Evaluations of the Procedure ........................... 44
  3.4 Limitations of our Procedure .................................... 46
  3.5 Conclusion .......................................................... 47

4 On Extending and Combining Constraint Networks ...................... 49
  4.1 Combining networks ................................................ 49
  4.2 Formal Preliminaries .............................................. 53
  4.3 Extending Networks ................................................ 56
    4.3.1 3-extensibility .............................................. 57
    4.3.2 4-extensibility .............................................. 62
      4.3.2.1 Counter Example: The B_9 Algebra ....................... 62
      4.3.2.2 Strong 4-extensibility .................................. 64
      4.3.2.3 One-shot extensibility .................................. 66
  4.4 Finding an Atomic Refinement .................................... 67
    4.4.1 One-shot proto-extensibility ................................. 67
    4.4.2 Refining the intersections ................................... 69
    4.4.3 Empirical Evaluations ....................................... 73
  4.5 Summary and Future Work ......................................... 74
5 On Partitioning Constraint Networks

5.1 Decomposing Networks

5.1.1 A Theoretical Property

5.1.2 Decomposing Networks

5.1.3 Implementation

5.2 Tractability of Deciding Consistency

5.2.1 Recursive Conditioning

5.2.2 Solving the SAT Encoding

5.3 Experimental Results

5.3.1 Performance of Compact SAT Encoding

5.3.1.1 Smaller Networks

5.3.1.2 Larger Networks

5.3.2 Benchmarks of Bounded Treewidth

5.4 Conclusion and Discussion

6 Reasoning with Large Qualitative Calculi

6.1 Why Large Qualitative Calculi

6.2 Current Approaches for Large Calculi

6.2.1 Combinations of Calculi

6.2.2 Propositional SAT Encoding

6.2.3 The Divide-and-Conquer Approach

6.3 A Hybrid Procedure for RA and BA

6.3.1 Network Decomposition

6.3.2 Point-Based Representation

6.3.3 Encoding to Propositional Satisfiability

6.3.4 Example

6.4 Empirical Evaluation

6.4.1 Rectangle Algebra
Contents

6.4.2 Block Algebra .................................................. 117
6.5 Summary and Discussion ........................................ 117

7 Conclusion ................................................................ 119

7.1 Summary of Contributions .................................... 119
    7.1.1 On Algorithms for Automated Complexity Proofs .... 119
    7.1.2 On Network Amalgamation .................................. 120
    7.1.3 On Network Decomposition ............................... 121
    7.1.4 On Reasoning with Networks of Large Qualitative Calculi . 121

7.2 Direction for Future Work ...................................... 122

Bibliography .............................................................. 125
1.1 Qualitative Spatial and Temporal Reasoning

Spatial and temporal information is ubiquitous in our lives. Everyday, we reason with such information to move about our house, get to work, fetch a beer from the fridge and interact with others. We also use our spatial and temporal knowledge to plan our trips, schedule our work, play sports and navigate in unfamiliar terrain. We continuously observe the world around us and our perceptions influence our decisions. Because of this level of familiarity, we have developed a certain intuitive understanding about space and time. We know that we cannot be on both sides of a wall, that one instant in time cannot occur both before and after another, and that if a wallet is in a bag, and the bag is in a box, then the wallet cannot be outside the box. Today's computers, or to be precise the programs that run on computers, have no such intuitions. Therefore, understanding and inferring spatial and temporal information is an important component in the development of artificial intelligence, where the ultimate aim is to create systems that mimic and surpass human rationality.

Research in qualitative spatial and temporal reasoning (QSTR) arose from the need to reason about entities in space and time in a qualitative, non-numerical manner; one that reflects human understanding and common-sense. Similar to human reasoning, QSTR represents knowledge about space and time using a limited vocabulary, describing only the essential spatial or temporal relationships. An example would be a statement such as
"Canberra is part of Australia and is girt by the shires of New South Wales". Here, the spatial relationships between Canberra, Australia, and the shires of New South Wales are described qualitatively, but one can develop an intuitive sense of how each region relates with all the others. In contrast to fuzzy approaches which make approximations of real values, QSTR makes only as many distinctions as necessary [66]. QSTR has been shown to be close to human common-sense reasoning [30], and suited to model incomplete knowledge and uncertainty about spatial or temporal information [15]. In short, QSTR is the art and practice of making the computer think more humanly about space and time.

A qualitative calculus is used to develop a qualitative abstraction of spatial or temporal reality [41]. It selects a set of spatial or temporal entities to be its domain. Examples of these domains are the set of all points on a plane, continuous segments along a directed line, or regions in 3D topological space. The domain is usually an infinite set as there are, e.g., infinitely many points along a line, infinitely many regions on a plane, etc. In a binary qualitative calculus, a set of binary spatial relations is selected which forms a partition of the domain, i.e., between any two elements of the domain exactly one of the relations hold. Relations with this property are called base relations or atomic relations. It is common to select the partition according to a particular aspect of space, such as topology, direction or distance and to make the partition such that relations correspond to spatial distinctions which are commonly made by humans. In this thesis we restrict the qualitative calculi to only binary qualitative calculi.

To date there have been many qualitative calculi proposed for different spatial or temporal domains. Consider the following incomplete list in alphabetical order: the 9-intersection model for topological relations between "simple regions" in 2D space [14]; the Allen's Interval Algebra [1] for intervals along a directed line; the Closed Disk Algebra [12] for closed disks in 2D topological space; the Cardinal Direction Calculus and its variants [23; 40; 68], the Double Cross Calculus [17], the LR Calculus [39], the Star Calculus [62] and the OPRA Calculus [49] for directions between points in 2D space;
the Point Algebra [73] for points along a directed line; the Qualitative Trajectory Calculus [72] for the moving points in 2D space; the Rectangle Calculus [4] for rectangles in 2D Cartesian space; and the Region Connection Calculus [57] for topological relationships between regions. Each calculus models relationships of a particular aspect of space between elements in its domain.

There are many applications developed for the various qualitative calculi. For brevity we will only enumerate the two best known examples. The oldest and also the best known is Allen’s Interval Algebra [1] used in temporal reasoning. It has been applied in automated planning [3; 2], computational biology [21], multimedia planning [16], interactive system design [55], diagnosis [53], and natural language processing [69]. The qualitative spatial calculus RCC8 [57] is involved in applications in spatial information systems [24], navigation, computer vision and natural language processing [43], qualitative simulations of physical processes [10], and visual programming languages [22]. In short, spatial and temporal applications, a qualitative approach is preferred when precise, numerical information is either not required or not available.

In summary, today we have a rich variety of qualitative calculi modelling many different aspects of space and time, where each uses a finite vocabulary to describe spatial and temporal scenarios. They contribute to the development of intelligent systems that understand and reason about spatial and temporal information.

1.2 Reasoning About Constraints

An intuitive representation of qualitative spatial or temporal information is to describe it as a set of constraints. That is, given a set of spatial or temporal entities, we specify what relations must hold between the entities. In turn, there are some constraints that are inherent to the spatial and temporal domain, as they specify what scenarios are permitted and what are forbidden for a set of spatial or temporal entities. To go back to the previous example, if a wallet is inside a bag, and the bag is inside a box, then the wallet cannot
be outside the box. The set of constraints over a set of spatial or temporal variables in a qualitative calculus can be represented as a network, where the nodes denote the variables, and the label on the edge between any two nodes denotes what relations of the calculus may hold between the two variables. The label can be an atomic relation, where the exact relation between the variables is known, or a disjunction of more than one atomic relations that outlines the possible relations which can hold between the variables. When nothing is known about the relation between the two variables, we denote the label with the \textit{universal relation}, which is the disjunction of all possible atomic relations.

For a set of constraints of a qualitative spatial or temporal calculus, we want to know whether we can assign variables to elements in the domain where all the constraints are satisfied. This is known as the \textit{consistency problem}. As there are typically infinitely many elements in the domain of a qualitative calculus, it is not feasible to check them one by one to determine whether they are part of the solution to the problem. Instead, we transform the problem to reason about the relations between the variables. For example, we know that if A is inside B, and B is inside C, and C is inside A, then we cannot find any 3D topological regions for A, B and C to satisfy the constraints. However, if we remove the constraint that C is inside A, then there are infinitely many solutions in 3D topological space, and we do not have to explicitly search for the exact regions for A, B and C.

The consistency problem is a fundamental reasoning problem in QSTR. This is because many other problems can be transformed into the consistency problem. One example is query answering. To go back to the same example, given a spatial knowledge-base where we know that A is inside B and B is inside C, suppose we want to query whether A is inside C. To answer the query, the standard knowledge-representation technique is to negate the query and merge it with the existing knowledge base. If the constraint A is not inside C contradicts the knowledge base, then we know that the answer to our original query is true.
1.2.1 Tractable Reasoning

Computational complexity is the field in computer science where we mathematically evaluate how difficult a problem can be. The complexity analysis gives us an indication of how likely we will be able to solve a problem. Nebel in 1996 [50] argued that in addition to providing a feasibility analysis on the class of problems, complexity analysis also gives us a deeper understanding of the structures of a problem.

In the theory of computational complexity, we are interested in the worst time and space complexity of a problem. That is, for any given problem, in the worst case what are the time and memory required to actually solve the problem. This is given as a function of the size of the problem. Currently we have a clear line that divides the easy problems and the hard problems. Specifically, we say that a problem is easy, or tractable, if there is a worst-case polynomial algorithm that can solve the problem. Otherwise the problem is hard or intractable.

A major breakthrough in the history of computer science was Cook’s theorem [9], which showed that the Boolean satisfiability problem (SAT) is NP-complete. From this it follows that a problem can be shown to be NP-hard (at least as hard as the hardest problem in NP), if a restricted version of the problem can be transformed into a SAT problem. If a problem is NP-hard, then we know that there is no polynomial algorithm to solve the problem unless \( P = NP \). As the affirmative answer to that proposition is considered extremely unlikely by the research community, it is reasonable to assume that the NP-hard problems are difficult problems in computer science.

In general, reasoning with a qualitative calculus can be undecidable [25]. For many qualitative calculi the consistency problem is decidable, but NP-hard. Vilain and Kautz in 1986 showed that reasoning with the full Allen’s Interval Algebra is NP-hard [73]. The full RCC8 was also shown to be NP-hard by Renz and Nebel in 1997 [63]. In both cases the authors showed that a restricted version of the consistency problem of the respective qualitative calculus can be transformed into a 3SAT problem. However, with the myriad of qualitative calculi being proposed in the last few years, there is an increasing need
to look for an automated approach to decide whether reasoning with a new qualitative calculus is NP-hard. A part of this thesis addresses this issue by proposing an automated algorithm to detect sufficient conditions for NP-hardness for a new qualitative calculus.

1.2.2 Efficient Reasoning

In the case where reasoning with a qualitative calculus is NP-hard, this should not signal the end of efficient reasoning. Rather, it is the beginning. The most obvious example is the progress of the Boolean Satisfiability (SAT) solver. The SAT research community have shown that tremendous advances can be made for solving even NP-complete problems. The SAT solvers today are by order of magnitude more efficient than SAT solvers 10 years ago. Similar to many industrial SAT instances, spatial and temporal information all possesses some inherent structure. Therefore, we can exploit these structures to make the reasoning algorithm more efficient.

One of the first major advances in efficient reasoning in QSTR is the identification of the tractable subsets, where we restrict the set of the relations permissible in the constraint network, and deciding consistency networks with this restricted set of relations is polynomial to network size. Therefore, we only have to perform a search on the constraint network where at each step we restrict the relation of an edge of the network to be part of the tractable subset. The network is consistent if all the edges are part of the tractable subset and do not fail the tractable algorithm. Nebel showed that the use of the maximal tractable subset ORD-Horn of Allen’s Interval Algebra significantly speeds up the reasoning process [51]. Later, Renz and Nebel found that similarly the maximal tractable subsets of the RCC8 calculus are the key for efficiently solving large instances of the calculus [65]. In this thesis we will show that there are other inherent structures within a qualitative calculus that we can exploit, and we propose methods for deciding consistency that surpass the current state-of-the-art for the most difficult instances of some of the best known qualitative calculi available.
1.3 About this thesis

Current constraint-based approaches for solving instances of a qualitative calculus generally fail with a large instance with many variables, or when the qualitative calculus is too large in that the calculus contains too many base relations. The challenge is first to determine whether tractable reasoning is possible for a problem in QSTR. If the answer is negative, we then need to be able to deal with the large number of possible relations between numerous spatial or temporal entities, and to find the solution in an efficient manner.

This thesis addresses this challenge by looking into properties of constraints of a qualitative calculus. By doing so, we find out what makes reasoning with a qualitative calculus hard, how do the constraints interact with each other when we combine them together, and how we can exploit the structures in a constraint network that allow us to reason with it more efficiently.

We first provide an automated procedure to detect sufficient conditions for NP-hardness of a qualitative calculus. The procedure allows us to automatically detect if reasoning with all the relations of a qualitative calculus is NP-hard, and if a tractable subset is maximal. Empirically we showed that the procedure successfully identified the maximality of tractable subsets in Allen’s Interval Algebra, Ligozat’s Cardinal Direction Calculus and the Region Connection Calculus RCC8.

Secondly, we look at how two constraint networks of a qualitative calculus can be combined together without introducing inconsistencies. We provide an algorithm for verifying this property when the problem is restricted to allowing two networks sharing only two nodes in common. Incidentally, this also allows us to prove some other important properties of the qualitative calculus, namely tractability results of a relation can be transferred to its closure, and the relation algebra of the calculus is representable.

We follow up by reversing the process of combining constraint networks. We show that by using a theoretical property, we can decompose a given constraint network in
QSTR into several smaller networks. This was previously impossible because a qualitative calculus enforces a constraint between any two variables in the network. Our work shows that we can bypass some of these constraints when deciding consistency. Empirically, this leads to a significant performance improvement upon the state-of-the-art for solving the most difficult class of problems in Interval Algebra. We also show that this leads to the first fixed-parameter-tractability result in QSTR.

Finally, we tackle some of the largest known qualitative calculi. These calculi contain many base relations, leading to a combinatorial explosion of constraints for even relatively smaller instances involving only a few variables. We show that by combining the network-decomposition approach with a novel SAT-encoding, we are able to efficiently solve problems that were previously thought impossible to complete in a reasonable time. Our result on these calculi indicates efficient reasoning with these highly expressive qualitative spatial representations is not beyond reach.

1.3.1 Thesis Organization

This thesis is organized as follows. In Chapter 2, we summarize the essential aspects of formalizing spatial and temporal knowledge as a qualitative calculus, how information can be represented as a constraint network over a qualitative calculus, and how we can perform reasoning with such representations.

Chapter 3 introduces an automated method for analyzing the computational complexity of a qualitative calculus.

Chapter 4 investigates whether a constraint network can be combined without introducing consistencies. We reverse this process in Chapter 5, where a constraint network can be decomposed into many separate networks for deciding consistency.

Chapter 6 is concerned about reasoning with constraint networks over very large calculi. It looks at whether it is possible to perform efficient reasoning for these very expressive spatial representations.

We conclude in Chapter 7 with a summary of what has been achieved in this thesis,
and suggest possible directions for future work.

Background

In this Chapter we study a qualitative calculus, in reasoning problems, methods and tools. We also give a brief analysis of the methodologies that the field is related to, and the particular ones which are used in this thesis.

2.1 A Qualitative Calculus

A qualitative calculus assigns to relations on certain domains over a given set of qualitative relations the domain $D$. The base relation $D$ of the system forms a partition of $D$ or $D$, but it is mainly exhaustive and thus does not use CEPPO. That is, the two elements of the domain case of the basis relation $D$ is $D$. The base

The qualitative calculus are also known as domain relations or fully exhaustive in theorems. Formally, a qualitative calculus is a tuple $(A, D, u)$ where $A$ is a qualitative universe, $D$ is a set which exhaustively covers the elements in the domain, and $u$ is a weak representation of $A$ over $D$. The weak representation can be seen as a mapping from elements of the classes of the universe $A$ to the set $D$.

For example, a spatial calculus RST is [27]. This qualitative type of extended relations to the domain $A$ with some qualitative relations to represent the topological relationships between the objects. The base relations are exhaustive and thus do not use CEPPO. The base relations also allow actions such as move, rotate, and constraint, such as path constraints. This qualitative type of extended relations to the domain $A$ with some qualitative relations to represent the topological relationships between the objects. The base relations are exhaustive and thus do not use CEPPO. The base relations also allow actions such as move, rotate, and constraint, such as path constraints.
Chapter 2

Background

In this Chapter we briefly summarize what is a qualitative calculus, its reasoning problems, methods and tools. We also give brief accounts of the terminologies that the field developed in the last twenty years and which are used in this thesis.

2.1 A Qualitative Calculus

A qualitative calculus uses an algebra of relations to define relationships over a given set of spatial or temporal entities, the domain $\mathcal{D}$. The base relations $\mathcal{B}$ of the algebra form a partition of $\mathcal{D} \times \mathcal{D}$ that is jointly exhaustive and pairwise disjoint (JEPD). That is, between any two elements of the domain exactly one base relation holds [41]. The base relations are also known as atomic relations or basic relations in literature. Formally, a qualitative calculus is a triple $(\mathbb{A}, \mathcal{U}, \mu)$ where $\mathbb{A}$ is a nonassociative algebra; $\mathcal{U}$ is a set which corresponds to all the entities in the domain; and $\mu: \mathbb{A} \to \mathcal{U}$ is a weak representation of $\mathbb{A}$ over $\mathcal{U}$. The weak representation can be seen as a mapping from elements of the algebra $\mathbb{A}$ to the set $\mathcal{U} \times \mathcal{U}$.

For example, a spatial calculus, RCC8 [57], uses a topological space of extended regions as the domain. It defines eight base relations to represent the topological relationships between the regions. The base relations are disconnected, externally connected, partial overlap, equal, tangential proper part, non-tangential proper part and the converse relations of the latter two. They are illustrated in Figure 2.1. It is obvious that between any two regions only one of these relationships holds.
In an alternative example representing temporal information, Allen’s Interval Algebra [1] is a qualitative calculus modelling the relationships between intervals along a directed line. The domain, $D$, is the set of intervals along a continuous directed line. The relation algebra of the calculus contain 13 base relations $B$, which are $\text{before}$, $\text{meets}$, $\text{overlaps}$, $\text{starts}$, $\text{finishes}$, $\text{during}$, their respective converses, and the $\text{equal}$ relation (Figure 2.2).

This section shall review in detail the formal foundations of a qualitative calculus and its reasoning mechanisms. We will describe the three components of the qualitative calculus: its domain, algebra, and weak representation.
2.1.1 Domain

The domain \((\mathcal{D})\) of a qualitative calculus is the set of possibly infinite spatial or temporal entities that the calculus wishes to model. It can be the non-empty point-sets in an arbitrary topological space, the set of 2D closed disks or the set of all possible intervals along a line. The set may be a collection of any spatial or temporal entity, or a hybrid of both. However, there must be some intuitive relationships between the entities, as it does not make much sense to enquire about the relationship between a time point and a region, unless they are tied together in a wider context. The actual relations between elements in the domain is modelled by a relation algebra, which we will elaborate in the following section.

2.1.2 Algebra

The algebra of a qualitative calculus defines a set of relations that we use to model relationships between elements in the domain. The general notion about this algebra of relations is known as nonassociative algebra. Its formal definition is as follows:

Definition 1 (nonassociative algebra \([35]\)). A nonassociative algebra \((NA)\) is an algebraic structure \(A = (\mathcal{A}; \land, \lor, ;, ,^-, ^+, 1, 0, 1)\), such that

- \((\mathcal{A}; \land, \lor, -, 0, 1)\) is a Boolean algebra
- \((\mathcal{A}; , ^-, 1)^+\) is an involutive groupoid with unit, that is, a groupoid satisfying the following equations
  - \(x ; 1^+ = 1^+ ; x\)
  - \(x^- = x\)
  - \((x ; y)^- = y^- ; x^-\)
- the operations \(;\) (multiplication) and \(^-\) (inverse) are normal operators, that is, they satisfy the following equations
- \( x;0 = 0 = 0;x \)
- \( 0^* = 0 \)
- \( x;(y \lor z) = (x;y) \lor (x;z) \)
- \( (x \lor y);z = (x;z) \lor (y;z) \)
- \( (x \lor y)^* = x^* \lor y^* \)

- the following equivalences hold

\[
(x;y) \land z = 0 \iff (z;y^*) \land x = 0 \iff (x^*;z) \land y = 0
\]

These are called Peircean laws or triangle identities.

A nonassociative algebra is a relation algebra (RA) if the multiplication operation (\( ; \)) is associative. Detailed analysis on relation algebras and nonassociative algebras can be found at [26] and [47].

### 2.1.3 Weak Representations

The algebra defines the set of base relations that the calculus use to partition the cross product of the domain \( \mathcal{D} \times \mathcal{D} \). The base relations are also called the atomic relations. Each base relation consists of a set of tuples in \( \mathcal{D} \times \mathcal{D} \). Furthermore, any element of the set of tuples \( \mathcal{D} \times \mathcal{D} \) is assigned to exactly one base relation. This property is known as being jointly exhaustive and pairwise disjoint (JEPD). This partition scheme of the domain is known as the weak representation of the corresponding algebra.

Formally, let \( \mathbf{A} \) be an NA. For any set \( U \), called a domain, let \( \mathcal{R}(U) \) be the algebra \((\varnothing(U \times U), \cup, \cap, \circ, -^1, \Delta, \emptyset, U \times U)\), where the operations are union, intersection, composition, complement, converse, the identity relation, the empty relation and the universal relation (all with their standard set-theoretical meaning). More specifically, the composition (\( \circ \)) of two relations \( S \) and \( T \) is defined as \( S \circ T = \{(a,b) | \exists c : (a,c) \in S \land (c,b) \in T\} \). Notice that since composition is associative, \( \mathcal{R}(U) \) is an RA. We say
that \( A \) is \textit{weakly represented over} \( U \) if there is a map \( \mu: A \rightarrow \wp(U \times U) \) such that \( \mu \) commutes with all operations except multiplication (\( ; \)), for which we require only

\[
\mu(a ; b) \supseteq \mu(a) \circ \mu(b)
\]

This property of weak representation gives rise to a notion of \textit{weak composition} of relations (\( \circ \)) [61]. Intuitively, it can be understood as for variables \( a, b, c \), given that the relation \( R \) holds from \( a \) to \( b \), and the relation \( S \) holds from \( b \) to \( c \), the weak composition tells us the possible relations from \( a \) to \( c \). More formally, for \( R, S \in \mu[A] \), we define \( R \circ S \) to be the smallest relation \( Q \in \mu[A] \) containing \( R \circ S \). Every NA has a weak representation, for example a trivial one, with \( U = \emptyset \). Of course, interesting weak representations are nontrivial, and typically injective. A weak representation is a \textit{representation} if \( \mu \) is injective and the inclusion above is in fact equality, that is, if \( \mu \) is an embedding of relation algebras. In such a case weak composition equals composition, and that is expressed by saying that weak composition is \textit{extensional}. Not every NA, indeed not every RA, is \textit{representable}.

