UNIVERSITÉ D'ARTOIS

École Doctorale SPI Université Lille Nord de France

Algorithmic Contributions to Qualitative Constraint-based Spatial and Temporal Reasoning

THÈSE

présentée et soutenue publiquement le 27 février 2017

pour l'obtention du

Doctorat de l'Université d'Artois

(spécialité Informatique)

par

Michael Sioutis

Rapporteurs :	Philippe Balbiani - Directeur de Recherche	CNRS, IRIT
	Maroua Bouzid - Professeur	Université de Caen Basse-Normandie
Examinateurs :	Mehul Bhatt - Professeur	Université de Brême (Allemagne)
	Jean-François Condotta - Professeur	Université d'Artois (Co-directeur)
	Bertrand Mazure - Professeur	Université d'Artois (Co-directeur)
	Yakoub Salhi - Maître de Conférences	Université d'Artois (Co-encadrant)
Invité :	Gérard Ligozat - Professeur	Université Paris-Sud 11

Centre de Recherche en Informatique de Lens — UMR 8188

Acknowledgments

I would like to thank the Centre de Recherche en Informatique de Lens (CRIL), the Université d'Artois, and the region of Nord-Pas-de-Calais (which has become a part of the new region Hauts-de-France as of 1 January 2016) for funding my PhD studies. In particular, I would like to thank the CRIL laboratory for offering me an office and the technical resources to work on my thesis and, most importantly, for providing me with an environment of brilliant researchers that kept me motivated and in pursuit of my full potential.

I would like to thank my advisors, Professors Jean-François Condotta and Bertrand Mazure and Assistant Professor Yakoub Salhi, for trusting me to undertake this thesis topic under their guidance and direction. Each of them contributed in a different, yet always positive, way to my work. I would especially like to mention Jean-François Condotta, who I was fortunate enough to meet in Paris prior to my PhD journey, and who was one of the major reasons I decided to move to Lens and become a member of the CRIL team.

I would like to thank Professor Sanjiang Li for collaborating with me on several research problems and for inviting me to the University of Technology Sydney (UTS) in Australia, where I was also honored to get to know and collaborate with one of his excellent PhD students, Shufeng Kong, and an exceptional research fellow, Dr. Jae Hee Lee. Unfortunately, it was only after my stay in Australia that I also got to know in person Dr. Zhiguo Long, another excellent researcher and a former PhD student in UTS under the supervision of Professor Li. However, I had the pleasure of collaborating with Dr. Long remotely during and after my research visit in UTS.

I would like to thank my family for their love and their support. Having a warm and welcoming place back in your home country, always makes things significantly easier.

I would like to thank my friends and collaborators, Panagiotis Liakos and Dr. Katia Papakonstantinopoulou, for keeping me in balance and for engaging in wonderful conversations with me about science and otherwise.

Finally, special thanks go to Maria Kontogianni, for whom I have a deep affection. Special thanks also go to my special friend on the other side of the Atlantic Ocean, Fátima Peñaloza.

I dedicate this thesis to my late grandfather, Michael Sioutis, Sr.

Table of Contents

List of Tables

List of Figures

Chapter 1

Introduction

Qualitative Spatial and Temporal Reasoning (QSTR) is a major field of study in Artificial Intelligence and, particularly, in Knowledge Representation, which deals with the fundamental cognitive concepts of space and time in an abstract manner. This qualitative manner of dealing with space and time is in line with the qualitative abstractions of spatial and temporal aspects of the common-sense background knowledge on which the human perspective of physical reality is based. For instance, in natural language we use verbs such as *inside*, *before*, and *north of* to spatially or temporally relate one object with another object or oneself, without resorting to providing quantitative information about these entities. More formally, qualitative spatial and temporal reasoning restricts the rich mathematical theories that deal with spatial and temporal entities to simple qualitative constraint languages. The conciseness of the constraint languages used in the qualitative approach provides a promising framework that further boosts research and applications in spatial and temporal reasoning, as it allows for rather inexpensive reasoning about entities located in space and time. For example, some of these calculi may be implemented for handling spatial Geographic Information Systems (GIS) queries efficiently and some may be used for navigating and communicating with a mobile robot [Hazarika, 2012; Bhatt *et al.*, 2011].

The first constraint language to deal with space or time in a qualitative manner was proposed by Allen in [Allen, 1981; Allen, 1983], called Interval Algebra, and it has been mostly used for reasoning about time ever since. Allen wanted to define a framework for reasoning about time in the context of natural language processing that would be reliable and efficient enough for reasoning about extracted and newly indrocuded temporal information in a qualitative manner. In particular, Interval Algebra uses intervals on the timeline to represent entities corresponding to actions, events, or tasks. Interval Algebra has become one of the most well-known qualitative constraint languages, due to its use for representing and reasoning about temporal information in various applications. Specifically, typical applications of Interval Algebra involve planning and scheduling [Allen and Koomen, 1983; Allen, 1991; Pelavin and Allen, 1987; Dorn, 1995], natural language processing [Song and Cohen, 1988], temporal databases [Snodgrass, 1987; Chen and Zaniolo, 1998], multimedia databases [Little and Ghafoor, 1993], molecular biology [Golumbic and Shamir, 1993] (e.g., arrangement of DNA segments/intervals along a linear chain involves particular temporal-like problems [Benzer, 1959]), and workflow [Lu *et al.*, 2006].

Inspired by the success of Interval Algebra, Randell, Cui, and Cohn developed the Region Connection Calculus (RCC) in [Randell *et al.*, 1992]. As its name suggests, the Region Connection