Although weak representations are not as interesting as representations, curiously, it is the former that gave rise to a notion of \textit{qualitative calculus}, which is a triple \((A, U, \mu)\) where \( A \) is an NA, \( U \) is a set and \( \mu: A \rightarrow U \) is a weak representation of \( A \) over \( U \). Since \((A, U, \mu)\) is notationally cumbersome, we will later write \( A \) for both an NA and a corresponding calculus \((A, U, \mu)\), if \( U \) and \( \mu \) are clear from context or their precise form is not important. A calculus \((A, U, \mu)\) is \textit{extensional} if \( \mu \) is a representation of \( A \). Notice that if \((A, U, \mu)\) is extensional, then \( A \) is an RA, indeed a representable one. The converse need not hold, as the example of RCC8 demonstrates that its algebra is a representable relation algebra, but the calculus is not extensional.

All NAs considered in QSTR are assumed to be finite and such that \( 1' \) (identity) is a base relation. These are severe restrictions on the class of NAs, but natural from a spatial calculi point of view.
2.2 Constraint Satisfaction

Given a qualitative calculus, knowledge about the entities of the domain can be provided in the form of constraints. An example of binary constraint that involves two variables is “World War I is before World War II”. A query about this information can be formalized as a constraint satisfaction problem (CSP).

A CSP is made up of a set of variables, $V$, over a domain, $\mathcal{D}$, and a set of constraints, $\Theta$. The goal is to find a solution to the CSP, which is an instantiation of the variables $V$ in the domain $\mathcal{D}$ such that all the constraints $\Theta$ are satisfied.

An example of a temporal CSP for Allen’s Interval Algebra is the following. “Peter reads the newspaper during breakfast, and goes to work after breakfast.”. The domains are the possible time periods for the three events “reading”, “breakfast” and “work”. The constraints are “reading” during breakfast and “work” after “breakfast”.

The CSP is consistent if and only if there is an instantiation in the domain that all constraints are satisfied. Hence the CSP in the example is consistent if we can find intervals corresponding to the three variables along a line such that the no constraint is violated. If we show that it is impossible to find such an instantiation, then the CSP is inconsistent.

For CSPs where the domain of the variables is finite, they can be solved by the backtracking method, where one successively instantiates variables with values in the domain until either all variables are instantiated or an inconsistency is detected. In the case where all variables are instantiated, the solution to the CSP has been found. When an inconsistency is detected, we simply instantiate the variable with the next value in the domain. If all possible instantiations of the current variable have led to inconsistency, then the previous variable becomes the current variable and we instantiate it with the next value in the domain and repeat the process.

Unfortunately, CSPs in QSTR almost always involve infinite domains, as we assume space and time are continuous. This is a realistic assumption, as there can be infinitely
many possible regions on a given map, or infinitely many possible events along a time line. As a result the standard backtracking search technique described previously does not apply. Instead, we transform the problem into a different problem about relations between the variables on a constraint network. We will now describe this process of deciding consistency in QSTR through constraint networks.

### 2.2.1 Constraint Networks in QSTR

A binary CSP in QSTR can be represented by a *constraint network*. The network can be seen as a graph, where the nodes of the network denote the variables of the CSP, and the labels on the directed edges between two nodes describe the constraint (relation) that holds between them. If there are no constraints between two nodes, then the edge is labelled with the *universal relation*. On the other hand, if no label exists between two nodes, then it means that no relation is possible between the two variables. Recall the JEPD property of the qualitative calculus introduced earlier in this Chapter, which enforces that between any two objects in the domain exactly one base relation holds. Therefore, if no base relation is possible between two variables, then the CSP is inconsistent. Similar to a CSP, a constraint network is consistent if and only if there exists an instantiation in the domain where all constraints of the network are satisfied.

Formally, a constraint network $N$ over an NA $A$ is a pair $(V, \ell)$ where $V$ is a set of vertices (nodes) and $\ell : V^2 \rightarrow A$ is any function. Thus, a network is a complete directed graph labelled by elements of $A$. We will write $V_N$ and $\ell_N$ for the set of vertices of $N$ and its labelling function, respectively. For convenience we assume that the set $V$ of nodes is always a set of natural numbers. We will also frequently refer to the label on the edge $(i, j)$ as $R_{ij}$.

A network $N$ is a *sub-network* of another network $M$ if $V_N \subseteq V_M$ and the labelling $\ell_N$ of the edges of $N$ is exactly the same as the labelling $\ell_M$ of the same edges in $M$, i.e., $\ell_N(i, j) = \ell_M(i, j) \forall i, j \in V_N$. We write $N \subseteq M$ in such cases. $N$ is a *refinement* of $M$ if $V_N = V_M$ and $\ell_N(i, j) \subseteq \ell_M(i, j) \forall i, j \in V_N$. The intersection ($\cap$) of two networks
Background

$N_1, N_2$ is the maximal network that is a sub-network of both $N_1$ and $N_2$. When the two networks satisfy $V_{N_1} \cap V_{N_2} = V_{N_1 \cap N_2}$, we also define their union ($\cup$) to be the minimal\(^1\) network of which both $N_1$ and $N_2$ are sub-networks. A network $N$ is atomic if all the labels of $N$ are atomic (base) relations of $\mathbf{A}$.

Networks may be viewed as approximations to weak representations, indeed, if $\mu$ is a weak representation of $\mathbf{A}$ over a domain $U$, then $\mu[A]$ is an a-closed network over $\mathbf{A}$. An arbitrary network $N$ over $\mathbf{A}$ is consistent with respect to a weak representation $\mu$, if $N$ is a subnetwork of $\mu[A]$.

2.2.2 Algebraic-Closure / Path-Consistency

As the standard backtracking search for deciding consistency cannot be applied to a qualitative calculus with an infinite domain, we must rely on some other methods that allow us to approximate consistency. The first approximation of consistency for a constraint network in QSTR is to perform the algebraic closure (a-closure) algorithm [61]. A network $N$ is algebraically-closed (a-closed) if the following hold

1. $R_{ii}$ is the equality relation (identity element of $\mathbf{A}$)

2. $R_{ij} \circ R_{jk} \supseteq R_{ik}$ for any $i, j, k \in N$

The a-closure algorithm enforces a-closure by successively applying the following operation until a fixed point is reached:

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

where $i, j, k$ are nodes in the network and $R_{ij}$ is the label, hence the relation between $i$ and $j$. If after applying a-closure $R_{ij} = \emptyset$ for any $i, j$ in the network, then the network is inconsistent. Otherwise the network is a-closed.

\(^1\)Minimality here means that the network has the fewest vertices and the fewest edges possible, where we regard edges labelled with universal relations as invisible.
If weak composition equals composition, then the a-closure algorithm is equivalent to the more widely known path-consistency algorithm [48; 44; 45]. The difference lies in that path-consistency requires that for every instantiation of two variables, an instantiation can be found for the third variable. This cannot be guaranteed for an a-closed network where composition is a subset of weak-composition. Composition is equivalent to weak-composition in Allen’s Interval Algebra [61], but is a subset to weak-composition in RCC8 [38].

A-closure is seen as an approximation of consistency because it is an incomplete method for detecting inconsistency. That is, if a-closure detects an inconsistency, then the network is inconsistent, hence there exists no possible instantiation of the network in the domain. However a network could still be inconsistent, though the inconsistency is not detected by a-closure.

### 2.2.3 Methods for Consistency

To bypass the problem with infinite domain, existing reasoning methods for deciding consistency for many qualitative calculi such as IA rely on a key theoretical result, that consistency of atomic networks, where every pair of nodes is constrained by a single base relation, can be decided by a-closure/path-consistency [1]. Therefore, instead of searching over an infinite number of objects over each variable, we only need to search for an a-closed atomic refinement of the constraint network. Hence the basic approach for deciding consistency of constraint networks in QSTR is a backtracking search where branching is performed on the atomic relations in the label of each edge [32]. Note that this is different to the backtracking search described earlier where branching is performed on the domain of the variables. Here, algebraic closure is enforced immediately after each branch is created, and any inconsistency thus detected results in backtracking. The network is consistent if and only if a leaf of the search tree can be successfully reached where every edge has been assigned an atomic relation.

It should also be noted that a-closure does not decide consistency of atomic networks
for all qualitative calculi. In fact, in Chapter 4 we will show a sufficient condition (Theorem 3) where this cannot be the case for a qualitative calculus. However, this thesis is primarily focused on reasoning with qualitative calculi where this nice property is satisfied.

2.2.3.1 Constraint-based approach

Fundamentally, the constraint-based approach for deciding consistency is the backtracking search on the relations between the variables as described above. There have been numerous works in literature to make the process more efficient. After the initial proposal by Ladkin and Reinefeld in 1992 [32], there have been several improvements over this basic algorithm in the form of variable and value ordering heuristics—the order in which the next edge in the constraint network is selected, and the order in which atomic refinements are made to an edge [71]. However, the major improvement in the efficiency of constraint-based solvers has been the use of large and preferably maximal tractable subsets of the full set of relations. Consistency for networks that contain only relations from such sets can be decided in polynomial time, often with the algebraic closure algorithm. Instead of branching on the atomic refinements of every edge, the solver branches only on the tractable refinements of the edges whose labels are not part of the tractable subset. This significantly reduces the problem size for a typical problem from a qualitative calculus with large tractable subsets.

The only maximal tractable subset (ORD-Horn) of the Interval Algebra that contains all atomic relations has been identified by Nebel and Bürckert [52]. Nebel [51] further showed that the right combination of tractable subsets and heuristics can lead to very fast solutions.

While Nebel's solver has been the standard IA solver for the past ten years, recent improvements have been proposed [8] and implemented in the latest version of the GQR solver [18], which is currently the fastest constraint-based IA solver [74].
2.2.3.2 SAT-based approach

Propositional Boolean Satisfiability (SAT) was the first class of problems that had been shown NP-complete [9], and any problem in the class NP can be transformed into a problem in SAT. The SAT problem is about determining whether a given set of clauses is satisfiable, that is, whether the propositional variables can be given truth-values so that all clauses are true. Over the last ten years, there has been tremendous progress in the area, and the state of the art SAT solvers today are successful in solving many problems encountered in industrial applications that were previously thought to be unsolvable in a reasonable time [31].

Recently, Pham et. al. [54] proposed an encoding of constraint networks of Interval Algebra (IA) into a propositional Boolean formula. The encoding can be based on either the constraints between the intervals as given in the network (event-based), or the constraints between the end-points of those intervals (point-based). In either case, the Boolean formulas are constructed in such a way that each solution of the formula corresponds to a path-consistent atomic refinement of the network, and vice versa. Hence the formula is satisfiable if and only if the network is consistent.

Pham et. al. studied several alternative encoding methods and found that empirically the “point-based 1D support” encoding performs best. The “1-D support” encoding works by allocating a Boolean variable, $x^v_{ij}$, for every atomic relation, $v$, on the edge between two intervals $(i, j)$. The variable is true if and only if the corresponding atomic relation holds between $i$ and $j$ in the final solution of a path-consistent atomic network. Two sets of at-least-one (ALO) and at-most-one (AMO) clauses are introduced to ensure that exactly one of these atomic relations holds in the final solution. This ensures the final network in the solution is an atomic network.

- ALO: $\bigvee_{v \in R_{ij}} x^v_{ij}$
- AMO: $\bigwedge_{u,v \in R_{ij}} \neg x^u_{ij} \lor \neg x^v_{ij}$

To ensure the final network is path consistent, a set of support (SUP) clauses is
Background

introduced to encode that every triangle \( (i, j, k) \) of the network (regarded as a complete graph) be closed under composition and intersection.

- **SUP:** \( \bigwedge_{u \in R_{ik}, v \in R_{kj}} \neg x_{ik}^u \lor \neg x_{kj}^v \lor x_{ij}^{u_1} \lor \cdots \lor x_{ij}^{w_m} \)

where \( \{w_1, \ldots, w_m\} = R_{ij} \cap (u \circ v) \).

In the point-based encoding, one includes the same set of clauses as above, but for the point algebra (PA) instead of IA, prescribing the relations between the endpoints of the intervals. The variable \( x_{i-j}^u \), for example, denotes the atomic relation \( u \) between the starting points of intervals \( i \) and \( j \) (substitute + for − for the ending points). To ensure soundness, this encoding requires an additional set of clauses to forbid spurious IA relations introduced by the intervals-to-points translation:

- \( \bigwedge_{r \in R_{ij}} \neg x_{i-j}^r \lor \neg x_{i+j}^r \lor \neg x_{i+j}^{r_1} \lor \neg x_{i+j}^{r_m} \)

In their empirical study of IA networks of up to 100 nodes, Pham et al. [54] have shown that the “point-based 1-D support” encoding used in conjunction with the MiniSat SAT solver [13] outperforms Nebel’s solver with its default settings on many hard instances.

### 2.3 Identifying Tractable Subsets

Given that algebraic-closure decides consistency for atomic networks, the next step is to investigate whether there exist any tractable subsets where a-closure or another polynomial algorithm is sufficient. This is important for a qualitative calculus because the constraint-based methods in QSTR as described previously benefit significantly from using these large, preferably maximal tractable subsets. They reduce the branching factor at each step of the search, and empirically it had been shown that these large tractable subclasses speed up the search for consistency of binary constraint networks in qualitative spatial and temporal reasoning [51; 65]. To this date, the identification of maximal
tractable subsets for the calculi is considered one of the major significant developments in the quest for efficient reasoning in QSTR.

In the beginning, detecting these tractable subclasses required long and arduous manual proof. For Interval Algebra, Van Beek and Cohen in 1990 first shown a-closure is sufficient for deciding consistency for networks with relations in the Continuous Endpoint Class $C$ [70]. Following this work, Nebel and Bürckert in 1995 identified the maximal tractable subclass of ORD-Horn, which correspond to 868 relations [52]. For the spatial calculus RCC8, three different maximal tractable subsets were identified by Renz in 1999 [58].

An important step in developing automated methods for detecting tractable subclasses of a qualitative calculus was the refinement algorithm, which serves as a heuristic for proving tractability of given subsets of relations. The refinement algorithm was used in detecting the three maximal tractable subclasses $H8$, $C8$ and $Q8$ of RCC8. Renz in 2007 extended the algorithm and presented a procedure that would identify large tractable subsets of a qualitative calculus without any manual intervention [59]. The procedure require only that path-consistency entails consistency for networks of base relations.
Background
Computational complexity is the analytical study of the inherent complexity of computational problems. Identifying the complexity of qualitative spatial or temporal calculi has been an important research topic in Qualitative Spatial and Temporal Reasoning (QSTR). Most interesting calculi have been shown to be at least NP-hard, but if large tractable fragments of the calculi can be found, then efficient reasoning with these calculi is possible. In order to get the most efficient reasoning algorithms, we are interested in identifying maximal tractable fragments of a calculus, which are tractable fragments such that any extension of the fragment leads to NP-hardness.

In the previous Chapter we have introduced existing literature in automated methods for identifying large tractable subsets introduced by Renz in 2007 [59]. The work in this Chapter intends to complement that work, where we propose an automated method to identify intractable subsets. That is, we identify subsets of relations of a qualitative calculus where reasoning is NP-hard. We found that small, but interesting constraint networks are the key for identifying these intractable subsets, as they can be used as building blocks in constructing a transformation of reasoning problems in the qualitative calculus to 3SAT problems, a known class of NP-complete problems. We introduce an automated procedure to look for these interesting networks, and these networks are used for generating NP-hardness proofs. Our automated approach can be applied to arbitrary spatial or temporal calculi. By combining our approach and the automated method pro-
posed by Renz [59] on detecting tractable subsets, it is now possible to automatically identify maximal tractable fragments of qualitative spatial and temporal calculi.

### 3.1 Computational complexity

As introduced in Chapter 2, QSTR uses a qualitative calculus to model relations between entities in space or time. A qualitative spatial or temporal calculus such as RCC8 [57] or the Interval Algebra [1] is usually based on a given domain, $\mathcal{D}$, of spatial or temporal entities, such as a set of three-dimensional regions or time intervals, and a set, $B$, of jointly exhaustive and pairwise disjoint (JEPD) relations, $R \subseteq \mathcal{D} \times \mathcal{D}$, called base relations or atomic relations. Between any two entities of $\mathcal{D}$ exactly one of the base relations holds. If the exact relation is not known, we can also use the union of different possible base relations, i.e., the set of all relations we can use is the power set $2^B$ of the set of base relations.

Given a spatial or temporal domain $\mathcal{D}$ and a set of base relations $B$ over the domain, there are different reasoning problems we can study. Since most of these problems can be reduced to the consistency problem CSPSAT($\mathcal{S}$) [64], it is commonly used as the main reasoning problem for qualitative spatial and temporal information: *Given a set variables $\mathcal{V}$ over $\mathcal{D}$ and a set $\Theta$ of constraints $xRy$ with $x, y \in \mathcal{V}$ and $R \in \mathcal{S} \subseteq 2^B$, we ask whether $\Theta$ is consistent, i.e., can we find an instantiation of every variable in $\mathcal{V}$ with a value from $\mathcal{D}$ such that all constraints in $\Theta$ are satisfied.*

The consistency problem is a constraint satisfaction problem over infinite domains which is NP-hard for most calculi if all relations $2^B$ are allowed. However, it is sometimes possible to find subsets of $2^B$ for which reasoning becomes tractable. It has been shown that if large tractable subsets can be found which contain all base relations, then it is possible to find extremely efficient solutions to the reasoning problems even in the general case when all relations are permitted [51; 65]. Since efficient reasoning is of utmost importance, it has been a major challenge in qualitative spatial and temporal reasoning.
research to find large tractable subsets of different calculi. Ideally, we are interested in finding maximal tractable subsets, i.e., tractable subsets for which every extension is NP-hard [52]. These subsets mark the boundary between tractable and NP-hard fragments of a calculus and lead to the most efficient reasoning algorithms.

3.1.1 Existing methods for proving complexity

In order to identify maximal tractable subsets, we have to first find tractable subsets and then show that every extension of them is NP-hard. While in earlier research both of these tasks had always been done manually, Renz [59] recently developed a procedure which automatically computes large tractable subsets of a spatial or temporal calculus given only the relations $B$ and their composition table. Provided that path-consistency decides $\text{CSPSAT}(B)$, the output sets of Renz’s procedure are tractable subsets of $2^B$.

In addition to identifying these large tractable subsets, we would like to know whether they are maximal. That is, if we include any other relation to the tractable subset, reasoning with this new set of relations is NP-hard. In order to prove that a tractable subset is a maximal tractable subset, we also need some NP-hardness proofs for showing that any extension of a tractable subset is NP-hard. A very helpful result for proving complexity of subsets states that the closure of a set $S$ under composition, union, intersection and converse has the same complexity as $S$ [64]. This allows us to derive that if a set is tractable, its closure is also tractable, and similarly, if a set is NP-hard, every set that contains the NP-hard set in its closure is also NP-hard. It considerably reduces the number of actual NP-hardness proofs we need. So far there is no automated method for finding NP-hardness proofs and these proofs are usually done manually by using computer assisted enumeration methods that exploit the closure of sets.

In the case where weak-composition ($\circ$) do not equal composition ($\odot$), tractability of a relation can still be transferred to its closure if we show that the calculus is one-shot extensible [37]. That is, for any consistent network $N$ of the calculus, take any two variables $x, y \in V_N$ and $\ell_N(x, y) = R$, suppose $R \in S \odot T$, then we can add a
fresh variable $z$ to $V_N$ and two constraints $\ell_N(x, z) = S$, $\ell_N(z, y) = T$ to $\ell_N$ without changing the consistency of the network. In other words, any consistent network can be extended by a triangle, and the resulting network is still consistent. This result allows Renz’s automated procedure for detecting tractable subsets to be applied with a weak-composition table if the calculus has the properties a-closure decides consistency and one-shot extensibility.

Renz and Nebel [58] proved NP-hardness of sets of RCC8 relations $S$ by reducing variants of the NP-hard 3-SAT problem to $\text{CSPSAT}(S)$. In the next section we will generalize the reduction schema proposed by Renz and Nebel and develop an efficient procedure which automatically generates reductions according to this schema. In the following we will summarize the transformation schema.

A propositional formula $\phi$ in 3-CNF can be reduced to a corresponding set of constraints $\Phi$ over the relations $S \subseteq 2^S$ in the following way. Every variable $v$ of $\phi$ is transformed to two literal 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 \(R_t \cup R_f \in S\) 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 \textit{vice versa}. This is ensured by additional \textit{polarity constraints} \(x_v \{P_1\} x_{\neg v}, y_v \{P_2\} y_{\neg v}, x_v \{P_3\} y_{\neg v},\) and \(y_v \{P_4\} x_{\neg v}\) (see Fig. 3.1.a). In addition, every literal occurrence \(l\) of \(\phi\) 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, we need the same polarity constraints \(P_1, P_2, P_3, P_4\) 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 \(v\) (see Fig. 3.1.b).

Further, \textit{clause constraints} have to be added to assure that the clause requirements of the specific propositional satisfiability problem\(^1\) are satisfied. For example, if \(\{i, j, k\}\) is a clause of an instance of \textsc{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. The clause constraints are all constraints between \(x_i, x_j, x_k, y_i, y_j,\) and \(y_k\) that ensure this behaviour (see Fig. 3.1.c).

If relations \(R_t, R_f\), polarity constraints and clause constraints can be found for a set \(S\), then this transformation schema gives us a polynomial transformation from \(\phi\) to \(\Phi\). In order to show that \(\text{CSP} (S)\) is \(NP\)-hard we have to show that the propositional formula \(\phi\) is satisfiable \textit{if and only if} the resulting set of constraints \(\Phi\) is consistent, i.e., the transformation schema must be a \textit{many-one reduction}. With the specified transformation schema it is clear that whenever the constraints are consistent, then the \textsc{3Sat} formula is satisfiable. For the other direction (if \(\phi\) satisfiable then \(\Phi\) consistent), Renz and Nebel (\cite{RenzNe07}, proofs of Theorem 3 and Lemma 10) had to manually construct a consistent \textsc{Rcc8} realization schema that depends on the \textsc{3Sat} formula \(\phi\).

\(^1\)Renz and Nebel (\cite{RenzNe07}) used three different propositional satisfiability problems for their transformation schema: \textsc{3Sat}, \textsc{One-in-Three-3Sat} and \textsc{Not-All-Equal-3Sat} \cite{RenzNe07}
3.1.2 Novelty of our approach

Using the above described polynomial transformation schema, it is possible to generate the polarity and the clause constraints automatically and therefore it seems possible to generate NP-hardness proofs automatically as well. However, there are two reasons why this method cannot immediately be used to automatically find NP-hard subsets.

The main problem is that the transformation schema gives an NP-hardness proof only if we can prove that it is a many-one reduction. Previously, this required the construction of a consistent spatial or temporal realization which is not likely to be automated as it has to refer to the infinite domain and the semantics of the relations. Indeed, it has always been believed that this is impossible to achieve and that, therefore, NP-hardness proofs for qualitative calculi require a creative step that can only be made by a human expert.

The second problem is the required runtime for finding suitable relations $R_t, R_f$ and the polarity and clause constraints. Even for a small calculus like RCC8 with 256 relations, there are $255^6$ different networks that might have to be checked for finding polarity constraints and $255^{12}$ different networks for clause constraints. If we check one million networks per second, this requires a runtime of almost nine years just for the polarity constraints.

In the following sections we show how we can solve these two problems and present a highly efficient procedure for automatically generating NP-hardness proofs based on the given transformation schema, and prove its correctness. We solve the first problem by developing a novel proof method for showing that the transformation schema gives a many-one reduction. Our proof method requires additional conditions for polarity and clause constraints, which distinguish valid from invalid polarity and clause constraints. The experiments in Section 4 show that these extra conditions are essential for the correctness of our procedure. Our proof method does not require us to identify consistent realizations, it can be verified automatically and guarantees correctness of our procedure. The second problem is solved by exploiting several general properties of qualitative calculi and the transformation schema.
3.2 A General Procedure for Proving NP-hardness

In this section we present a procedure for automatically identifying intractable subsets. As mentioned in the introduction, these subsets mark the upper bound for tractability. This procedure automatizes the transformation of a 3SAT instance $\phi$ (and its variants NOT-ALL-EQUAL-3SAT and ONE-IN-THREE-3SAT) into an instance $\Phi$ of CSPSAT by using the polarity and clause constraints method as described in the previous section. In this way we can systematically identify intractable subsets of a given set of relations $2^B$. Similar to [59] we assume that a-closure decides consistency for CSPSAT($B$).

Our procedure consists of three steps which we will describe separately. We start with an input set $\mathcal{I} \subseteq 2^B$ that is closed under weak-composition, union, intersection and converse, and for which a-closure decides CSPSAT($\mathcal{I}$). As the first step, we identify all relations $R \in 2^B$ with $R \not\in \mathcal{I}$ for which we can find literal and polarity constraints. With these, variables and literal occurrences of $\phi$ can be transformed to constraints of $\Phi$. The second step is to find clause constraints for each relation for which we found polarity constraints. For both steps we use several optimizations which considerably reduce the runtime of our procedure. The third step is then to identify those relations $R' \in 2^B$ for which the closure of $\{R'\} \cup \mathcal{I}$ contains any of the already identified relations $R$. Each of these relations $R'$ leads to NP-hardness when added to $\mathcal{I}$.

During this procedure we will test several conditions that the polarity and the clause constraints have to satisfy in order to guarantee a many-one reduction of $\phi$ to $\Phi$. After sketching the details of our procedure, we prove that the procedure correctly identifies intractable subsets whenever the polarity and clause constraints satisfy all conditions. The correctness proof is based on inductive proofs that show how a set of constraints as generated by the transformation schema can become inconsistent and which changes preserve consistency. It exploits the fact that there are only finitely many possible triples of relations that occur in the transformation of $\phi$ to $\Phi$ and that once all possible triples are tested for consistency, no new inconsistencies can be introduced by adding more
triples. The reason why testing triples is sufficient is that we use the algebraic-closure algorithm for consistency checking, which only considers triples of relations.

In order to test all possible consistent triples, we will add particular redundant clauses to \( \phi \), i.e., 3CNF clauses that do not change satisfiability of \( \phi \) because they contain at least one literal of \( \phi \) that is assigned true. Using the redundant clauses, we can ensure that all possible triples have been tested. It is clear that when transforming \( \phi \) to \( \Phi \), these redundant clauses shouldn’t change consistency of \( \Phi \). If they do make \( \Phi \) inconsistent, then we know that the corresponding clause constraints cannot lead to a many-one reduction. Properties of redundant clauses are specified in the following proposition.

**Proposition 1.** If the transformation schema gives a many-one reduction of a 3SAT formula \( \phi \) to a set of constraints \( \Phi \), then the following properties must hold:

1. If \( \phi \) is satisfiable, then for any satisfiable 3SAT formula \( \phi' \) over the same variables as \( \phi \) with \( \phi \subseteq \phi' \), the corresponding set of constraints \( \Phi' \) is a refinement of \( \Phi \).

2. Adding a redundant clause to \( \phi \) does not change consistency of the resulting set of constraints \( \Phi' \).

3. For any variable \( x \in \phi \), we can arbitrarily add many redundant clauses containing one of the literals \( x \) or \( \neg x \).

4. For any pair of variables \( x, y \in \phi \), we can arbitrarily add many redundant clauses containing one of the literals \( x \) or \( \neg x \) and one of the literals \( y \) or \( \neg y \).

While it is clear that redundant clauses can be added to a 3SAT formula \( \phi \) without changing satisfiability, it is not clear that this will also hold for the resulting set of constraints \( \Phi \). By proving that it does hold for \( \Phi \) when using the transformation schema, we can show that the transformation schema gives a many-one reduction.
3.2.1 Step 1: Finding polarity constraints

Polarity constraints are the core of the transformation and ensure proper assignments of literal constraints. Intuitively, they are small constraint networks which can be used to model literals in a SAT problem.

Definition 2 (Polarity constraints). Given a set of relations $2^B$, four variables $x_v, x_{-v}, y_v, y_{-v}$ and a relation $R_{ff} = \{R_t, R_f\} \in 2^B$, the constraints $x_v\{P_1\}x_{-v}, y_v\{P_2\}y_{-v}, x_v\{P_3\}y_{-v},$ and $y_v\{P_4\}x_{-v}$ with $P_1, P_2, P_3, P_4 \in 2^B$ (see Fig. 3.1a) are called polarity constraints of $R_{ff}$ (abbreviated as $(R_t, R_f; P_1, P_2, P_3, P_4)$) if they ensure opposite assignment of the literal constraints $x_v\{R_t, R_f\}y_v$ and $x_{-v}\{R_t, R_f\}y_{-v}$, i.e., if they satisfy the following five basic requirements:\footnote{The original instantiation consists of the to-be-tested polarity constraints and the given literal constraints. If we enforce $a$-closure, we have to obtain the specified $a$-closed refinement in order to get the required flip-flop behaviour.}

<table>
<thead>
<tr>
<th>Original Instantiation</th>
<th>A-closed Refinement</th>
</tr>
</thead>
<tbody>
<tr>
<td>$(x_v, y_v)$</td>
<td>$(x_{-v}, y_{-v})$</td>
</tr>
<tr>
<td>${R_t, R_f}$</td>
<td>${R_t, R_f}$</td>
</tr>
<tr>
<td>${R_t, R_f}$</td>
<td>${R_f}$</td>
</tr>
<tr>
<td>${R_t, R_f}$</td>
<td>${R_f}$</td>
</tr>
<tr>
<td>${R_f}$</td>
<td>${R_t, R_f}$</td>
</tr>
<tr>
<td>${R_f}$</td>
<td>${R_t}$</td>
</tr>
</tbody>
</table>

The task of the transformation schema is to prove NP-hardness of the set of relations that are used for the construction of polarity and clause constraints. In order to find NP-hard subsets, we could generate all possible polarity and clause constraints, but, since we are mainly interested in finding relations that make a given tractable subset NP hard, we can restrict the relations we use for the polarity and clause constraints. For our procedure we will therefore use a tractable set $T$ as input set and assume that $a$-closure decides consistency for $\text{CSPSAT} (T)$. This is the case if we use the large tractable subsets resulting from Renz’s procedure [59] as input sets to our procedure. For a given
input set $\mathcal{I}$, we want to test for all relations $R \in 2^\mathcal{I} \setminus \mathcal{I}$ whether CSPSAT(closure($\mathcal{I} \cup R$)) is NP-hard. Therefore, we have to test if we can find a relation $R_{tf}$ and polarity constraints for $R_{tf}$ which are all contained in closure($\mathcal{I} \cup R$).

In order to prove that the polarity constraints are valid, they have to satisfy two additional conditions. We need both of these conditions for proving that a satisfiable instance $\phi$ is transformed to a consistent set $\Phi$. The key for this proof is to show that whenever all variables of $\phi$ are consistently assigned and the corresponding relations $R_{tf}$ of $\Phi$ are each set to $R_t$ or $R_f$, then enforcing a-closure to $\Phi$ results in a set $\Phi'$ which consists only of relations of $\mathcal{I}$, for which we know that a-closure decides consistency.

**Condition 1** (Extra conditions for polarity constraints). Given a variable $v$ of $\phi$ and the corresponding set of polarity constraints $(R_t, R_f; P_1, P_2, P_3, P_4)$ over the variables $x_v, x_{\neg v}, y_v, y_{\neg v}$ as shown in Figure 3.1.a. The polarity constraints are valid, if they satisfy the following two conditions:

1. If we impose $x_v \{R_t\} y_v$ and $x_{\neg v} \{R_f\} y_{\neg v}$ or vice versa, and apply a-closure, then $P_1, P_2, P_3,$ and $P_4$ must be refined to relations of $\mathcal{I}$.

2. Given a positive literal occurrence $p_1$ of $v$ and a negative literal occurrence $n_1$ of $v$. We transform these to the corresponding constraints (as shown in Figure 3.1.b) and apply a-closure, resulting in the set of constraints $\Theta$. We construct a set $\Theta'$ by modifying $\Theta$ in the following way:

(a) We add two more positive $(p_2, p_3)$ and two more negative $(n_2, n_3)$ literal occurrences of $v$. We transform the literal occurrences to the corresponding literal and polarity constraints (see Figure 3.2.b) and add them to $\Theta$. This introduces the new variables $x_{p_i}, x_{n_i}, y_{p_i}, y_{n_i}$ for $i = 2, 3$.

(b) We add constraints to ensure that each new pair of variables is different from any other pair, i.e., $x_{p_i} \neq x_{p_j}$ or $y_{p_i} \neq y_{p_j}$, and $x_{n_i} \neq x_{n_j}$ or $y_{n_i} \neq y_{n_j}$, for each $i, j \in \{1, 2, 3\}$ with $i \neq j$. 
(c) We apply a-closure.

Then $\Theta$ must be a proper subset of the resulting set $\Theta'$, i.e., no constraint in $\Theta$ changes by adding more literal occurrences.

Both conditions can be easily tested by constructing the required sets of constraints and applying a-closure to them. The first condition ensures that a-closure will be sufficient for determining consistency. We will use the second condition for proving that an arbitrary number of redundant clauses can be added to a variable (see Proposition 1.3). This is proven in the following Lemma.

**Lemma 1.** Given a 3SAT formula $\phi$, a set of relations $2^B$ and polarity constraints $P_1, P_2, P_3, P_4$ for a relation $R_{if} \in 2^B$. If the polarity constraints satisfy Condition 1.2, then we can add an arbitrary number of literal occurrences of $v$, transform them to the corresponding constraints and enforce a-closure without changing the constraints of the existing literal occurrences.
Proof. Assume we can add \( n \) literal occurrences and enforcing a-closure does not change any existing constraints. We now show that we can add \( n + 1 \) literal occurrences. We assume wlog. that we add a positive literal \( p \) which results in adding the corresponding constraints \( x_p \{ R_{tf} \} y_p \), the polarity constraints connecting \( x_p, y_p \) with \( x_{=w}, y_{=w} \), and the constraints that \( x_p \) and \( y_p \) are not both identical to any existing variable. The constraints between \( x_p, y_p \) and any other variable \( u \) are exactly the same as between any other existing positive literal occurrence \( q \) when it was added and \( u \). Since we have already shown as a precondition that it is possible to add three literal occurrences, we have already computed every possible triple of variables that can occur when adding \( x_p, y_p \) and therefore the result of enforcing a-closure is the same as with any existing triple. Since none of the previous literal occurrences changed any existing constraints, the new one will not change anything either. The lemma follows by induction. \( \square \)

3.2.2 Improvements in finding polarity constraints

The following modifications can considerably reduce the runtime for computing polarity constraints.

Symmetry of Polarity Constraints: For any valid polarity constraints for \( R_{tf} \), we can modify the polarity constraints and apply them to other relations without having to compute them again. So if \( \{ R_t, R_f; P_1, P_2, P_3, P_4 \} \) is a valid assignment of the six constraints, then the following five assignments are also valid via symmetry:

- \( \{ R_f, R_t; P_1, P_2, P_3, P_4 \} \),
- \( \{ R_t^-, R_f^-; P_2, P_1, P_4, P_3 \} \),
- \( \{ R_f^-, R_t^-; P_2, P_1, P_4, P_3 \} \),
- \( \{ R_t, R_f; P_2^-, P_1^-, P_3^-, P_4^- \} \),
- \( \{ R_f, R_t; P_2^-, P_1^-, P_3^-, P_4^- \} \).
Improving A-Closure: The a-closure algorithm always looks at triples of variables. For a given set of relations, the possible number of different triples is limited. Instead of computing all triples again and again, we can precompute all possible triples and store those that are a-closed in a table. Then we only need to look up relations in a table to decide whether a particular triple is a-closed, which is much faster than making it a-closed. In addition, we are hard-coding a-closure for networks with four nodes instead of using a queue, which further optimizes this frequently used operation.

Algebraic-closure-based selection: In order to identify polarity constraints for \( R_{1f} \) we normally have to loop through all possible instantiations of the four relations \( P_1, P_2, P_3, P_4 \). However, we are only looking for polarity constraints which are already a-closed. Therefore, a more efficient method for a given \( x_v \{ R_t, R_f \} y_v \), is to first select any \( P_1 \) such that \( x_v \{ P_1 \} \bar{x}_v \). Then the \( P_4 \) for \( y_v \{ P_4 \} \bar{x}_v \) can only be selected from those relations that make \( \{ x_v, \bar{x}_v, y_v \} \) a a-closed triple. Since we have already precomputed all a-closed triples, we can easily look up which relations we can choose for \( P_4 \) and likewise for \( P_2 \) and \( P_3 \). There are four triples in a well-connected graph with four nodes, therefore if the probability of a triple being a-closed is \( m \), we only have to perform a factor \( m^4 \) of the original computation.

3.2.3 Step 2: Finding clause constraints

Once we have identified valid polarity constraints, we then have to find clause constraints in order to transform all clauses of an instance of one of the three 3-SAT variants into CSPSAT. Each literal occurrence \( l_{ij} \) of a clause \( \{ l_{i1}, l_{i2}, l_{i3} \} \) is transformed into a literal constraint \( x_{ij} \{ R_t, R_f \} y_{ij} \). The clause constraints have to ensure that at least one of the three literal constraints will be \( R_t \) for 3SAT, exactly one of them will be \( R_t \) for ONE-IN-THREE-3SAT and one or two of them will be \( R_t \) for NOT-ALL-EQUAL-3SAT. So each clause is transformed into a set of constraints with six variables. The clause constraints consist of the twelve remaining constraints between these six variables (see Fig. 3.1c). It is impossible to systematically enumerate all possibilities for
these twelve constraints as there would be far too many combinations. We can, however
reduce the twelve constraints to only four different ones: The literals of a clause can be
permuted without changing the truth value of the clause. Therefore, we must also be
able to permute the clause constraints without changing consistency as it does not make
any difference which of the three literal constraints are assigned \(R_t\). This has the conse-
quence that whenever it is possible to find clause constraints, it is possible to find clause
constraints such that the four clause constraints between any two literal constraints are
the same. We need four clause constraints of type \(x_{ij}C_1x_{ik}, y_{ij}C_2y_{ik}, x_{ij}C_3y_{ik}, y_{ij}C_4x_{ik}\)
(for \(j, k = (1, 2), (2, 3), \) or \((3, 1)\)). Similar to the polarity constraints, we abbreviate
the clause constraints as \((R_t, R_f; C_1, C_2, C_3, C_4)\). Enumerating all possibilities for four
different constraints is possible, especially if we use the same optimizations as described
for polarity constraints. The basic property of clause constraints is that they have to be
a-closed and, if some of the literal constraints are refined to either \(R_t\) or \(R_f\), a-closure
must enforce one of the possible behaviours of the three 3SAT variants to the other literal
constraints. As we did for polarity constraints, we also require valid clause constraints
for an input set \(\mathcal{I}\) to satisfy additional conditions which will be used for proving correct-
ness of our procedure. We provide the conditions for 3SAT, and the conditions for the
other two 3SAT variants can be similarly derived.

**Condition 2 (Extra conditions for clause constraints).** Given a set of clause constraints
\((R_t, R_f; C_1, C_2, C_3, C_4)\). The clause constraints are valid, if they satisfy the following
three conditions:

1. Given three variables \(u, v, w\) of \(\phi\) and the corresponding set of clause constraints
\((R_t, R_f; C_1, C_2, C_3, C_4)\) over the variables \(x_u, y_u, x_v, y_v, x_w, y_w\) as shown in Fig-
ure 3.1.c. If we set each of the three literal constraints \((x_i, y_i)\) for \(i \in \{u, v, w\}\) to
either true or false, then a-closure refines all clause constraints to relations of \(\mathcal{I}\).
We test this for all eight possible instantiations of \(u, v, w\).

2. Given three instantiated variables \(u, v, w\) of a satisfiable 3SAT instance \(\phi\), i.e.,
each of $u, v, w$ is either true or false. There are eight different clauses ($u \lor \neg u, v \lor \neg v, w \lor \neg w$), seven will be true (i.e., they are redundant clauses) and one of them will be false—which depends on the instantiation of $u, v, w$. We select the seven redundant clauses and also the seven redundant clauses in reverse order ($w, v, u$). We transform these 14 redundant clauses and the variables $u, v, w$ to the corresponding constraints and apply $a$-closure, resulting in the set of constraints $\Theta$. Then $\Theta$ contains only constraints over relations of $I$.

3. We construct a set $\Theta'$ by modifying $\Theta$ in the following way: We triple the 14 redundant clauses from the previous condition, resulting in 42 redundant clauses. We transform them to the corresponding constraints, add the constraints to $\Theta$ and apply $a$-closure. Then $\Theta$ must be a proper subset of the resulting set $\Theta'$, i.e., no constraint in $\Theta$ changes by adding more redundant clauses.

Even though the previous conditions sound complicated, they can be quickly tested with one application of the $a$-closure algorithm and a simple comparison of constraints. We use the second and third condition to prove that an arbitrary number of redundant clauses can be added to a pair of variables (see Proposition 1.4). Note that the last two conditions do not restrict any possible polarity or clause constraints as these conditions have to be satisfied by any many-one reduction. Only polarity and clause constraints that do not lead to a many-one reduction will be eliminated.

**Lemma 2.** Given a set of relations $2^{\phi}$, valid polarity constraints $P_1, P_2, P_3, P_4$ for a relation $R_{if}$. clause constraints $C_1, C_2, C_3, C_4$, three instantiated variables $u, v, w \in \phi$ and a corresponding set of constraints $\Theta$ as specified in Condition 2.2.

If the clause constraints satisfy Condition 2.2/3, then we can add to $u$ and $v$ any number of redundant clauses containing one of the literals $u$ or $\neg u$ and one of the literals $v$ or $\neg v$, transform them to the corresponding constraints, add them to $\Theta$ and enforce $a$-closure without changing the existing constraints of $\Theta$.

**Proof.** We first prove the case where all redundant clauses have either $w$ or $\neg w$ as a third
Figure 3.3: Extra conditions for clause constraints as specified in Condition 2. The figure highlights one of the 14 redundant clauses for three variables $u, v, w$ and shows literal constraints introduced by other redundant clauses.
literal. Similar to Lemma 1, we prove this by induction over the number of redundant clauses. The induction base holds because Condition 2.2 holds. When adding three new literal occurrences for a new redundant clause and the corresponding clause constraints, there is at least one existing redundant clause for which we added exactly the same constraints. We have already computed every possible triple of variables that can occur when adding these constraints and therefore the result of enforcing a-closure is the same as with the existing redundant clause. Since adding the previous redundant clause did not change any existing constraints (this is guaranteed by Condition 2), the new one will not change anything either.

Now we prove the case where the third literal of a redundant clause can be any other variable \( w_i \). We assume that all redundant clauses have one of \( w_i \) or \( \neg w_i \) as a third literal, where \( i \in \{1, 2, 3, \ldots\} \) and each \( w_i \) is instantiated. We first group the variables \( w_i \) into those that are instantiated as true and those instantiated as false. All \( w_i \) that are instantiated as true can have the same \( 2 \times 7 \) redundant clauses as described in Condition 2, and likewise for for all \( w_i \) that are instantiated as false. Since there are no other constraints between the different \( w_i \) other than those given by the clause constraints of the redundant clauses, we can assume that all true \( w_i \) are equal and also all false \( w_i \) are equal, we call them \( w_t \) and \( w_f \), respectively (assumption 1). If we replace all \( w_f \) in the redundant clauses with \( \neg w'_f \) and all \( \neg w_f \) with \( w'_f \), then both \( w_t \) and \( w'_t \) can have exactly the same \( 2 \times 7 \) redundant clauses, and therefore we can assume that \( w_t \) and \( w'_t \) are equal (assumption 2). It follows from the above proven case that adding any number of redundant clauses over three variables \( u, v, w \) cannot change the existing constraints of \( \Theta \). Since both, assumption 1 and assumption 2 make the additional constraints on \( \Theta \) more restrictive, it follows that the redundant clauses that use the different variables \( w_i \) cannot change the existing constraints \( \Theta \) either.

With Lemma 1 and Lemma 2 we have shown that the properties of redundant clauses for satisfiable 3SAT formulas \( \phi \) as specified in Proposition 1 also hold when transforming \( \phi \) to a set of constraints \( \Phi \) using the given transformation schema.
Algorithm: NPHARD-EXTENSIONS(\mathcal{I}, \mathcal{E})

Input: A tractable subset \mathcal{I} and possible extensions \mathcal{E}

Output: A set \mathcal{O} \subseteq \mathcal{E} of relations each of which makes \mathcal{I} NP-hard

1. \mathcal{O} = \emptyset;
2. For all \mathcal{R} \in \mathcal{E} do
3. \quad \mathcal{C} = \text{closure}(\mathcal{I} \cup \{\mathcal{R}\}); \text{ loop} = \text{true};
4. \quad \text{If } \mathcal{C} \cap \mathcal{O} \neq \emptyset \text{ then } \mathcal{O} = \mathcal{O} \cup \{\mathcal{R}\}; \text{ continue;}
5. \quad \text{while (loop == true) do}
6. \quad \quad \text{Find a } \mathcal{R}_{tf} \text{ and new pol. constraints for } \mathcal{R}_{tf} \text{ in } \mathcal{C}
7. \quad \quad \text{If none can be found then loop = false; continue;}
8. \quad \quad \text{If pol. constraints do not satisfy Cond. 1 continue;}
9. \quad \quad \text{Find new clause constraints in } \mathcal{C} \text{ for the } \mathcal{R}_{tf} \text{ and the pol. constraints which satisfy Condition 2;}
10. \quad \quad \text{If found, then } \mathcal{O} = \mathcal{O} \cup \{\mathcal{R}\}; \text{ loop} = \text{false;}
11. \quad \quad \text{end while}
12. \quad \text{end for}
13. \text{return } \mathcal{O}

Figure 3.4: Procedure for finding NP-hard extensions of \mathcal{I}

3.2.4 Step 3: Applying closure

In the previous two steps we identified relations \mathcal{R} for which we can find a literal constraint, and polarity and clause constraints when adding them to a tractable set \mathcal{I} \subset 2^\mathcal{B}.

In the final step of our procedure, we compute for which relations \mathcal{R}' \in 2^\mathcal{B} the closure of \mathcal{I} with \mathcal{R}' contains a relation \mathcal{R}. Adding \mathcal{R}' to \mathcal{I} gives the same complexity as adding \mathcal{R}. It is possible to interleave this step with the previous steps and, once we have found polarity and clause constraints for one relation \mathcal{R} immediately transfer the result to all relations \mathcal{R}' that contain \mathcal{R} in their closure. We can speed up our procedure by selecting the order in which we process the relations according to the size of their closure, relations with smaller closure first. Another improvement is to first test whether a relation \mathcal{R} can be shown to be NP-hard when adding it to the base relations \mathcal{B}. This reduces the search space for finding polarity and clause constraints and implies NP-hardness of \mathcal{I} \cup \{\mathcal{R}\} if successful. If unsuccessful, we test \mathcal{I} \cup \{\mathcal{R}\}.

We can now prove that every set of literal constraints, polarity constraints and clause
constraints that satisfies our conditions gives us a many-one reduction of 3SAT to CSP-SAT. The procedure is sketched in Figure 3.4. Note that not all mentioned improvements are included in the sketch. Depending on which clause constraints we use, we can use the same procedure to also reduce ONE-IN-THREE-3SAT or NOT-ALL-EQUAL-3SAT to CSP-SAT.

Theorem 1. Given a set of relations $2^E$ and a tractable subset $\mathcal{I} \subset 2^E$ for which $a$-closure decides consistency as input to our procedure. For every relation $H$ of the output set $O$ of our procedure, $\text{CSPSAT}(\{H\} \cup \mathcal{I})$ is NP-hard.

Proof. We have to show that for every $H \in O$, our procedure finds a many-one reduction of a 3SAT variant to $\text{CSPSAT}(\{H\} \cup \mathcal{I})$. It is clear from the transformation schema that whenever a 3SAT instance $\phi$ is unsatisfiable, then the corresponding set of CSPSAT constraints $\Phi$ is inconsistent. We now show that whenever $\phi$ is satisfiable, $\Phi$ will be consistent. We assume that we have a satisfiable instantiation of all variables of $\phi$. Since $\phi$ is consistent, we know that all clauses of $\phi$ are redundant clauses. We now show that a refinement of $\Phi$ is already consistent:

Given a satisfiable 3SAT formula $\phi(n)$ with $n$ variables that contains all possible redundant clauses. The transformation schema transforms $\phi(n)$ to $\Phi(n)$, applying $a$-closure leads to $\Phi'(n)$. We prove by induction over the number of variables $n$ that $\Phi(n)$ is consistent for any $n \geq 3$. The induction base with $n = 3$ holds, because the transformation schema satisfies Condition 2.2. Therefore, $\Phi'(3)$ is an $a$-closed set that contains only relations of $\mathcal{I}$, and hence $\Phi'(3)$ is consistent. We assume that $\Phi(n)$ is consistent and prove consistency of $\Phi(n+1)$. We successively transform the redundant clauses for the new variable $v_{n+1}$ to the corresponding constraints. Each new redundant clause contains two literals over variables in $\phi(n)$. By Lemma 2 we know that these new constraints do not change any existing constraints of $\Phi'(n)$. Therefore, the only possibility how $\Phi(n+1)$ can become inconsistent is via the newly introduced constraints. However, the $a$-closure algorithm only ever looks at triples of relations and any triple can belong to constraints corresponding to at most three different variables.
Because of Condition 2, we have already tested all possible triples for three variables \(v_i, v_j, v_k\) and none of them can lead to an inconsistency. The additional literal constraints also do not affect consistency, as shown in Lemma 1. Since all redundant clauses are the same for all triples of variables, it makes no difference if \(v_{n+1}\) is one of the three variables. Therefore, \(\Phi(n + 1)\) must be consistent too, and by induction it follows that \(\Phi(n)\) is consistent for all \(n\).

We know that \(\phi \subseteq \phi(n)\) and therefore \(\Phi(n)\) must be a refinement of \(\Phi\). Since \(\Phi(n)\) is consistent, \(\Phi\) must be consistent too. \(\square\)

### 3.3 Empirical Evaluations of the Procedure

Our procedure can be applied to any binary spatial or temporal calculus, and correctness of our procedure is guaranteed if \(a\)-closure decides consistency for the input set. The only limitation of our procedure is the runtime it takes to compute polarity constraints for a given set of relations (computing clause constraints is much faster and the different conditions can be checked instantly). In the worst case, this is of the order \(O(n^6)\) where \(n\) is the cardinality of the largest closure we test. Using the optimizations we presented we can bring this down considerably, but there is clearly a limit in the size we can handle. We will further discuss the limitations in the following section.

We did all the tests on an Intel Core2Duo 2.4GHz processor with 2GB of RAM and used only the 3SAT version of our procedure. We initially applied our implementation to RCC8 and used the set of base relations as the input set, i.e., we tested NP-hard subsets as well as tractable subsets. Our procedure terminated in less than four hours and we identified all 76 known NP-hard relations. None of the polarity and clause constraints we found violated the additional conditions we require, i.e., polarity and clause constraints were only found for NP-hard relations. We then used the three known tractable subsets \([58]\) that were also identified automatically in \([59]\) as the input sets. We showed maximality of \(H_8\) in 12 minutes, \(Q_8\) in 17 minutes and \(C_8\) in 30 minutes. This means
that for RCC8 we can identify all maximal tractable subsets automatically, prove that they are tractable and now also that they are maximal.

The next calculus we tested was the Cardinal Direction Calculus [40]. It has nine base relations and 512 relations in total. Ligozat (|40|) identified one maximal tractable subset which consists of all relations with the pre-convexity property. We first ran our algorithm with the base relations as input set. After 21 hours it returned a set of NP-hard relations that was exactly the complement of the set of pre-convex relations. One interesting observation we made was that there were many relations for which we found polarity and clause constraints but for which the additional conditions were not satisfied. This shows that the conditions we identified in order to be able to prove correctness of our procedure are very important in practice. We ran our procedure again with the set of pre-convex relations as input and were able to prove maximality of the set in two minutes.

Our next test was the Interval Algebra. As for the previous calculi, we first attempted to use the set of base relations as input set. Again, we found many polarity and clause constraints for tractable relations, but none of them satisfied the additional conditions. While some relations were solved quickly, we stopped our procedure as for some relations the computation took too long. Then, we used ORD-Horn as input set [52], the only maximal tractable subset of the Interval Algebra that contains all base relations. It was also identified as tractable by [59]. Our procedure found that there are only 27 distinct closures when adding relations to ORD-Horn, but some of them are also very large. Fortunately, it turned out that two of the 27 closures are contained in all the remaining 25 closures. Our procedure, therefore, had to test only these two relations: (d,di) has 106 relations in its closure with B, and (o,oi) has 162 relations. Our procedure proved NP-hardness of the two sets (and consequently maximality of ORD-Horn) in 1.5 hours.
3.4 Limitations of our Procedure

We proved the correctness of our algorithm, i.e., whenever the algorithm identifies a set to be NP-hard, then it is indeed NP-hard, but we did not prove completeness, i.e., whenever a set is NP-hard, then our algorithm identifies it. Due to the definition of NP-hardness, there must be a polynomial transformation from 3SAT to CSPSAT($S$) whenever $S$ is NP-hard. The question is whether this transformation always coincides with our transformation schema. Our schema is very generic and does not have any restrictions regarding the set $S$ of relations it can be applied to. It individually transforms variables, literals, and clauses of a 3SAT formula to corresponding sets of constraints over $S$ in a very natural and intuitive way. The transformation is invariant with respect to the order of literals in clauses and scales to any number of clauses. These are good indications that the procedure might be complete. This is also supported by the tests we have done so far, where all NP-hard subsets were identified.

Apart from these indications, we cannot currently see a completeness proof for our procedure. Tractability, however, can already be shown by the sound procedure developed by Renz ([59]). It will be very interesting to analyze the interactions of these two procedures. Together, they might give us a sound and complete decision procedure for NP-hardness and tractability.

The main limitation of our procedure is its runtime. Even though our procedure can be applied to any binary calculus with a given weak-composition table, it is clear that for large calculi our procedure might not terminate in reasonable time. The worst-case time-complexity of our procedure is $O(n^6)$ where $n$ is the size of the largest subset we have to fully analyze. On today's computers, the limit of an $O(n^6)$ algorithm is reached if $n \approx 1000$. We assume that the practical limit for our procedure are calculi of a size not much larger than the Interval Algebra. This covers many existing and useful spatial and temporal calculi. Much larger calculi are unlikely to terminate in reasonable time on today's computers.
However, large calculi such as the rectangle algebra [4] with 169 base relations and approximately $10^{50}$ relations in total (exactly $2^{169}$ relations), are impossible to analyze computationally. For these calculi we cannot even enumerate all relations. The minimum requirement for analysing a calculus with computer assistance is to be able to compute all possible closures of the base relations with any other relation. If this is not possible, computational analysis can only be done manually. A simple approximation of whether a computational analysis is possible for a calculus is to compute and to store its full composition table. Given these limitations, the gap between calculi that can be analyzed computationally and calculi that can be solved by our procedure is very small.

3.5 Conclusion

We developed a procedure for automatically proving NP-hardness of subsets of qualitative spatial or temporal calculi. Our procedure produces correct proofs provided that a-closure decides consistency for the input set to our procedure. The results of our procedure mark the upper bound of tractability. When combining it with the lower bound of tractability, which can also be obtained automatically using the procedure presented by Renz ([59]), it becomes possible to identify maximal tractable subsets of a calculus completely automatically. We tested our procedure on different well-known calculi and reproduced all known NP-hardness results automatically in a very reasonable time. Future work will analyze situations where upper and lower bounds do not meet and to develop ways in which the two procedures can interact in order to bring the two bounds together. It would also be interesting to find algebraic conditions which guarantee that relations satisfy the additional conditions we require. From a more general perspective, our results demonstrate that automatic generation of complexity results is possible and we hope that our work inspires other researchers to analyze the applicability of similar methods to other problems.
Automated Complexity Proofs for Qualitative Calculi
This Chapter concerns combining constraint networks in Qualitative Spatial and Temporal Reasoning. It partly arose from two questions about work in the previous Chapter. First, in section 3.2, we look to combine different networks together over a common edge and ensuring no existing edge changes when enforcing path-consistency. Although we found a specialized solution for those special networks in lemma 1 and lemma 2 by imposing extra conditions on the networks, the question whether this can always hold in a qualitative calculus remains. Second, if weak-composition is not equal to composition for a qualitative calculus, we would require the calculus to be one-shot extensible in order to transfer tractability results of a relation to its closure [37]. One-shot extensibility can be seen as a condition ensuring combining a consistent network with a triangle would always result in a consistent network. It is an open question whether there is a generic proof for this property that can be applied to any qualitative calculus. This Chapter answers these questions by investigating a generalized proof-procedure ensuring two networks can always be combined together.

4.1 Combining networks

Consider two consistent sets of spatial or temporal information. It is clear that if the two refer to different entities, or independent aspects of spatial or temporal information,
then there are no potential conflicting constraints when the two sets of information are combined. However, if they contain information about the same entities, then it is clear that combining the two sets may lead to inconsistencies.

Here we are interested in a particular kind of combination of information, namely, combining spatial or temporal constraints that share only a very small number of entities. Furthermore, we are interested in the cases where there are no trivial inconsistencies. That is, the constraints between the shared entities are identical in both sets. Intuitively, the combined information should also be consistent. However, in QSTR there are several examples of qualitative calculi where this property is not satisfied and inconsistencies are introduced when combining two sets of information that share a small number of entities with identical relations [37]. This is generally due to the fact that there are certain constraints that are not explicitly stated, but which can be derived from the weak-composition table of the calculus. When two sets of information are combined together these implicit constraints could introduce inconsistencies between variables previously not associated with each other.

An example of this situation can be found for the INDU algebra [56], where the base-relations describe both the relation between endpoints of two intervals along a directed line and the duration of the intervals. Consider two consistent networks with three nodes as shown in Figure 4.1 (a). The first network states that interval 1 meets interval 2, and the length of 1 is shorter than 2, 1 starts with another interval 3, and 2 finishes with 3. The second network states that interval 1 meets another interval, 4, and the length of 1 is greater than 4, 1 starts with 3 and 4 finishes with 3. The constraints between the shared entities 1 and 3 are the same, and a consistent realization of each network is shown in Figure 4.1 (b). However, the combined constraint network involving all four intervals is not consistent, as the first network also enforces that the length of interval 1 is more than half of the length of interval 3, while the second network requires it to be less than half of the length of interval 3. Combining the two networks introduces conflict between these two implicit constraints.
Apart from the problem of introducing inconsistencies that are somewhat counter-intuitive, there are some important advantages for a qualitative calculus that always preserves consistency when combining two sets of information. First, it opens up the possibility to use divide-and-conquer techniques which split a large set of qualitative constraints into smaller sets that can be solved independently. This would be an essential requirement for any hierarchical reasoning, and it would have the potential to speed up reasoning. An extensive analysis of this can be found in Chapter 5. Secondly, it allows us to ignore or filter inconsequential information that is not important to the essence of a problem, as we know that introducing this information later would not affect the consistency of the solution.

Unfortunately, there is currently no general method for determining whether information can always be consistently combined for a qualitative calculus. Some initial results were obtained by Li and Wang [37], where they analyzed a special case of this problem called one-shot extensibility. This is essential for the analysis of computational complexity of reasoning with a qualitative calculus, as it is an essential requirement to transfer tractability of a relation to its closure under intersection and weak-composition. Li and Wang considered the case of consistently extending a consistent atomic set of
On Extending and Combining Constraint Networks

RCC8 constraints by one additional entity. This is the case where two consistent networks, one restricted to only three nodes, can always be consistently combined over an edge. Through extensive case analysis, they showed that this is always possible for the RCC8 calculus. Furthermore, Lutz and Milicic [42] gave proofs for "patchwork property" of the qualitative calculi RCC8 and Interval Algebra, which preserves consistency when combining information for these calculi. They showed that this is important for generalized T-Boxes in Description Logics. However, there is no current method to generalize the proof to be applied to other qualitative calculi in QSTR.

Finally, every qualitative calculus comes with its own algebra of relations, which is a relation algebra if the weak-composition operation is associative. This can be easily checked mechanically from the weak-composition table of the algebra. A more interesting, and more difficult property to verify, is whether the algebra is representable. A relation algebra is representable if it is isomorphic to some relation algebra consisting of binary relations on some set, and closed under the intended interpretation of the operations of the algebra. If this is the case where the set is the domain of the calculus, we know that weak-composition equals composition in the set-theoretic sense, and we say that the calculus is extensional. Unlike the case for relation algebras, the class of representable relation algebras cannot be finitely axiomatized. It remains an interesting mathematical question about whether a relation algebra is representable.

In this Chapter we show that through algebraic manipulations involving the weak-composition operator of the qualitative calculus, we can derive whether it is possible to combine two constraint networks that at most share two variables in common. Unlike previous methods through extensive case-analysis in the semantics of the calculus, our method is applicable to any qualitative calculus given its weak-composition table. It is important for three very different reasons. First, it provides a purely algebraic and generally applicable proof for one-shot extensibility; second, it solves some fundamental questions related to algebraic closure, consistency and weak-composition, and third, it provides a purely symbolic test for when a relation algebra is representable.
4.2 Formal Preliminaries

Recall from Chapter 2 that a qualitative calculus is a triple \((A, U, \mu)\) where \(A\) is an algebra of relations, formally known as a Nonassociative Algebra (NA); \(U\) is a set, considered to be the domain of the calculus; and \(\mu: A \rightarrow U\) is a weak representation of \(A\) over \(U\). For simplicity we frequently refer to the calculus as \(A\) if it is clear from the context. A calculus \((A, U, \mu)\) is **extensional** if \(\mu\) is a representation of \(A\), in which case the weak-composition operation equals set-theoretic composition.

In this Chapter we intensively use the precise mathematical definitions of relations, algebras and different algebraic operators which we summarize in the background Chapter. In particular, we will frequently refer to the triangle identities, which are applicable to all weak-composition operators:

**Lemma 3** (Triangle identities). For relations \(r, s, t\), the following always holds:

\[
r \cap s \circ t \neq \emptyset \iff s \cap r \circ t \neq \emptyset \iff t \cap s^\circ r \neq \emptyset
\]

A constraint network over \(A\) can be interpreted as a graph \((V, \ell)\), where the nodes are the variables in the CSP and the label \(\ell\) on the edge denotes the relation from one variable to another. A network is atomic if all the labels are base relations of \(A\), and algebraically-closed (a-closed) if the following conditions hold:

1. \(R_{ii}\) is the equality relation (identity element of \(A\))

2. \(R_{ij} \circ R_{jk} \supseteq R_{ik}\) for any \(i, j, k \in N\)

The closest approximation to consistency for a network is **proto-consistency**, where we removed any ambiguity represented by disjunctions, and all the constraint enforced by a-closure is satisfied. We say that a network is proto-consistent if it has an atomic a-closed.
The simplest network is a node itself, which is not very interesting. The most basic interesting network is a triangle, where the constraints of weak-composition enforce algebraic-closure.

**Definition 3 (A-closed triangles).** A network is called an a-closed triangle if the network has three nodes and that the network is a-closed. That is, for nodes \( \{0, 1, 2\} \), the following conditions hold:

1. \( R_{01} \subseteq R_{02} \circ R_{21} \)
2. \( R_{02} \subseteq R_{01} \circ R_{12} \)
3. \( R_{12} \subseteq R_{10} \circ R_{02} \)

If all the relations in the triangles are base relations, we will call the network an atomic, a-closed triangle.

**Lemma 4.** A network with nodes \( \{0, 1, 2\} \) is an atomic, a-closed triangle if and only if all its relations are base relations and \( R_{01} \cap R_{02} \circ R_{21} \neq \emptyset \).

**Proof.** (\( \rightarrow \)) From the triangle identities, we know that the following three equations are true if and only if one of them is true:

\[
R_{01} \cap R_{02} \circ R_{21} \neq \emptyset \\
R_{02} \cap R_{01} \circ R_{12} \neq \emptyset \\
R_{12} \cap R_{10} \circ R_{02} \neq \emptyset
\]

As \( R_{01}, R_{02} \) and \( R_{12} \) are base relations, we know that:

\[
R_{01} \in R_{02} \circ R_{21} \\
R_{02} \in R_{01} \circ R_{12} \\
R_{12} \in R_{10} \circ R_{02}
\]
which fulfils the requirement for the network to be an atomic, a-closed triangle.

\((\to)\) Suppose that \(R_{01} \cap R_{02} \circ R_{21} = \emptyset\), then \(R_{01} \not\subseteq R_{02} \circ R_{21}\). Hence the network is not a-closed.

Now we proceed to define amalgamation of a-closed networks. Let \(N_0, N_1, N_2\) be a-closed networks, such that \(N_0 = N_1 \cap N_2\). The triple \((N_0, N_1, N_2)\) is called a V-formation. A V-formation \((N_0, N_1, N_2)\) can be amalgamated if there is an a-closed network \(M\) such that \(N_1 \subseteq M\) and \(N_2 \subseteq M\). Such an \(M\) is called an amalgam of \(N_1\) and \(N_2\) over \(N_0\) or just an amalgam if the rest is clear from the context. Notice that we do not formally require \(V_M = V_{N_1} \cup V_{N_2}\). However, if an amalgam \(M\) exists, its restriction to \(M' \subseteq M\) with \(V_{M'} = V_{N_1} \cup V_{N_2}\) is an amalgam as well, so we can always assume that \(M\) only has nodes from \(N_1\) and \(N_2\).

**Definition 4** (Network Amalgamation Property). Let \(A\) be a qualitative calculus (NA). A has Network Amalgamation Property (NAP), if any V-formation \((N_0, N_1, N_2)\) of networks over \(A\) can be amalgamated by an a-closed network \(M\) over \(A\).

Clearly NAP is an elusive property for a qualitative calculus, and we do not yet know a single qualitative spatial or temporal calculus that has this property. Therefore, some restrictions are necessary. One such restriction calls for the common subnetwork \(N_0\) to be small in the following sense.

**Definition 5** \((k\text{-Amalgamation Property})\). Let \(A\) be a qualitative calculus (NA). \(A\) has \(k\) Amalgamation Property \((k\text{-AP})\), if any V-formation \((N_0, N_1, N_2)\) of networks over \(A\), such that \(|V_{N_0}| \leq k\), can be amalgamated by a network \(M\) over \(A\).

Another possible restriction is to limit all the networks involved in the amalgamation process to only atomic networks. We denote the properties as atomic Network Amalgamation Property (aNAP), and \(k\)-atomic Amalgamation Property \((k\text{-aAP})\).

It is obvious that \(n\text{-AP}\) implies \(m\text{-AP}\) for \(n \geq m\). The smallest interesting case for a qualitative calculus is that of 2-AP. We will approach it step by step, beginning with
\[ |V_{N_1}| = |V_{N_2}| = 3, \text{ i.e., amalgamation of two triangles over a common edge. We will show that this follows from the associativity of } A. \text{ The next case, namely, } |N_1| = 4 \text{ and } |N_2| = 3 \text{ (adding a triangle to a square) is crucial. We will analyse it in some detail and then show that certain strong version of this case implies 2-AP for atomic networks.} \\

### 4.3 Extending Networks

In this section we look at the very basic case of combining two constraint networks of a calculus. Here we are only dealing with amalgamating a constraint network \( N_1 \) with a triangle over an edge.

Formally, it is 2-AP with \( |V_{N_2}| = 3 \). The intuition is that this is similar to extending a network by a triangle.

**Definition 6 ((generic) \( k \)-extensibility).** Let \( A \) be a qualitative calculus \((NA)\) and \( k \) a natural number. \( A \) is \( k \)-extensible if any atomic V-formation \((N_0, N_1, N_2)\) of networks over \( A \), such that \( |V_{N_0}| = 2 \), \( |V_{N_1}| = k \) and \( |V_{N_2}| = 3 \), can be amalgamated by an a-closed network \( M \) over \( A \). If \( N_i \) \((i \in \{0, 1, 2\})\) are non-atomic, then \( A \) is generically \( k \)-extensible.

Therefore, 3-extensibility ensures that any atomic, a-closed network of 3 nodes can be amalgamated with another atomic, a-closed network with 3 nodes over an edge, therefore extended. Both 3 node networks are a subnetwork of the new amalgam. 4-extensibility ensures that any atomic, a-closed networks of 4 nodes can be extended. If a calculus is generically 3-extensible, then any two non-atomic, a-closed triangles sharing an edge in common can be amalgamated by a non-atomic, a-closed network. They are shown in Figure 4.2, where the dotted lines denote the new relation derived in the a-closed amalgam.

We will proceed to show that 3-extensibility is related to associativity of the weak-composition operation of the algebra in the qualitative calculus, and a special case of 4-extensibility allows us to amalgamate any two atomic, a-closed networks over an edge.
4.3.1 3-extensibility

We begin by looking at the case of 3-extensibility, where both $|V_{N_1}| = |V_{N_2}| = 3$. As $N_1$ and $N_2$ are both algebraically-closed with 3 nodes, we call them a-closed triangles.

We want to show that when we combine these two triangles over an edge, the result is an a-closed network of four nodes, and no existing relations change.

**Lemma 5.** Let $A$ be a qualitative calculus and $N$ be a network over $A$ with $|V_N| = 4$ with $\{0, 1, 2\}$ and $\{1, 2, 3\}$ are a-closed triangles, and $R_{03} = R_{01} \circ R_{13} \cap R_{02} \circ R_{23}$. $R_{03} \neq \emptyset$ if and only if $A$ is associative.

**Proof.** ($\Leftarrow$) Suppose $A$ is associative. We use a proof by contradiction to show that $R_{01} \circ R_{13} \cap R_{02} \circ R_{23} \neq \emptyset$. First suppose:

$$(R_{10} \circ R_{02}) \cap (R_{13} \circ r_{32}) = \emptyset$$

Then by the triangle identities:

$$R_{13} \cap [(R_{10} \circ R_{02}) \circ R_{23}] = \emptyset$$
As the weak-composition operator is associative, this is equivalent to:

\[ R_{13} \cap [R_{10} \Diamond (R_{02} \Diamond R_{23})] = \emptyset \]

This is a contradiction, because from the definition of \( R_{03} \) we know that:

\[ R_{03} \subseteq R_{02} \Diamond R_{23} \]

By weak-composing \( R_{10} \) on both sides, we get

\[ R_{10} \Diamond R_{03} \subseteq R_{10} \Diamond (R_{02} \Diamond R_{23}) \]

We also know from the definition of \( R_{03} \) that

\[ R_{03} \subseteq R_{01} \Diamond R_{13} \]

Hence by the triangle identities:

\[ R_{13} \cap R_{10} \Diamond R_{03} \neq \emptyset \]

As \( R_{13} \cap R_{10} \Diamond R_{03} \neq \emptyset \), and \( R_{10} \Diamond R_{03} \subseteq R_{10} \Diamond (R_{02} \Diamond R_{23}) \), it cannot be the case that \( R_{13} \cap [R_{10} \Diamond (R_{02} \Diamond R_{23})] = \emptyset \), hence the contradiction. Therefore \( R_{03} \neq \emptyset \) if \( A \) is associative.

(\( \Rightarrow \)) We will now show that if \( A \) is not associative, we can create a counter-example of two a-closed triangles such that \( R_{03} = \emptyset \).

First, as \( A \) is not associative, there exists base relations \( a, b, c, r \) such that \( r \in (a \Diamond b) \circ c \) but \( r \cap a \Diamond (b \circ c) = \emptyset \). We begin by showing that there exists a base relation \( u \) such that
• For $N_0$: $R_{12} = u$;

• For $N_1$: $R_{10} = a$, $R_{02} = b$, $R_{12} = u$;

• For $N_2$: $R_{13} = r$, $R_{23} = c$, $R_{12} = u$;

• $N_1$ and $N_2$ are $a$-closed.

$u$ exists because we know that as the weak-representation of $A$ is jointly exhaustive and pairwise disjoint (JEPD), $a \circ b \neq \emptyset$. Let $R = a \circ b$. We know that:

$$(a \circ b) \circ c = R \circ c$$

As $r \in (a \circ b) \circ c$, it is clear that $r \cap R \circ c \neq \emptyset$. Then there must exist a base relation $u \in R$ such that $r \cap u \circ c \neq \emptyset$. Also because $R = a \circ b$, it must be the case that $r \cap a \circ b \neq \emptyset$. Therefore, by lemma 4, $N_1$ and $N_2$ are atomic, $a$-closed triangles.

Now we will show that for the above networks, in the amalgam $R_{03} = \emptyset$. Because of the existing relations, we can already narrow down to $R_{03} = a^* \circ r \cap b \circ c$. We use proof by contradiction. Suppose:

$$a^* \circ r \cap b \circ c \neq \emptyset$$

Then, by the triangle identities:

$$r \cap a \circ (b \circ c) \neq \emptyset$$

which is a contradiction, as we defined $r$ as $r \in (a \circ b) \circ c$ but $r \cap a \circ (b \circ c) = \emptyset$. Therefore, $R_{03} = \emptyset$ if $A$ is not associative.

\[\square\]

**Corollary 1.** Let $A$ be a qualitative calculus. $A$ is 3-extensible if and only if the algebra of $A$ is associative.
Proof. Given any atomic \( v \)-formation \((N_0, N_1, N_2)\) where \(|V_{N_0}| = 2\) and \(|V_{N_1}| = |V_{N_2}| = 3\). We simply construct an amalgam from \(N_1 \cup N_2\) and substitute \(R_{03}\) by any base relation \(R \in R_{01} \circ R_{13} \cap R_{02} \circ R_{23}\). It follows from Lemma 5 that the amalgam is a-closed. \(\blacksquare\)

**Theorem 2.** Let \(A\) be an RA. If \(A\) is associative, then \(A\) is generically 3-extensible.

**Proof.** Let \(V_{N_0}\) be \(\{1, 2\}\), \(V_{N_1}\) be \(\{0, 1, 2\}\) and \(V_{N_2}\) be \(\{1, 2, 3\}\). In the amalgam of \(N_1\) and \(N_2\) over \(N_0\), put \(R_{03} = R_{01} \circ R_{13} \cap R_{02} \circ R_{23}\). By Lemma 5 we know that \(R_{03} \neq \emptyset\). To prove that \(A\) is generically 3-extensible, we have to show that the network \(\{0, 1, 3\}\) and \(\{0, 2, 3\}\) are a-closed. As the two are symmetrical, it suffices to show that \(\{0, 1, 3\}\) is a-closed, which requires the following:

\[
(R_0 \circ R_{13}) \cap (R_{02} \circ R_{23}) \subseteq R_0 \circ R_{13} \tag{4.1}
\]

\[
R_{13} \subseteq R_{10} \circ [(R_0 \circ R_{13}) \cap (R_{02} \circ R_{23})] \tag{4.2}
\]

\[
R_0 \subseteq [(R_0 \circ R_{13}) \cap (R_{02} \circ R_{23})] \circ R_{31} \tag{4.3}
\]

Equation 4.1 is trivially true, as the left hands side is the right hand side intersecting with something else.

We will now show that equation 4.2 is also true. We show this by proving that for all \(R \in R_{02}\), \(R \in R_{01} \circ [(R_{10} \circ R_{02}) \cap (R_{13} \circ R_{32})]\).

We use proof by contradiction. Let \(R\) be any base relation such that \(R \in R_{02}\) and \(R \cap R_{01} \circ [(R_{10} \circ R_{02}) \cap (R_{13} \circ R_{32})] = \emptyset\). Then by the triangle identities:

\[
[(R_{10} \circ R_{02}) \cap (R_{13} \circ R_{32})] \cap (R_{10} \circ R) = \emptyset
\]

Because \(R_{10} \circ R \subseteq R_{10} \circ R_{02}\), this can be simplified to:

\[
(R_{13} \circ R_{32}) \cap (R_{10} \circ R) = \emptyset
\]
We apply triangle identities again we get:

\[ R \cap R_{01} \circ (R_{13} \circ R_{32}) = \emptyset \]

This is a contradiction because as \( R_{12} \subseteq R_{13} \circ R_{32} \), therefore \( R_{01} \circ R_{12} \subseteq R_{01} \circ (R_{13} \circ R_{32}) \). Therefore, we have \( R \in R_{02} \subseteq R_{01} \circ R_{12} \subseteq R_{01} \circ (R_{13} \circ R_{32}) \), but \( R \cap R_{01} \circ (R_{13} \circ R_{32}) = \emptyset \). Hence the contradiction.

Equation 4.3 can be proved in similar ways to equation 4.2, that for all \( R \in R_{01} \),

\[ R \in [(R_{01} \circ R_{13}) \cap (R_{02} \circ R_{23})] \circ R_{31}. \]

Again we use proof by contradiction. Let \( R \) be any base relation such that \( R \in R_{01} \) and \( R \cap [(R_{01} \circ R_{13}) \cap (R_{02} \circ R_{23})] \circ R_{31} = \emptyset \). By the triangle identities:

\[ (R_{01} \circ R_{13}) \cap (R_{02} \circ R_{23}) \cap R \circ R_{13} = \emptyset \]

As \( R \circ R_{13} \subseteq R_{01} \circ R_{13} \), this is simplified to:

\[ (R_{02} \circ R_{23}) \cap R \circ R_{13} = \emptyset \]

Again, applying the triangle identities and we get:

\[ R \cap (R_{02} \circ R_{23}) \circ R_{31} = \emptyset \]

By associativity of weak-composition, we have

\[ R \cap R_{02} \circ (R_{23} \circ R_{31}) = \emptyset \]

Similar to the case for Equation 4.2, this is a contradiction because \( R_{21} \subseteq R_{23} \circ R_{31} \), so \( R_{02} \circ R_{21} \subseteq R_{02} \circ (R_{23} \circ R_{31}) \). So we have \( R \in R_{01} \subseteq R_{02} \circ R_{21} \subseteq R_{02} \circ (R_{23} \circ R_{31}) \), but \( R \cap R_{02} \circ (R_{23} \circ R_{31}) = \emptyset \). Hence the contradiction.

As we have shown that all three equations are true, \( A \) is generically extensible.
4.3.2 4-extensibility

A qualitative calculus is 4-extensible if any a-closed network with 4 nodes can be amalgamated with a triangle over an edge. Unlike 3-extensibility, 4-extensibility fails in associative algebras, indeed even representable ones. We now illustrate a counter example.

4.3.2.1 Counter Example: The B₉ Algebra

We propose a qualitative calculus that has a representable relation algebra, but not 4-extensible. Consider the group $\mathbb{Z}_7$ (the integers under addition modulo 7) and for $x, y \in \mathbb{Z}_7$ define

- $x I y$ if $x = y$
- $x G y$ if $x = y \pm 1 \pmod{7}$
- $x B y$ if $x = y \pm 2 \pmod{7}$
- $x R y$ if $x = y \pm 3 \pmod{7}$

Then, this forms a qualitative calculus where the domain is the set of natural numbers from 0 to 6, and $\{I, R, G, B\}$ are the base-relations of its algebra. The algebra is a representable relation algebra as it is the weak-composition operator is associative, and it equals to set-theoretic composition. Its representation using red for $R$, green for $G$ and blue for $B$ is shown in Figure 4.3 a). Its composition table is listed in Table 4.1, and the converse of any relation is itself. This algebra is known as $B_9$ (cf. [28]), and we will refer to this calculus as the $B_9$ calculus.

Consider the network $N$ with $V_N = \{0, 1, 2, 3\}$, $R_{01} = R_{23} = R$, $R_{03} = R_{12} = B$ and $R_{02} = R_{13} = G$. Let $M$ be an a-closed triangle with $V_M = \{2, 3, 4\}$, $R_{23} = R$, and $R_{24} = R_{34} = B$. The amalgam of $N$ and $M$ is inconsistent, as $R_{02} \circ R_{24} \cap R_{03} \circ R_{34} = R$, and likewise $R_{12} \circ R_{24} \cap R_{13} \circ R_{34} = R$. Consequently all the relations in the
triangle with nodes \{0, 1, 4\} are R. As R \cap R \circ R = \emptyset, the triangle, hence the amalgam, is inconsistent.

Because for an extensional calculus the weak-representation equals representation, this allows us to derive the following theorem relating to the computational property of the qualitative calculus.

**Theorem 3.** If a qualitative calculus \((A, U, \mu)\) is extensional and \(A\) is not 4-extensible, then \(a\)-closure does not decide consistency for atomic networks.

**Proof.** In an extensional calculus, the weak-composition operation equals composition. Therefore consistent networks can always be extended by one-shot \cite{37}. However, if \(A\) is not 4-extensible, then by definition there exists an \(a\)-closed, atomic network \(N\) on four nodes that has no \(a\)-closed one-shot extension. Therefore \(N\) is not consistent. \(\square\)

One example of such qualitative calculus is the \(B_9\) Calculus and the network \(N\) as described above. The network is \(a\)-closed but it is not a sub-network of the representation. Therefore the network is not consistent, and \(a\)-closure do not decide consistency for the \(B_9\) calculus.
4.3.2.2 Strong 4-extensibility

Recall that 4-extensibility allows two atomic networks of size 3 and 4 respectively to be amalgamated over one edge without introducing inconsistencies due to a-closure. We now consider a special case of 4-extensibility that allows any two atomic networks of arbitrary size to be amalgamated over one edge.

**Definition 7** (Strong 4-extensibility). Let \( A \) be a qualitative calculus (NA). \( A \) is strongly 4-extensible if any V-formation \((N_0, N_1, N_2)\) of atomic networks over \( A \), with \( N_0 = \{1,2\} \), \( N_1 = \{0,1,2\} \) and \( N_2 = \{1,2,3,4\} \), can be amalgamated by a network \( M \) over \( A \), such that for all \( i \in N_2 \setminus N_0 \)

\[
R_{i0} = (r_{i1} \circ r_{10}) \cap (r_{i2} \circ r_{20})
\]

It is obvious that strong 4-extensibility is simply a special case of 4-extensibility, as the proposed amalgam \( M \) is a-closed. The beauty of strong 4-extensibility is that for a given one-shot extension of a network, the labels for the new edges are precisely the intersection of weak-compositions of labels on existing edges. This property is in fact possessed by both the spatial calculus RCC8 and the temporal calculus Interval Algebra. The property can be easily verified for the given composition table of any calculus through exhaustive search of networks of only base relations. The pseudo-code of the algorithm checking strong 4-extensibility property of a calculus is described in Algorithm 1.

We now show that if a calculus possesses the strong 4-extensibility property, then any two atomic networks can be amalgamated over one edge. The resulting amalgam is algebraically closed.

**Theorem 4.** If a qualitative calculus \( A \) is strongly 4-extensible, then \( A \) has 2-Amalgamation Property if \( N_1, N_2 \) are atomic.

**Proof.** Let \((N_0, N_1, N_2)\) be a V-formation of atomic networks, with \( V_{N_0} = \{0,1\} \).
Algorithm 1 Checking Strong 4-Extensibility Property

FUNCTION: isStrongFourExtensible()

Require: A qualitative calculus \( A \) with base-relations \( B \)

Ensure: Returns true if \( A \) has strong 4-extensibility property, otherwise returns false.

1: for all Atomic V-formation \((N_0, N_1, N_2)\) with
   \( V_{N_0} = \{1, 2\}, V_{N_1} = \{0, 1, 2\} \) and \( V_{N_2} = \{1, 2, 3, 4\} \).
2: \( M = N_1 \cup N_2; \)
3: \( R_{03} = R_{01} \circ R_{13} \cap R_{02} \circ R_{23}; \)
4: \( R_{04} = R_{01} \circ R_{14} \cap R_{02} \circ R_{24}; \)
5: if \( (R_{03} \notin R_{04} \circ R_{43}) \lor (R_{04} \notin R_{03} \circ R_{01}) \lor (R_{34} \notin R_{30} \circ R_{04}) \)
   then
   6: return false;
   7: end if
8: end for
9: return true;

Let \( M = N_1 \cup N_2 \) be the network retaining all the labels from \( N_1 \) and \( N_2 \) and with
the new labels for edges \((x, y)\) with \( x \in N_i \setminus N_j \) and \( y \in N_j \setminus N_i \) \((\{i, j\} = \{1, 2\})\)
defined by \( \ell(x, y) = r_{x0} \circ r_{0y} \cap r_{x1} \circ r_{1y} \). We will show that \( M \) is a-closed. Suppose
the contrary. Then, there is a triangle in \( M \) with edges labelled by relations \( A, B, C \),
such that \( C \not\subseteq A \circ B \). Now, \( A, B \) and \( C \) cannot all be edges from \( N_i \) \((i \in \{1, 2\})\), for
\( N_i \) is a-closed. So at least one of \( A, B, C \) is of the from \( \ell(x, y) \) with \( x \in N_i \setminus N_j \)
and \( y \in N_j \setminus N_i \) \((\{i, j\} = \{1, 2\})\). Notice also that at most two of \( A, B, C \) can be such
(three such edges do not form a triangle). We have then two cases. If there is exactly
one such edge among \( A, B, C \), it violates the assumption of 3-extensibility; if there are
exactly two such edges, then it violates the assumption of strong 4-extensibility. Thus,
\( M \) is a-closed as claimed.

The result shows that any two atomic, a-closed network \( N_1, N_2 \) sharing an edge
can be amalgamated by an a-closed network \( M \). Moreover, \( M \) can be interpreted as
an atomic CSP, as we can include all the atomic constraints from \( N_1 \) and \( N_2 \). We do
not need to include the non-atomic constraints in \( M \) because they can be interpreted as
universal relations. This leads us to the following corollary:

Corollary 2. If a qualitative calculus \( A \) is strongly 4-extensible, and a-closure decides
consistency for atomic CSPs, then the atomic V-formation \((N_0, N_1, N_2)\) with \(|V_{N_0}| = 2\) can be amalgamated by a consistent network \(M\).

**Proof.** We know from Theorem 4 that the amalgam is a-closed. As the amalgam can be interpreted as an atomic CSP, we know the amalgam is consistent. □

This ensures that two atomic, a-closed network can be consistently amalgamated. Therefore, the information can be combined without introducing inconsistencies.

### 4.3.2.3 One-shot extensibility

A qualitative calculus is **extensional** if the weak-composition operation equals composition. However, if this is not the case, it then could raise serious questions about other results relating to the computation complexity of the calculus [37]. This is due to the fact that consistent atomic networks are algebraically closed, but not necessarily path-consistent, and therefore complexity results for a relation in the calculus cannot be transferred to its closure. Li and Wang in 2006 solved this problem for RCC8 by showing that despite that weak-composition is not equal to composition, RCC8 constraint networks can always be consistently extended. This resolves the issues arise from non-extensionality of the calculus, and they called the new property **one-shot extensibility**.

**Definition 8** (One-Shot Extensibility [37]). A qualitative calculus \((A, U, \mu)\) is **one-shot extensible** if any consistent atomic V-formation \((N_0, N_1, N_2)\) with \(|N_0| = 2\) and \(|N_2| = 3\), can be amalgamated by a consistent atomic network \(M\).

Li and Wang proved one-shot extensibility of RCC8 through an extensive case analysis of the semantics of the RCC8 calculus. However, the property of one-shot extensibility is closely related to our interest in combining constraint networks, as it is a special case of Theorem 4 where one of the networks is an atomic, a-closed triangle. Therefore, we can derive the following corollary:

**Corollary 3.** If a qualitative calculus \(A\) is strongly 4-extensible, and a-closure decides consistency for atomic CSPs, then \(A\) is one-shot extensible.
An atomic CSP can be interpreted as a network with atomic and universal relations, hence a-closure decides consistency for atomic CSPs is a strictly stronger condition compared to a-closure decides consistency for atomic networks. However, if we know that a-closure decides consistency for atomic CSPs, which was the case for RCC8, then a simple check for strong 4-extensibility of the calculus is sufficient to determine one-shot extensibility.

**4.4 Finding an Atomic Refinement**

In the previous section we showed that two atomic, a-closed networks can be amalgamated by an a-closed network. However, some of the edges in the amalgam are labelled with a disjunction of many base relations. Furthermore, the amalgam may not have an a-closed, atomic refinement. In such cases the amalgam cannot be consistent. Therefore, we need to show that the amalgam is always proto-consistent, meaning that it will always have an atomic-refinement of the network that is a-closed.

In this section, we first identify a syntactic property that is a close-cousin to one-shot extensibility. We show that it is a sufficient condition to prove representability of a relation algebra. We then follow by providing a purely syntactic proof-procedure to check whether this property would hold. This ensures that all the amalgams of the V- formations would have an a-closed atomic refinement.

**4.4.1 One-shot proto-extensibility**

One-shot extensibility is defined as any consistent network can be consistently extended by one shot. That is, it can be amalgamated with a triangle, and the resulting amalgam is consistent. However, consistency depends on the weak-representation of the calculus, and in order for us to have a generic procedure, we are looking for something that can be syntactically checked by examining the weak-composition table, and hence independent of the weak-representation of the calculus. We now propose a similar property that only
involves the algebra of the calculus, that any extension from an atomic network must be proto-consistent.

**Definition 9 (One-Shot Proto-Extensibility).** A qualitative calculus \((NA) A\) is one-shot proto-extensible if any atomic V-formation \((N_0, N_1, N_2)\) with \(|N_0| = 2\) and \(|N_2| = 3\), can be amalgamated by an atomic network \(M\).

One-shot proto-extensibility ensures that the amalgam has an a-closed atomic refinement. Its advantage over one-shot extensibility is that it is a syntactic notion that is independent of any (weak) representation. We now show that one-shot proto-extensibility is closely related to representability of the algebra of the calculus.

**Theorem 5.** If a calculus is one-shot proto-extensible, then it's algebra \(A\) is a representable relation algebra.

**Proof.** Let \(A\) be a relation algebra with the required property. We build a representation of \(A\) inductively, beginning with any atomic a-closed triangle. At any given stage \(i\), we have constructed an atomic a-closed network \(N_i\). By one-shot proto-extensibility, we can pick any atomic a-closed triangle \(T\) and add it to \(N_i\), in effect amalgamating \(N_i\) and \(T\) over an edge that they share, obtaining an atomic a-closed network \(N_{i+1}\). Let \(N = \bigcup_{i \in \omega} N_i\). Define \(\mu: A \to N\) putting \(\mu(a) = \{(x, y): \ell_N(x, y) = a\}\) for an atom \(a \in A\). By finiteness of \(A\), each \(u \in A\) is a join of finitely many atoms. Thus, we can extend \(\mu\) onto the whole universe of \(A\) setting \(\mu(u) = \mu(a_1) \cup \cdots \cup \mu(a_n)\), where \(a_1, \ldots, a_n\) are atoms with \(u = a_1 \lor \cdots \lor a_n\). It can be verified that the so defined \(\mu\) is a representation of \(A\).

We have shown that one-shot proto-extensibility of a calculus is a sufficient condition for representability of its relation algebra. However, it is not the case that one-shot extensibility implies one-shot proto-extensibility. This is due to the existence of atomic a-closed networks that are not consistent. A counter-example again can be found in the \(B_9\) Algebra, which is representable, and hence one-shot extensible. It is not one-shot proto-extensible, as illustrated by the network in Figure 4.3 b).
4.4.2 Refining the intersections

We now show the steps necessary for proving one-shot proto-extensibility. Under strong 4-extensibility, each non-atomic relation in the amalgam of two networks over a common edge is precisely the intersection of the two paths from one node of the network to another. However, to refine them to base-relations, we have to make sure that all the refinements from the intersections are compatible with each other ensuring that the amalgam is still a-closed.

One way to ensure that there is always an atomic refinement for these relations in an a-closed atomic amalgam is for the algebra to have a base-relation that possesses the property of a flexible atom (cf. [46]).

**Definition 10** (Flexible Atom). *A relation algebra with a set of base-relations \( \mathcal{B} \) has a flexible atom \( a \) if the following condition holds:

\[
\exists a \in \mathcal{B} : \forall b, c \in \mathcal{B} \setminus \{1'\}, a \in b \circ c
\]

Intuitively, a flexible atom is a base-relation that is contained in any weak-composition of two atomic relations. Therefore, one only has to replace all the non-atomic relations in the amalgam by the flexible atom to make the amalgam atomic and a-closed.

Although a flexible atom is extremely difficult to come by, as we do not know of a qualitative calculus in qualitative spatial and temporal reasoning whose algebra has this property, it gives us a hint of what a solution may be like. So instead we propose to construct an ordering of base-relations that will emulate this property when refining amalgams. Specifically, we find a sequence of base-relations such that for any non-atomic relation \( R \) in the amalgam, we can refine it to the first element in the sequence that is a member of \( R \), and after all the refinements the network is guaranteed to be a-closed.
Algorithm 2 Finding the choice-refinement of \( R \) over \( \Theta \)

**FUNCTION:** \texttt{choiceRefinement}(\( R, \Theta \))

**Require:** A relation \( R \), and a sequence of base-relations \( \Theta \).

**Ensure:** Returns the choice-refinement of \( R \) over \( \Theta \)

1. \( \Theta' \leftarrow \Theta \);
2. \textbf{while} \( \Theta' \neq \emptyset \) \textbf{do}
3. \quad \texttt{r = removeFirst}(\( \Theta' \));
4. \quad \textbf{if} \( r \in R \) \textbf{then}
5. \quad \quad \textbf{RETURN} \( r \);
6. \quad \textbf{end if}
7. \textbf{end while}
8. \textbf{RETURN} \emptyset;

**Definition 11** (Choice-refinement). Let \( A \) be a relation algebra with a set of atoms \( B \) and \( \Theta \) be a sequence of base-relations. We define a choice-refinement of a non-atomic relation \( R \) over \( \Theta \) as the first member of \( S \) that is a refinement of \( R \).

As we have defined the choice-refinement of a relation to a sequence of base-relations, we can proceed to use it to define the sequence of base-relations that serves our purpose. Because the sequence is designed to emulate the workings of a flexible atom, we call it **Flexibility Ordering**.

**Definition 12** (Flexibility Ordering). For a strongly 4-extensible relation algebra \( A \), its Flexibility Ordering is a sequence \( S \) of base relations, such that for any an atomic \( V \)-formation \( (N_0, N_1, N_2) \) with \( V_{N_0} = \{1, 2\}, V_{N_1} = \{0, 1, 2\} \) and \( V_{N_2} = \{1, 2, 3, 4\} \), the non-atomic relations in \( V_1 \cup V_2 \) can be replaced by their respective choice-refinements over \( S \) and the resulting network is \( a \)-closed.

The main idea is that we define a sequence \( \Theta \) of base-relations such that for any amalgam \( M \), we can iteratively replace any non-atomic edge \( R \) in \( M \) by its choice-refinement \( r \) over \( \Theta \) (Algorithm 2). We define \( \Theta \) is such a way that the new relation \( r \) will never cause the algebraic closure algorithm to fail with any existing atomic relations in \( M \). Whether a sequence \( \Theta \) is a Flexibility Ordering can be checked by exhaustively testing all possible atomic \( V \)-formations with \( V_{N_0} = \{1, 2\}, V_{N_1} = \{0, 1, 2\} \) and
§4.4 Finding an Atomic Refinement

Algorithm 3 Determine whether a sequence \( \Theta \) is a Flexibility Ordering

FUNCTION: isFlexOrdering(\( \Theta \))

Require: A sequence of base-relations \( \Theta \) of qualitative calculus \( A \)
Ensure: Returns true if \( \Theta \) constitute a Flexibility Ordering of \( A \), otherwise returns false.

1: for all Atomic V-formation \( (N_0, N_1, N_2) \) with 
\( V_{N_0} = \{1, 2\}, V_{N_1} = \{0, 1, 2\} \) and \( V_{N_2} = \{1, 2, 3, 4\} \). do
2: \( M = N_1 \cup N_2 \);
3: \( R_{03} = R_{01} \circ R_{13} \cap R_{02} \circ R_{23} \);
4: \( R_{04} = R_{01} \circ R_{14} \cap R_{02} \circ R_{24} \);
5: \( r = \text{choiceRefinement}(R_{03}, S) \);
6: \( r' = \text{choiceRefinement}(R_{04}, S) \);
7: if \( r \cap r \circ R_{34} = \emptyset \) then
8: RETURN false;
9: end if
10: end for
11: RETURN true;

We now propose an algorithm to find such a sequence for a qualitative calculus that possesses the strong 4-extensibility property. First, we describe conditions in which base-relations can be added to a partially-constructed sequence (Algorithm 4). For a given sequence \( \Theta \), that may not cover all cases, we test if a new base-relation \( r \) that is not in \( \Theta \) to see if we can add it to the end of \( S \). The base-relation \( r \) can be added to the end of \( S \) if for an amalgam \( M \) of any atomic V-formation \( (N_0, N_1, N_2) \) with \( |V_{N_0}| = 2, |V_{N_1}| = 3 \) and \( |V_{N_1}| = 4 \), \( r \) is contained in \( R_{03} \) but no member of \( S \) is, the following conditions hold:

1. If \( R_{04} \) is already atomic, then when we replace \( R_{03} \) with \( r \), the triangle \( \{0, 3, 4\} \) is a-closed.

2. If \( R_{04} \) is not atomic, but there exists a choice refinement \( r_{04} \) of \( R_{04} \) over \( \Theta \), then when we replace \( R_{03} \) with \( r \), \( R_{04} \) with \( r_{04} \), the triangle \( \{0, 3, 4\} \) is a-closed.

3. If \( R_{04} \) is not atomic, and there is no choice refinement \( r_{04} \) of \( R_{04} \) over \( \Theta \) but \( R_{04} \) contains \( r \), then when we replace both \( R_{03} \) and \( R_{04} \) by \( r \), the triangle \( \{0, 3, 4\} \) is a-closed.
Second, we describe how such a sequence could be constructed (Algorithm 5). We start off with an empty sequence. We then incrementally add base-relations of the calculus to the sequence satisfying the above conditions. When the sequence of base-relations already cover all cases, then we are done. Otherwise, when no further base-relations can be added to the sequence, we backtrack by deleting the last element of the sequence, and proceed with the next base-relation in the natural-ordering of the base-relations.

Equipped with a Flexibility Ordering, we can now derive the following result:

**Theorem 6.** *If a calculus is strongly 4-extensible, and it has a Flexibility Ordering, then it has 2-atomic Amalgamation Property.*

**Proof.** From Theorem 4 we get a network $M$ that is an amalgam of the atomic V-formation $(N_0, N_1, N_2)$, but the new edges between $N_1$ and $N_2$ may not be atomic. However, with a Flexibility Ordering we can refine each of these edges to atomic relations, knowing that similar atomic refinements of other new edges will not introduce an inconsistent triple, since we have checked all possible cases in the construction of the Flexibility Ordering. Therefore the entire network is refined to be atomic and a-closed, thus it has 2-atomic Amalgamation Property.

One-shot proto-extensibility would follow from the above theorem, as it is a special case of 2-aAP with $|V_{N_2}| = 3$.

**Corollary 4.** *If a calculus has 2-atomic Amalgamation Property, then it is one-shot proto-extensible.*

This result allows us to prove representability of an RA $A$ for a given weak-composition table. This means that the algebra $A$ can be a part of an extensional qualitative calculus $(A, U, \mu)$. It also implies that consistency can be preserved when amalgamating two atomic a-closed networks over two nodes if we know that a-closure decides consistency for only atomic networks.
Algorithm 4 Test if a base relation \( r \) can be added to a partially-built Flexibility Ordering \( S \)

FUNCTION testCompatible\((r, S)\)

Require: A base-relation \( r \), and a sequence of base-relations \( S \) from a qualitative calculus \( A \)

Ensure: Returns true if \( r \) can be added to the end of \( S \), else returns fail.

1: for all Atomic V-formation \((N_0, N_1, N_2)\) with

\[
V_{N_0} = \{1, 2\}, \quad V_{N_1} = \{0, 1, 2\} \quad \text{and} \quad V_{N_2} = \{1, 2, 3, 4\}. \quad \text{do}
\]
2: \( M = N_1 \cup N_2 \);
3: \( R_{03} = R_{01} \circ R_{13} \cap R_{02} \circ R_{23} \);
4: \( R_{04} = R_{01} \circ R_{14} \cap R_{02} \circ R_{24} \);
5: \( r' = \text{choiceRefinement}(R_{04}, S) \);
6: if \( r \in R_{03} \land \text{choiceRefinement}(R_{03}, S) = \emptyset \) then
7: if \( R_{04} \notin B \land r \cap R_{04} \circ R_{43} = \emptyset \) then
8: RETURN false;
9: else if \( r' \neq \emptyset \land r' \cap r \circ R_{34} = \emptyset \) then
10: RETURN false;
11: else if \( r' = \emptyset \land r \cap r \circ R_{34} = \emptyset \) then
12: RETURN false;
13: end if
14: end if
15: end for
16: RETURN true;

4.4.3 Empirical Evaluations

Both RCC8 and IA are prime candidates to test for Flexibility Orderings, as they are well known and non-trivial calculi in the spatial-temporal domain, and their respective relation algebras are both strongly 4-extensible. For RCC8, the procedure found the Flexibility Ordering: \((\text{DC}, \text{EC}, \text{PO}, \text{TPP}, \text{TPPi})\), whereas for IA, the procedure found \((<, di, o, s, oi, f)\). Hence we have proved from their composition table that both have 2-aAP and their relation algebras are representable.

Computationally the worst case of the procedure is \( O(|B|!) \). This is due to the permutations involved when constructing a sequence of length \(|B|\). However, the occurrence of the worst-case scenario would be extremely rare, as most branches of the search tree will be terminated earlier than exhaustive search, thus trimming down a majority of potential search space. For example, if the base-relation \( a \) is not compatible with the
Algorithm 5 Constructing a Flexibility Ordering

FUNCTION: flexOrdering()

Require: A qualitative calculus A with a set of base relations B
Ensure: Output a Flexibility Ordering of A, otherwise fail.

1: Φ ← B;
2: RETURN searchFlexOrdering(∅, Φ)

FUNCTION: searchFlexOrdering(Θ, Φ)

Require: Two sequences of base-relations Θ and Φ over calculus A.
Ensure: Return false if a Flexibility Ordering cannot be found based on partial sequence Θ, otherwise a full Flexibility Ordering of A.

1: if isFlexOrdering(Θ) then
2:  RETURN Θ;
3: else if Φ = ∅ then
4:  RETURN false;
5: end if
6: Φ′ ← Φ;
7: r ← removeFirst(Φ);
8: while testCompatible(r, Θ) do
9:  Θ′ ← addToTail(Θ, r);
10:  remove(Φ′, r);
11:  Θ″ ← searchFlexOrdering(Θ′, Φ′);
12:  if Θ″ ≠ false then
13:   RETURN Θ″
14: end if
15:  addToTail(Φ′, r);
16:  r ← removeFirst(Φ);
17: end while
18: RETURN false;

empty sequence, then we would not have to check all possible sequences with a as the first element. For IA, with 13 base-relations, the procedure found an ordering in 4 seconds on a Intel Core2Duo 2.4GHz processor with 2GB RAM, and for RCC8 it found a solution in less than a second. Therefore, our procedure is widely applicable.

4.5 Summary and Future Work

In this Chapter, first we provided algebraic methods over the weak-composition table to prove that two atomic, a-closed networks of any size can be amalgamated by an a-closed
over one edge given strong 4-extensibility property. Therefore, if a-closure decides consistency for atomic CSPs of the calculus, then the two networks can always be consistently combined. We also know that if a calculus do not have this property, but weak-composition equals composition, then a-closure cannot decide consistency for atomic networks. If weak-composition is not equal to composition, but a-closure decides consistency for atomic CSPs, then strong 4-extensibility allows us to know whether tractability of a relation can be transferred to its closure. The strong 4-extensibility property can be quickly checked by our proposed algorithm for any calculus given its weak-composition table.

Secondly, we show further procedures that could prove the resulting amalgamated network is guaranteed to have an atomic, a-closed refinement. Like the algorithm determine strong 4-extensibility, the procedure is also a purely symbolic method involving the algebra of the calculus. This allows us to preserve consistency when combining two constraint networks over two nodes. It also allows us to show that the algebra of the calculus is indeed a representable relation algebra.

The first obvious future step is to see if there are any algebraic methods to show that two atomic a-closed networks can be amalgamated over \( n \) nodes for \( n > 2 \), and whether there are methods for determine amalgamation for non-atomic networks. It would also be interesting to see if there are any general methods for determining atomic Network Amalgamation Property, one that would be applicable to any newly-proposed qualitative calculus in spatial and temporal reasoning.

Secondly, our proposed notion of one-shot proto-extensibility is a sufficient, but not necessary condition for representability of a relation algebra, as there are other representable relation algebras which are not one-shot proto-extensible. Mathematically, it would be interesting to investigate the connection between one-shot proto-extensibility and Hirsch-Hodkinson type games [26], and whether Hirsch-Hodkinson games can be interpreted as a sequence of one-shot extensions.
In this Chapter we focus on decomposing constraint networks for deciding consistency in qualitative spatial and temporal reasoning (QSTR). Unlike standard constraint programming, a constraint network in QSTR is always a complete graph. This is due to the jointly-exhaustive and pairwise-disjoint (JEPD) nature of weak-representation of the calculus, where between any two elements of the domain exactly one base-relation holds. That means at least one base relation must hold between any two nodes in the constraint network for it to be consistent. If no base relation is possible between two nodes, the edge is labelled with an empty relation, and a network that contains empty relations cannot be consistent. When translating a CSP to a constraint network, we transcribe the constraint between two variables to the corresponding label between two nodes, and when there were no explicit constraints, we label the edge with the universal-relation, which is a disjunction of all possible base-relations. Because the network is always a complete graph, previous fixed-parameter-tractable results with respect to the treewidth of the constraints are not applicable, as the treewidth would always be equal to the number of nodes in the graph.

As mentioned in Chapter 2, deciding consistency for constraint networks of Allen’s Interval Algebra (IA) [1], is done generally by a backtracking search. We first approximate consistency of the network by applying the algebraic-closure (a-closure) algorithm, which removes the impossible relations from all triples of nodes in the network. If all the possible relations are removed between two nodes, the network is inconsistent, as no relation can hold between the two nodes and the network cannot be realized. A net-
work with all such impossible triples removed by a-closure, but which do not contain any empty relations, is known as an a-closed network. If after applying a-closure the network becomes an atomic network, where all the relations in the network are base relations of the calculus, then we know the network is consistent as it was shown that a-closure decides consistency for atomic networks of IA [70].

If after a-closure the network does not contain an empty relation, but it is also not an atomic network, then we need to search for an a-closed, atomic refinement of the original network, which is an a-closed, atomic network where any relation \((R'_{ij})\) between node \(i, j\) is a member of the possible disjunctive relation \((R_{ij})\) between \(i\) and \(j\) in the original network. Here we select any edge between nodes \(\{i, j\}\) whose label \(R_{ij}\) is not a base relation, refine the label to a base relation \(r_{ij} \in R_{ij}\), and apply a-closure again. When a-closure fails for the new network, we backtrack. This is done iteratively until we either find an a-closed atomic network that is a refinement of the original network, by which we know the original network is consistent, or we backtrack to the root node of our searchtree, which tells us the network is not consistent.

One major advancement in the effort for more efficient methods in deciding consistency was the discovery of tractable subsets of Interval Algebra. Formally, a tractable subset is a subset of relations from the algebra, where consistency of any network with only the subset of relations can be decided in polynomial time. Nebel and Bürgert [52] discovered the tractable subset ORD-Horn, and showed that it is actually a maximal tractable subset. They showed that a-closure decides consistency for any network with relations only from the ORD-Horn subset. This greatly simplifies the search process. Instead of selecting and refining any edge whose label is not a base relation, we only select those that are not in the ORD-Horn tractable subset, and refine it to members of ORD-Horn. Nebel [51] showed that this greatly improves the efficiency of search in practice. Subsequently improvements such as eligible and frozen constraints [8] have been included in most up-to-date constraint-based solver GQR [18], which has been compared favourably against all other existing approaches [74].
The major alternative to GQR-type constraint-based solving with tractable subsets is the SAT-based method proposed by Pham, Thornton and Sattar in 2007 [54]. The encoding can be based on either the constraints between the intervals as given in the network (event-based), or the constraints between the end-points of those intervals (point-based). In either case, the Boolean formulas are constructed in such a way that each solution of the formula corresponds to a path-consistent atomic refinement of the network, and vice versa. Hence the formula is satisfiable if and only if the network is consistent. We refer the reader to Chapter 2 for a detailed description of the encoding, but note that the size of the encoding is proportional to the number of node triples in the network (\( \frac{n!}{3!(n-3)!} \) for a network over \( n \) nodes), as it introduces a set of clauses to ensure that every triple satisfies the condition required by path consistency. This memory requirement is a major drawback to scale this approach to solve much larger problems. The advantage of this approach is that it encodes the problem declaratively, and solution finding is delegated to any of the off-the-shelf SAT solvers, the efficiency of which has greatly improved over the past decade.

In this Chapter we investigate a new paradigm for deciding consistency in QSTR, with a particular focus on the Interval Algebra (IA) [1]. We look at a property of a calculus about amalgamation of constraint networks proposed in the previous Chapter, and show that it can be used to decompose a large constraint network in IA into many smaller constraint networks. If each of the smaller subnetworks are consistent, and their overlapping parts agree with each other, then the large constraint network is also consistent. The decomposition process makes use of the universal relations from the initial network, and uses a theoretical property to ensure that they will never be reduced to the empty relation. We show that this leads to the first fixed-parameter-tractable result in QSTR, and the best-performing solver on the most difficult instances of current benchmark problems.
5.1 Decomposing Networks

As mentioned earlier, a weakness of the SAT-based approach is the size of the encoding, which for large networks can outgrow available memory or can make SAT solving otherwise inefficient. In particular, many networks are not densely connected (the missing edges implicitly represent universal relations) and the encoding must nevertheless treat the network as a complete graph and generally include clauses for all its triangles, leading to a cubic complexity in all cases.

Our solution to this issue starts with a theoretical result about how two consistent networks sharing a common subnetwork can be amalgamated into a larger consistent network, provided a particular property holds. We then present a method that uses this result and recursively partitions a network into subnetworks before they are individually encoded into Boolean formulas, thus avoiding the need to include many triangles across the subnetworks that have edges labelled by universal relations. For the network depicted in Figure 5.2a, for example, we will show that only 32 out of the 56 node triples need to be considered in checking a-closure.

5.1.1 A Theoretical Property

If two networks have common variables, it is clear that their combination (to be defined more precisely later) might not be consistent even if both are individually consistent. We are interested in those cases where the two networks $M, N$ assign the same labelling between their common variables, or more formally, where $M_{V_M \cap V_N} = N_{V_M \cap V_N}$. Previously, Li et al. [35] introduced the Network Amalgamation Property (NAP), which guarantees that two such networks over a calculus can be amalgamated into a larger network such that enforcing a-closure on the resulting network does not change any existing constraints.

Formally, two a-closed networks $M, N$ are said to be amalgamated by another network $P$, called their amalgam, if $V_P = V_M \cup V_N$, $M \subseteq P$, $N \subseteq P$, and $P$ is a-closed.
Here we consider a form of NAP where we restrict the two networks and their amalgam to be atomic networks. We call this the Atomic Network Amalgamation Property (aNAP).

**Definition 13 (aNAP).** A qualitative calculus $\mathbf{A}$ has the aNAP if any a-closed atomic networks $M, N$ over $\mathbf{A}$ satisfying $M_v \cap N_v = N_v \cap M_v$ have an atomic amalgam.

Figure 5.1 illustrates this property. Consider the networks $M$ and $N$ over a calculus $\mathbf{A}$ shown in Figure 5.1a, with their common subnetwork shown on the far left. A network over vertices $V_M \cup V_N$ is shown in Figure 5.1b, where the dotted edge represents the universal relation and all other edges retain their labels from $M$ and $N$. The dotted edge is the only edge with a non-atomic label and the calculus $\mathbf{A}$ having the aNAP would guarantee the existence of an atomic refinement of the edge that results in an a-closed atomic network.

We now present a simple lemma, and then use it to establish a sufficient condition for the aNAP and show that it is satisfied by $\mathbf{IA}$.

**Lemma 6.** Any consistent atomic network must be a-closed.

**Proof.** Let $i, j, k$ be any three nodes of any consistent atomic network $N$. We need to show that $\ell_N(i, k) \subseteq \ell_N(i, j) \circ \ell_N(j, k)$. Let $A = \ell_N(i, k) \cap (\ell_N(i, j) \circ \ell_N(j, k))$ (note the “$\circ$” for the standard composition). We know that $A \neq \emptyset$, or $N$ would be inconsistent. In other words, the atomic relation $\ell_N(i, k)$ has a nonempty intersection (A) with $\ell_N(i, j) \circ \ell_N(j, k)$; hence it must be a subset of $\ell_N(i, j) \circ \ell_N(j, k)$ by the definition of weak-composition.
Theorem 7. A qualitative calculus has the aNAP if a-closure implies strong $n$-consistency for all atomic networks of size $n$ over the calculus.

Proof. Given a qualitative calculus $A$ satisfying the above condition and two a-closed atomic networks $M, N$ over $A$ such that $M_{\mathcal{V}_M \cap \mathcal{V}_N} = N_{\mathcal{V}_M \cap \mathcal{V}_N}$, let $P$ be a new network such that $\mathcal{V}_P = \mathcal{V}_M \cup \mathcal{V}_N$, $\ell_P(i, j) = \ell_M(i, j) \forall i, j \in M$, $\ell_P(i, j) = \ell_N(i, j) \forall i, j \in N$, and $\ell_P(i, j)$ equals the universal relation otherwise. We wish to show that $P$ is consistent, which will establish the aNAP as any consistent network has a consistent, and hence a-closed (by Lemma 6), atomic refinement. Take any consistent instantiation $\alpha_m$ of $M$, and consider its projection $\alpha_{mn}$ on $\mathcal{V}_M \cap \mathcal{V}_N$. Since $N$ is strongly $|\mathcal{V}_N|$-consistent, $\alpha_{mn}$ can be extended to a complete consistent instantiation $\alpha_n$ of $N$. The extension will not conflict with $\alpha_m$ because none of the additional variables are in $\mathcal{V}_M$. Furthermore, none of the constraints of $P$ not in $M, N$ can be violated because they are all universal. Hence the combination of $\alpha_m$ and $\alpha_n$ is a consistent instantiation of $P$. 

Corollary 5. IA has the aNAP.

Proof. A-closed atomic networks of size $n$ in IA are guaranteed to be strongly $n$-consistent [70]. Hence IA has the aNAP.

5.1.2 Decomposing Networks

Having established that IA has the aNAP, we now proceed to show how it allows us to identify a subset of node triples that can be soundly ignored when checking a-closure.

Consider the network $P$ over eight nodes $\{v_1, \ldots, v_8\}$ shown in Figure 5.2a, where edges labelled with universal relations are omitted. Let us make a cut through $M$ as shown by the dotted line in Figure 5.2a, breaking the network into subnetworks $M$ (Figure 5.2b) and $N$ (Figure 5.2c). Note that we have added to $M$ all the endpoints of the cut edges from the other side together with the incident edges; this ensures that all constraints of $P$ are retained in the subnetworks.
§5.1 Decomposing Networks

Figure 5.2: Partitioning a network.

Now suppose we are able to establish that $M$ and $N$, respectively, have $a$-closed atomic refinements $M'$ and $N'$ such that $M'_{\{v_4,v_5,v_6\}} = N'_{\{v_4,v_5,v_6\}}$ (i.e., they agree on the atomic relations between their common vertices); the aNAP will then allow us to conclude that $M'$ and $N'$ have an atomic amalgam—but this amalgam must be a refinement of $P$, meaning that $P$ must be consistent. In other words, we have been able to decide consistency for $P$ without examining any of a total of 21 triangles, namely those that include at least one (invisible) edge between $\{v_1, v_2, v_3\}$ and $\{v_7, v_8\}$.

Formally, let $M$ and $N$ be two subnetworks of an IA network $P$ such that every edge of $P$ labelled with a non-universal relation appears in $M$ or $N$. The following theorem indicates that in finding an $a$-closed atomic refinement of $P$, all node triples of $P$ can be soundly ignored unless they reside entirely within $M$ or $N$.

**Theorem 8.** Let $M, N, P$ be IA networks such that $M \subseteq P$, $N \subseteq P$, and either $i, j \in V_M$ or $i, j \in V_N$ whenever $\ell_P(i, j)$ is not the universal relation. If $M$ and $N$, respectively, have $a$-closed atomic refinements $M'$ and $N'$ such that $M'_{V_M \cap V_N} = N'_{V_M \cap V_N}$, then $P$ has an $a$-closed atomic refinement $P'$ such that $M' \subseteq P'$ and $N' \subseteq P'$.

**Proof.** Without loss of generality, assume that $P$ is a connected graph when all edges labelled with the universal relation are disregarded (otherwise the proof applies to each of the connected components, and a-closed atomic refinements of these components can...
be trivially combined and extended into an a-closed atomic refinement of $P$).

By virtue of the aNAP, $M'$ and $N'$ have an atomic amalgam; call it $P'$. By the definition of amalgam, $M' \subseteq P'$, $N' \subseteq P'$, and $P'$ is a-closed. Hence it suffices to show that $P'$ is a refinement of $P$. First, $\mathcal{V}_{P'} = \mathcal{V}_P$ because every vertex of $P$ must appear in $M$ or $N$, and hence in $P'$. Second, for any pair of nodes $i, j \in \mathcal{V}_P$, if $i, j \in \mathcal{V}_M$, then $\ell_{P'}(i, j) = \ell_M(i, j) \subseteq \ell_P(i, j)$; if $i, j \in \mathcal{V}_N$, then $\ell_{P'}(i, j) = \ell_N(i, j) \subseteq \ell_P(i, j)$; otherwise $\ell_P(i, j)$ must be the universal relation and we trivially have $\ell_{P'}(i, j) \subseteq \ell_P(i, j)$. This means that $P'$ is a refinement of $P$.

We can apply the same idea recursively to each of the subnetworks. Let us proceed with $M$ from Figure 5.2b. In recursion we need to take one additional measure, to ensure that the common nodes of the two subnetworks previously created stay together (so that the atomic refinements found for the two subnetworks are guaranteed to agree on their common edges). In this example, that is to say that we cannot split $v_4$, $v_5$, and $v_6$ when partitioning $M$. Figure 5.3 gives a feasible partitioning in this regard, allowing us to further ignore 3 triangles: $(v_1, v_4, v_5)$, $(v_1, v_4, v_6)$, $(v_1, v_5, v_6)$. In the end, we can decide consistency for $P$ by examining only 32 out of the 56 triangles.
5.2 Tractability of Deciding Consistency

In the previous section, we have presented a method based on graph decomposition that allows more efficient checking of a-closure. We now use this technique to show that consistency of IA networks in general, and the solution of our compact SAT encoding in particular, are both fixed-parameter tractable with respect to treewidth.
For this theoretical analysis we enlist a formal tool known as a *dtree* (decomposition tree), which originated in probabilistic reasoning and has since found use in other domains [11]. In general, a dtree is a full binary tree where the root represents a given problem and for every non-leaf node, its two children represent a partitioning of the parent problem into two subproblems. For our purposes, the problems are IA networks, and for every network $P$ partitioned into subnetworks $M$ and $N$, we require, as mentioned earlier, that every non-universal edge of $P$ appears in $M$ or $N$. Figure 5.4 depicts a dtree for the network in Figure 5.2a, where each node is labelled with the network it represents. (Note that we can now view the recursive partitioning procedure described in the previous section as computing a dtree.)

Analogous to the corresponding notions in [11], we can define the *vertices*, *separator*, and *cutset* of a dtree node. For any dtree node $T$, we will denote its two children by $T.left$ and $T.right$, and identify the nodes with the networks they represent. The *vertices* of a dtree node $T$ is simply $V_T$. The *separator* of a non-leaf dtree node $T$ is the set of vertices shared between the networks at its two children: $\text{separator}(T) = V_{T.left} \cap V_{T.right}$; the separator of a leaf dtree node is its vertices. The *cutset* of a dtree node is its separator minus all its ancestors’ cutsets.

### 5.2.1 Recursive Conditioning

We now present a formal algorithm that decides the consistency of an IA network $N$ given a dtree $T$ for the network, and then discuss its computational complexity. The pseudocode is given in Algorithm 6, and follows the recursive conditioning framework of [11]. Given an atomic refinement $\alpha$ of a subnetwork of $N$, we write $N|_{\alpha}$ to denote the result of applying the same refinement to $N$.

The algorithm works as follows. At the root we identify the overlap of the two subnetworks ($N_{T.cutset}$ on line 7) and consider all its atomic refinements, each leading to a pair of recursive calls on the two children (line 8, note again that we use a dtree node to also refer to the network it represents). This ensures that any atomic refinements we
Algorithm 6  Recursive conditioning on IA network

\texttt{consistent\( (\text{Network} : N, \text{Dtree} : T) \)}

1: if \( T \) is a leaf then
2: \hspace{1em} for all atomic refinements \( \alpha \) of \( N_{T,\text{cutset}} \) do
3: \hspace{2em} if \( a \rightarrow \frac{\text{closed}}{}(N|\alpha) \) then
4: \hspace{3em} return true;
5: \hspace{2em} end if
6: \hspace{1em} end for
7: \hspace{1em} return false;
8: else
9: \hspace{1em} for all atomic refinements \( \alpha \) of \( N_{T,\text{cutset}} \) do
10: \hspace{2em} if \( \text{consistent}(T.\text{left}|\alpha, T.\text{left}) \land \text{consistent}(T.\text{right}|\alpha, T.\text{right}) \) then
11: \hspace{3em} return true;
12: \hspace{2em} end if
13: \hspace{1em} end for
14: \hspace{1em} return false;
15: end if

find for the children will agree on their common subnetwork. The recursion terminates when we reach a dtree leaf (line 1), where we replace the pair of recursive calls with a single efficient a-closure check (line 3).

In summary, Algorithm 6 returns true if and only if the network \( N \) has an a-closed atomic refinement (which is equivalent to \( N \) being consistent). We note that the correctness of this algorithm can be easily established by induction on the level of recursion, where the base case (lines 2–5) is trivial and the general case (lines 7–10) is correct by virtue of Theorem 8.

It is known that if the network \( N \) has \( n \) vertices and treewidth \( w \), then there exists a dtree of depth \( \leq \log n \) in which every cutset has size \( \leq w \) [11]. It is also known that a recursive conditioning algorithm, such as Algorithm 6, will finish in time \( O(n \exp(k \log n)) \) given such a dtree [11], as long as the number of branches at each recursion (lines 2 & 7) is \( O(\exp(k)) \). The number of atomic refinements of a network over \( w \) vertices is at most \( 13^w (w - 1)^2 \) (since IA has 13 atomic relations and a complete graph over \( w \) vertices has \( w (w - 1) / 2 \) edges). Therefore the number of branches on line 2 and line 7 of Algorithm 6 is \( O(\exp(w^2)) \), and hence Algorithm 6 has a time complexity
of $O(n \exp(w^2 \log n))$. In other words, consistency of IA networks is fixed-parameter tractable with respect to the treewidth of the network.

### 5.2.2 Solving the SAT Encoding

We now extend our analysis above in a simple way to show that our compact SAT encoding is optimal in a specific sense—that it is guaranteed to be solvable with the same time complexity. First observe that given a dtree $T$ for a network $N$, our SAT encoding is completely determined (i.e., we omit triangles across subnetworks $T.left$ and $T.right$, and so on recursively). The set of Boolean variables shared by the encoding for subnetwork $T.left$ and that for $T.right$ is precisely the set of variables encoding $N_{T.separator}$, and the same holds recursively for all other nodes.

To facilitate our further analysis, we now adapt the notions of vertices, separator, and cutset to the Boolean setting. Given our SAT encoding $\Delta$ of a network based on a dtree, and any node $T$ in the dtree, we will write $\Delta_T$ to denote the subset of the formula $\Delta$ encoding the subnetwork represented by $T$; the $b.variables$ of $T$ is the set of Boolean variables of $\Delta_T$; the $b.separator$ of $T$ is $T.left.b.variables \cap T.right.b.variables$ if $T$ is not a leaf, and $T.b.variables$ otherwise; the $b.cutset$ of $T$ is its $b.separator$ minus all its ancestors’ $b.cutsets$.

We are now ready to present Algorithm 7, which decides the satisfiability of our Boolean encoding $\Delta$ in a way analogous to Algorithm 6. Given a Boolean formula $\Delta$ and an instantiation $\alpha$ of a subset of its variables, we write $\Delta|_\alpha$ to denote the simplification of $\Delta$ after instantiating some of its variables to constants according to $\alpha$. We omit a detailed description of this algorithm given its similarity to Algorithm 6, but note that instead of an $a$-closure check, line 3 is now a direct evaluation of the Boolean formula as all variables have been instantiated.

The number of Boolean variables introduced to encode each edge of a network is bounded by a constant [54]; hence the size of $T.b.cutset$ (lines 2 and 7 of Algorithm 7) is proportional to the number of edges of $N_{T.cutset}$ (line 2 & 7 of Algorithm 6). This implies
Algorithm 7 Recursive conditioning on SAT encoding

\[
sat(\text{Encoding} : \Delta, Dtree : T)
\]

1: if \( T \) is a leaf then
2: for all instantiations \( \alpha \) of \( T.b\text{-cutset} \) do
3: if \( \Delta_T|_{\alpha} \) evaluates to true then
4: return true;
5: end if
6: end for
7: return false;
8: else
9: for all instantiations \( \alpha \) of \( T.b\text{-cutset} \) do
10: if \( sat(\Delta_T\text{,lefl}|_{\alpha}, T\text{,left}) \land sat(\Delta_T\text{,right}|_{\alpha}, T\text{,right}) \) then
11: return true;
12: end if
13: end for
14: return false;
15: end if

that the same complexity bound from Section 5.2.1, \( O(n \exp(w^2 \log n)) \), applies to Algorithm 7 as well.

5.3 Experimental Results

We conducted two groups of experiments. In the first, we evaluate the performance of our compact SAT encoding, showing that it significantly advances the state of the art in qualitative temporal reasoning. In the second, we synthesized a set of benchmarks of increasing size but bounded treewidth, and show that practical solvers indeed appear to exhibit a scaling behaviour there consistent with the theoretical tractability of these benchmarks.

We use three existing state-of-the-art solvers: Nebel’s solver [51], GQR (version 994) [18], and Pham’s solver [54]. The first two were used with their best-performing heuristics enabled. In particular, Nebel’s solver was run with options {static, global, queue} enabled, which had been shown to vastly improve the performance of the solver [51]. MiniSat v2.070721 [13] was used to solve all SAT formulas, both for Pham’s
encodings and for our compact encoding. The tests were ran on 2.4GHz processors with a 2-hour time limit and 2GB memory limit.

The parameters we used in calling hMETIS (for partitioning the networks) are: 
\[ \text{options}[] = \{1, 10, 1, 1, 1, 0, 1, 0\}, \text{ubFactor} = 35; \text{in recursion we set options}[6] = 1 \text{ and specify a set of nodes that needs to stay together.} \]

5.3.1 Performance of Compact SAT Encoding

The standard benchmarks for qualitative temporal reasoning are those randomly generated based on a given number of nodes, average degree, and average label size. In this study we used benchmarks around the phase-transition region with an average label size of 6.5, and excluded those that were quickly found to be inconsistent by an initial a-closure check.

5.3.1.1 Smaller Networks

In the first part of this study we tested 27,000 IA networks with from 50–100 nodes and an average degree ranging from 8–12. We tested the event-based and point-based variants of both Pham’s encoding and our compact encoding. Out of all the tests only Nebel’s solver failed on 9 instances; every other solver managed to solve all the instances.

We first examine the size of the Boolean formula generated for each encoding (in terms of the number of clauses), plotted in Figure 5.5. The data indicates that the size of our compact encoding (partEvent and partPoint in the graph) is approximately half that of the encoding of [54], in both the event-based and point-based variants. (Note that the partEvent and PhamPoint curves overlap and cannot be distinguished).

Figure 5.6 compares the performance of all solvers. It indicates that both Nebel’s solver and GQR solve easy instances very efficiently, whereas our point-based compact encoding (partEndpoint) dominates all other approaches in average CPU time on the harder instances.
5.3.1.2 Larger Networks

In the second part of the study we generated 100 networks of size 110–200 with an average label size 6.5 for each degree from 8–12 (the phase-transition region was identified as being between degrees 10 and 11.5). We recorded the number of instances solved by each solver within the time and memory limits, and the time taken to solve them.

Figure 5.7 plots the number of instances solved against network degree, showing that our compact encoding dominates all other solvers for networks of any degree. The superior scalability of our encoding is further illustrated in Figure 5.8, which focuses on
the hard region and plots the number of instances solved against network size.

![Figure 5.7: Number of solved instances against network degree.](image1)

![Figure 5.8: Number of solved instances against network size, for the phase-transition region.](image2)

To further complete the picture, Figure 5.9 shows the number of instances solved against CPU time. Nebel’s solver and GQR solved a large number of instances very quickly, but struggled to solve the harder instances even when more time was available. The point-based encoding of [54] performed better than Nebel’s solver, but was inferior to GQR. Once again our point-based compact encoding solved substantially more instances than all other approaches, and solved most of the instances in under 1 hour.
5.3.2 Benchmarks of Bounded Treewidth

In the second group of experiments, we synthesized a set of networks of increasing size but bounded treewidth. This is done by connecting a series of satisfiable 50-node networks into a "chain" with a small, fixed number of edges randomly inserted between each pair of neighbouring networks. The 50-node subnetworks are selected from the phase-transition region with average degree 9.5. We vary the number of such subnetworks from 2–10; hence the overall network size ranges from 100–500. To create multiple series of benchmarks, we also vary the number of edges connecting the subnetworks from 2–10.

It is known that the treewidth of such chain-shaped networks is close to the treewidth of the individual components (50-node subnetworks), and hence stays bounded regardless of the length of the chain. We therefore expected these to be relatively easy instances.

For these experiments we used GQR, the best-performance previous solver identified in Section 5.3.1, and our point-based compact SAT encoding. To obtain better control over the SAT solving process, we created a variant of the MiniSat solver that instantiates variables in an order consistent with that used by Algorithm 7, so that it can be regarded effectively as a direct implementation of Algorithm 7. This is achieved by extracting a
variable group ordering [27] from a dtree and enforcing that ordering on the SAT solver; the reader is refer to [27] for a detailed description of this technique.

The results of these experiments are plotted in Figure 5.10 and 5.11, where our compact encoding solved with MiniSat and with the modified MiniSat are referred to as “Original” and “VarMod,” respectively. (There were very occasional failures for all three solvers; the plots do not account for those.) Each of the six plots shown is for a specific number of edges between subnetworks (those omitted are similar to a subset of those shown).

The first observation we make is that in all cases, the performance of all solvers degrades gracefully with increased network size (one may compare this with the case in Section 5.3.1 where all solvers failed a substantial portion of instances at size 200). This concurs well with the theoretical tractability of these instances that can be predicted based on their bounded treewidth.

The second observation we make is that solving our SAT encoding with a variable ordering induced from the dtree is beneficial when the number of edges connecting the subnetworks is small; when that number increases, the default SAT solver catches up and the two more or less converge in performance. This can be explained intuitively by the fact that as the number of “linking” edges increases, they tend to “blend” into the neighbouring subnetworks and their uniqueness in being able to efficiently break up the network diminishes (the effectiveness of the dtree-induced variable ordering mainly stems from its tendency to put the variables encoding the linking edges first in the ordering).

Finally, we note that GQR performed well on these benchmarks (losing to SAT only when the number of linking edges is less then 4), which is consistent with our observation in Section 5.3.1 that it was the best performer on relatively easy instances (although GQR employs no structure-based techniques to directly exploit the bounded treewidth, it is still likely that the theoretical tractability is otherwise exploited—exactly how could be an interesting topic for future work).
Figure 5.10: Averaged CPU time for bounded treewidth instances with 2, 3 and 4 edges between subnetworks.
Figure 5.11: Averaged CPU time for bounded treewidth instances with 5, 7, and 10 edges between subnetworks
5.4 Conclusion and Discussion

We proposed a systematic method, based on graph decomposition, that allows a subset of node triples to be ignored when checking path consistency of atomic IA networks, and identified a theoretical property of the qualitative calculus that ensures the soundness of this method. Combining it with an existing SAT encoding, we produced a new solver for IA that empirically dominates existing approaches in efficiency and scalability on difficult instances.

Furthermore, we used our decomposition scheme to show that consistency of IA networks is fixed-parameter tractable with respect to treewidth, which is in contrast to the fact that previous approaches are obliged to treat all networks as complete graphs and hence are not able to directly exploit low treewidth. A similar analysis also revealed that the solution of our compact SAT encoding is fixed-parameter tractable in the same terms. This fixed-parameter tractability is empirically illustrated using a set of artificially synthesized instances of bounded treewidth.

Our notion of the aNAP is related to the patchwork property defined in [42]. We note that when path consistency decides consistency for a given calculus, then the aNAP implies the patchwork property.

On the encoding of networks into SAT, it should be noted that in general, reduced formula size does not necessarily lead to faster SAT solving. While our results indicate that detecting and removing redundant clauses via partitioning is indeed beneficial, a second interesting observation can be made regarding the correlation of encoding size and solution time: As Pham et. al. [54] have previously observed and we again confirmed, the point-based encoding in fact leads to both smaller formulas and faster solutions than the event-based encoding whether or not partitioning is employed. A plausible explanation is that the Point Algebra, which is used in the point-based encoding, is much smaller than IA, and the smaller composition table leads to fewer clauses required per triangle for a given network. This indicates that SAT-based approaches may be made more effi-
cient if a network can be translated into an equivalent set of constraints over a smaller calculus.

In this Chapter we demonstrated the usefulness of our method for solving IA networks. However, our decomposition-based method is applicable to any qualitative calculus that has the aNAP and for which algebraic closure implies consistency for atomic networks. While the former property holds for many calculi, it is far less known where the latter holds. We presented a sufficient condition (Theorem 1) for the aNAP, but it remains unclear whether it is also necessary. Another question is whether a purely syntactic and automated proof for the aNAP exists, and if so, if it would allow us to automatically identify calculi to which our decomposition-based method is applicable.
In this Chapter we investigate constraint networks of very large calculi. These calculi are characterized by a large number of base relations, and are often developed to represent multiple aspects of space and time. The approach is to combine existing calculi that cover the different aspects. However, this leads to these very large calculi where it is not possible to even practically enumerate all the relations. It is an ongoing discussion within the research community whether such large calculi are too large for practical reasoning.

To address this question, in this Chapter we develop a procedure for reasoning about some of the largest known qualitative calculi in qualitative spatial and temporal reasoning: the Rectangle Algebra (RA) with approximately $10^{50}$ relations, and the Block Algebra (BA) with approximately $10^{661}$ relations. We achieve this by first decomposing the network with techniques similar to those introduced in the previous Chapter. Secondly, we decompose the relations of the calculus into a point-based representation that is based on a much smaller qualitative calculus. We show that when this is followed, reasoning over RA and BA is possible and can be done efficiently in many cases. This is a clear indication that it is possible to achieve one of the main goals of the field: performing efficient reasoning for very highly expressive spatial and temporal representations.

The remainder of this Chapter is structured as follows. Section 6.1 introduces the need for large qualitative calculi. In Section 6.2 we introduce the necessary background on qualitative calculi and different methods and techniques used in this Chapter. Further
Reasoning with Large Qualitative Calculi

details and references can be found in [66]. In Section 6.3 we present a novel procedure for reasoning over large calculi such as RA or BA, and prove its correctness. In Section 6.4 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 6.5 discusses our results.

6.1 Why Large Qualitative Calculi

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, previously introduced examples are the Region Connection Calculus RCC8 [57] for representing topological relationships between spatially extended entities, and the Interval Algebra (IA) [1] for representing relationships between convex intervals on a directed line. RCC8 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 [67]. One of the major advantages of a qualitative representation is that it allows the easy representation of uncertainty and indefinite information by specifying a union of possible base relations. This gives us a total number of $2^{13}$ relations that can be distinguished for a given set of base relations $B$. For RCC8 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 [51], despite it being an NP-complete problem.

An obvious question now is how large can a qualitative spatial or temporal calculus become 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) [4], 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 too many to remember and 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 quickly, “effective reasoning” means to be able to do formal reasoning at all. The reason we can achieve efficient solutions for these NP-hard reasoning problems is in 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 to otherwise hard problem instances [51]. 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 [59; 60]. 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 Chapter. This leaves the possibility of manually identifying large tractable subsets. Other factors include 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 significant 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 today’s computers impossible.

The question of how large a qualitative calculus can reasonably be becomes more and more important because of the current direction in which the field is heading. One of the main challenges in the field of qualitative spatial and temporal reasoning is 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 (RCC8) or topology plus direction (IA). If we wish 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 [20]. 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 Chapter we analyze if reasoning with very large calculi is possible and if it can be performed 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 [33]. 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, BA [5], which is a combination of IA over three dimensions and consists of a massive $2^{13 \times 13 \times 13}$ (approximately $10^{661}$) 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.

### 6.2 Current Approaches for Large Calculi

#### 6.2.1 Combinations of Calculi

In recent years there has been 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 choose 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 occurs 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 of this is the bipath-consistency method introduced by Gerevini and Renz [20] 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 $b_ib_j \subseteq D \times D$ which is the intersection of $b_i$ and $b_j$. The RA is an example of a cross-product combination as there are $13 \times 13$ new base relations, all possible combinations are covered.

Wölfl and Westphal [75] analyzed 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 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.
6.2.2 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 have described the point-based propositional encoding of the IA as developed by [54] in Chapter 2. However, for the purpose of clarity, as it is intrinsically related to this Chapter, we will reintroduce it here. The encoding transforms a set of constraints $\Theta$ 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 $M_{lm}$ denotes the interval relation between interval $l$ and $m$, $D_{ij}$ denotes the PA relation between points $i$ and $j$, $x_{ij}^v$ denotes the propositional variable which is true if and only if the PA relation $\mu$ is true between points $i$ and $j$. We also state that $\mu(r)$ 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: $\bigwedge_{u,v \in D_{ij}} \neg x_{ij}^u \lor \neg x_{ij}^v$ for $u \neq v$

3. SUP: $\bigwedge_{u \in D_{ik}, v \in D_{kj}} \neg x_{ik}^u \lor \neg x_{kj}^v \lor x_{ij}^{w_1} \lor \cdots \lor x_{ij}^{w_m}$
   where $\{w_1, \ldots, w_m\} = D_{ij} \cap u \cup v$

4. FOR: $\bigwedge_{r \neq M_{lm}} \neg x_{l-}^u \lor \neg x_{l-m}^u \lor \neg x_{l+m}^y \lor \neg x_{l+m}^z$
   where $\mu(r) = (u, v, y, z)$
The first set of clauses (at-least-one) ensures that there is at least one PA relation between any two endpoints, the second (at-most-one) ensures that there is at most one PA relation between any two endpoints. The two set of clauses ensure that the final solution is an atomic network. The third set of clauses (support) encodes the composition constraint that enforces path-consistency. The fourth (forbidden) guarantees 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.

6.2.3 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 [74]. However, by utilizing a new divide-and-conquer approach [33], that divides a large constraint network into a number of smaller networks, and combines these with a SAT-based encoding, it leads to a significant speed-up. We have shown in the previous Chapter that this divide-and-conquer approach significantly reduces the size of the SAT encoding and the solving time for the most difficult IA networks.

6.3 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 creating a parallel combination. The resulting constraints are then encoded into a propositional SAT formula, while solution-
6.3 A Hybrid Procedure for RA and BA

finding can be delegated to state-of-the-art SAT solver Minisat [13].

6.3.1 Network Decomposition

We first partition the given CSP network by applying the network decomposition approach as described in the previous Chapter. The technique partitions the CSP network into a number of smaller networks sharing overlapping parts. Li et.al. 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 algebraic-closure.

For the first condition, it follows from the work in [4] that path-consistent atomic networks of RA and BA are strong-n-consistent. This result, together with the fact that path-consistency is equivalent to a-closure for these calculi [61] and the theorem 7 in the previous Chapter, entails that both RA and BA possess Atomic Network Amalgamation Property.

6.3.2 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 [54]. 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 \( p_1, p_2 \). Any BA base-relation \( r \) between two blocks \( p \) and \( q \) can be transformed into a collection of PA relations along the \( x, y \) and \( z \) axes between the representative points \( p \) and \( q \). We define this transformation as

\[ \mu(r) = \{ \forall x \ p_1 q_1, \forall x \ p_1 q_2, \forall x \ p_2 q_1, \forall x \ p_2 q_2, \forall y \ p_1 q_1, \forall y \ p_2 q_2, \forall y \ p_2 q_1, \forall y \ p_2 q_1, \forall z \ p_1 q_1, \forall z \ p_1 q_2, \forall z \ p_2 q_1, \forall z \ p_2 q_2 \} \]

where for example, \( \forall x \ p_1 q_1 \) denotes the PA relation between points \( p_1 \) and \( q_1 \) along the \( x \) axis.

**Definition 14.** Given a BA network \( \Theta \) with \( n \) nodes and its corresponding PA networks \( P_x, P_y \) and \( P_z \) (each with \( 2n \) points, \( P_{[x|y|z]}^{[1...2n]} \)), the corresponding point-based CSP of \( \Theta \) is a triple \((X,D,C)\) where

- \( X = \{ X_{x,ij}, X_{y,ij}, X_{z,ij} | i, j \in [1...2n], i < j \} \). 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:

\[
\begin{align*}
& \forall u \in D_{x,ik}, v \in D_{x,kj} \quad X_{x,ik} = u \wedge X_{x,kj} = v \\
& \quad \Rightarrow X_{x,ij} \in D_{x,ij} \cap (u \circ v) \\
& \forall u \in D_{y,ik}, v \in D_{y,kj} \quad X_{y,ik} = u \wedge X_{y,kj} = v \\
& \quad \Rightarrow X_{y,ij} \in D_{y,ij} \cap (u \circ v)
\end{align*}
\]
Theorem 9. Let $\Theta$ be a BA network and $\Phi$ be the corresponding point-based CSP defined in Definition 14, then $\Theta$ is consistent if and only if $\Phi$ is consistent.

Proof. ($\Rightarrow$) Let $\Theta'$ be a consistent scenario of $\Theta$. As the labels of $\Theta'$ consist of only base relations, it corresponds to a point-based CSP $\Phi'$ that is a consistent scenario of $\Phi$. ($\Leftarrow$) Let $\Phi'$ be an instantiation of $\Phi$ 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 $\Theta'$ from $\Phi'$ by labelling 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, $\Theta'$ is a consistent scenario of $\Theta$ as $\Phi'$ satisfies all constraints of $C$.

6.3.3 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 axes. 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 14, we need to add the following set of clauses for every relation $M$ between blocks $p$ and $q$: 

\[- \bigwedge_{u \in D_{z,ij}, v \in D_{z,kj}} X_{z,ik} = u \land X_{z,kj} = v \Rightarrow X_{z,ij} \in D_{z,ij} \cap (u \circ v)\]

\[- \bigwedge_{r \in M_{pq}} \mu(r) \neq (X_{x,p1q1}, X_{x,p1q2}, X_{x,p2q1}, X_{x,p2q2}, X_{y,p1q1}, X_{y,p1q2}, X_{y,p2q1}, X_{y,p2q2}, X_{z,p1q1}, X_{z,p1q2}, X_{z,p2q1}, X_{z,p2q2}).\]
where $\mu(r) = (u_1, v_1, y_1, z_1, u_2, v_2, y_2, z_2, u_3, v_3, y_3, z_3)$

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

### 6.3.4 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. $N_1(<, <, <) \lor (>, >, >) N_2$
2. $N_1(<, s, s) N_3$
3. $N_2(d_i, >, >) N_3$
4. $N_3(<, d, f) \lor (<, f, f_i) \lor (o, d, d_i) \lor (>, d, f_i) N_4$

First we apply the divide-and-conquer technique. It partitions the network into two smaller networks: $(N_1, N_2, N_3)$ and $(N_3, N_4)$. 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:
• \((N_{1s} < x N_{1e}) \land (N_{1s} < y N_{1e}) \land (N_{1s} < z N_{1e})\)

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:

• \((N_{1s} < x \lor > x N_{2s}) \land (N_{1s} < x \lor > x N_{2e})\)

• \((N_{1e} < x \lor > x N_{2s}) \land (N_{1e} < x \lor > x N_{2e})\)

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

• \(\neg (N_{1s} < x N_{2s} \land N_{1s} > y N_{2s})\)

• \(\neg (N_{1s} < x N_{2s} \land N_{1s} > z N_{2s})\)

• \(\neg (N_{1s} < y N_{2s} \land N_{1s} > z N_{2s})\)

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. Constraints 2 and 3 are encoded in a similar way and complete the transformation.

### 6.4 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...
Figure 6.1: Comparing Average and Median CPU time for instances solved by both solvers, and Failure Rate over RA Networks between our Hybrid Solver and GQR
Figure 6.2: Percentage of satisfiable instances found by the hybrid solver for RA networks size 20, 30, 40 and 50, over avg. label size and avg. degree. 5 instances per datapoint.

Figure 6.3: Percentage of unsatisfiable instances found by the hybrid solver for RA networks size 20, 30, 40 and 50, over avg. label size and avg. degree. 5 instances per datapoint.
Figure 6.4: Percentage of failed instances by the hybrid solver for RA networks size 20, 30, 40 and 50, over avg. label size and avg. degree. 5 instances per datapoint.

Figure 6.5: CPU Time by the hybrid solver for RA networks of size 20, 30, 40 and 50, over avg. label size and avg. degree. 5 instances per datapoint. For the purpose of illustration the averaged CPU time of the failed instances are marked at the time limit 3600s.
Figure 6.6: Percentage of satisfiable, unsatisfiable, failed instances and average CPU Time (top to bottom) by the hybrid solver for BA networks of size 10 and 20 (left to right). 5 instances per datapoint. For the purpose of illustration the averaged CPU time of the failed instances are marked at the time limit 3600s.
is much larger than the number of relations that are used in the randomly generated instances. We used an Intel Core2Duo 2.4 GHz processor with 2GB of RAM in all experiments.

6.4.1 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.

We compared the performance of our hybrid approach with GQR (Fig. 6.1) over average and median CPU time, and failure rate. Our hybrid solver consistently solved more instances than GQR, and the median CPU time is considerably lower. The average CPU time of both solvers are about equal when the network size equals 50. A possible explanation is that at this network size there is little difference in the degree of difficulty for the instances that are solvable by both solvers. However, there are other instances at this network size that are solved by our solver, but are beyond reach for GQR.

An interesting observation is that there is a phase-transition where randomly generated instances change from mostly consistent to mostly inconsistent (Fig. 6.2, 6.3). This is also where the hard problems are, as evident by the high failure rate in the region (Fig. 6.4). However, unlike for previously evaluated calculi such as RCC8 or IA, 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 RCC8 or IA do not allow us to vary the average label size significantly as there are only 8 and 13 different labels respectively. Therefore, it was not previously possible 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 which cannot be solved within the time limit. The
phase transition regions lie within the regions of the largest runtimes (see Fig. 6.5).

### 6.4.2 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 average label size and not the average degree. Furthermore, all instances that are not in the phase transition region can be easily solved (see Fig. 6.6).

### 6.5 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 as to whether such large calculi are desirable and feasible or whether they defeat the purpose of a qualitative representation that makes only a small number of distinctions. We argue 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 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 which is based on transformation of a set of spatial or temporal constraints into a propositional Boolean formula and which does not rely on tractable subsets. We pro-
pose 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 analyzed 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 with previous studies, randomly generated instances show 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. In our tests our procedure can easily solve most instances, while only some instances around the phase transition region could not be solved in reasonable time. For Rectangle Algebra, our procedure performs better and solves more instances than other methods, while for Block Algebra, our procedure is thus far the only method that can produce a solution in a reasonable time.

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. On the contrary, these instances are actually useful as they provide easy to generate benchmarks for modern SAT solvers.
Space and time have inherent structure. We intuitively use the structure to reason about everyday life, from getting credit cards from our wallets to moving a piano across a hall. This structure makes spatial and temporal problems interesting. This thesis investigates exploiting these structures in the constraint networks of a qualitative calculus. We investigated what makes the problem hard for networks of some qualitative calculi and easy for others. For the most difficult problems, we examined ways to decompose large networks into many smaller ones in order to make the process more efficient. Not only have we shown empirically that this new approach produces the best performing solver for the most difficult classes of problems, but also that it theoretically identifies the first class of fixed-parameter tractable instances in qualitative spatial and temporal reasoning. We have shown that by decomposing the constraint network, and transforming it into a Boolean Satisfiability problem, reasoning about even large qualitative calculi for very expressive spatial and temporal representations is not beyond reach.

7.1 Summary of Contributions

Here the contributions of the thesis from each of its chapters are summarized.

7.1.1 On Algorithms for Automated Complexity Proofs

Chapter 3 investigated whether certain constraint networks can help us to determine the computational complexity of reasoning with certain qualitative calculi. There have been
many qualitative spatial and temporal calculi proposed in the past 15 years, and identifying the computational complexity is one of the most important tasks for any new qualitative calculus. Reasoning had been shown to be NP-hard for most of the interesting calculi, but the use of maximal tractable subsets of a calculus can make reasoning significantly more efficient. A procedure proposed by Renz in 2007 automatically identifies tractable subsets of a qualitative calculus [59]. However, up until now all complexity results had to be manually derived, therefore we cannot know whether the tractable subsets identified by Renz’s procedure are maximal tractable subsets without manually proving that all extensions of these tractable subsets are NP-hard. This Chapter proposed a new procedure for automatically identifying intractable subsets of a qualitative calculus. When combined with Renz’s algorithm, the new procedure automatically identifies the maximal tractable subsets of the Allen’s Interval Algebra, the spatial calculus RCC8 and Ligozat’s Cardinal Direction Calculus. These results can also be adapted to other quantitative calculi to identify their maximal tractable subsets automatically.

7.1.2 On Network Amalgamation

Chapter 4 considered combining constraint networks in QSTR without introducing inconsistencies. This is an important issue, as Li and Wang [37] have showed that important computational complexity results in literature may be incorrect, unless a constraint network of the qualitative calculus can always be consistently combined with another network with 3 nodes over a shared edge. The work in this Chapter involved identifying a general property that allows any two consistent networks to be combined over an edge of a qualitative calculus. It proposed an algorithm that checks for this property when given the weak-composition table of a qualitative calculus. This allows us to test whether we can consistently combine information of a qualitative calculus, and maintain a consistent world-view while acquiring new information. This property is also important for ensuring complexity results of a relation can be transferred to its closure for a qualitative calculus, and proves that the relation algebra of the qualitative calculus is
indeed a representable relation algebra.

7.1.3 On Network Decomposition

Chapter 5 reversed the amalgamation process to look into decomposing a constraint network of a qualitative calculus. We first identified a key theoretical property of a qualitative calculus that ensures the soundness of the consistency problem after decomposition. We then introduce a systematic method based on graph decomposition that exploits the structure of the network to produce a more compact and efficient encoding of the Interval Algebra as a SAT problem. Empirically, the results demonstrated that the new methods scales to much larger problems and exhibits a consistent and significant improvement in the efficiency of the current state-of-the-art constraint-based solvers on the most difficult class of instances. A new algorithm was introduced to show that deciding consistency for Interval Algebra is fixed-parameter tractable based upon the decomposition scheme. The proposed SAT encoding is also guaranteed to have the same time-complexity.

7.1.4 On Reasoning with Networks of Large Qualitative Calculi

Chapter 6 examined whether efficient reasoning is possible for qualitative calculi with a large number of base relations. These calculi arise from the need to deal with multiple aspects of space and time, where we combine existing calculi that cover the different aspects. A procedure was developed for reasoning about some of the largest known qualitative calculi, involving the decomposition of the network and a novel SAT encoding that transforms the problem into one that involves a much smaller calculus. The empirical evaluations showed that by using the new approach, efficient reasoning is possible in many cases, and that the approach outperform the traditional constraint-based approach for the qualitative calculus Rectangle Algebra, and it produced the only viable solver to date for the much bigger calculus, Block Algebra. This result indicates that efficient reasoning is possible for highly expressive spatial and temporal representations.
7.2 Direction for Future Work

This thesis concludes with some remarks concerning profitable directions for future research.

Research in qualitative spatial and temporal reasoning is currently organized by dividing information into different aspects of space. These include distance, direction, topology, interval-relations, size or shape. For each aspect, different qualitative calculi have been proposed that represent spatial information with respect to this particular aspect and enable reasoning over this information. In this thesis we have looked at a few well-known examples: the Interval Algebra [1], which reasons about relations between intervals along a directed line; the Region Connection Calculus [57], which concerns the topological relationship between regions; and the Cardinal Directions Calculus [40], which concerns the relative direction of points in 2D space. Each aspect and its corresponding qualitative calculi are interesting in their own right. However, practical applications that use spatial information do not make this distinction between different aspects, and need to deal with information related to different aspects simultaneously. They require representation and reasoning capabilities for all the spatial information they have available, not just for topological information, and not just for distance information. Even though these requirements of practical applications have been recognized within the spatial reasoning community, there has been no attempt as yet to integrate spatial information in a general, systematic and coherent way.

A consequence of this lack of coherence is that, first, there has been a lack of applications for the tools created. There is no simple way in which a non-expert in QSTR can choose and adopt spatial calculi and algorithms and apply them to their problems. Second, there is currently a lack of clear direction for future research about what new tools to develop.

What is needed is a systematic re-engineering of the field. We need a more problem-centric view where appropriate tools are developed to handle available spatial data.
These tools must be able to be customized to specialized application domains. The general direction should be oriented towards addressing the increasing need for making sense of spatial data.

More specifically, I believe there are three main tasks ahead:

1. Categorization of Problems (Classify): We need to establish a focus on the problems that use spatial information and on their requirements. Therefore, we need a comprehensive classification of spatio-temporal problems into categories according to different criteria and practical requirements. We will then need to map the existing calculi and algorithms to these categories and identify areas that are not yet sufficiently covered and require further research. This will help to guide future research in QSTR.

2. Integration of Information (Combine): Real-world problems require spatial and temporal information over different aspects. Currently there is no coherent way of integrating the representations of these different aspects. This problem can be tackled from two directions. (a) We develop a general semantical foundation of different types of spatial information that allows integration via their common semantics. (b) We develop methods and tools for integrating spatial information on a syntactical level. Additional steps will be needed to guarantee consistent integration of the different semantics. Both approaches will be focused on problem categories and requirements identified in the first task.

3. Efficient Reasoning (Compute): Once the spatial and temporal information can be integrated, we need to be able to process the integrated knowledge, to reason about it or to query it. In most realistic settings, this will require a scalable approach that can deal with a significant amount of spatio-temporal data. The aim here is to develop algorithms that allow us to obtain efficient solutions to realistic reasoning problems. Currently this is only possible for individual aspects of spatial information. The task is to develop such algorithms for integrated spatial information.
Conclusion

over different aspects. We need to develop and modularly organize algorithms and tools to tackle the different categories of problems we identified.

I believe the field will turn to seek a deeper understanding of the problems and their requirements. Theoretical analysis of the various aspects of spatial and temporal knowledge is needed to understand how they can be integrated, and tools need to be built to deal with the integrated information required by practical applications. The modular organization of the tools can be expected to lead to efficiency gains and avoid re-inventing the wheel. This approach would lay the ground work for a rich and expressive representation of spatial knowledge. It will allow others to take off-the-shelf reasoning tools and further customize them for their specific applications.
Bibliography


[41] Gérard Ligozat and Jochen Renz. What is a qualitative calculus? a general framework. In Zhang et al. [76], pages 53–64.


[45] Alan K. Mackworth and Eugene C. Freuder. The complexity of some polynomial net-


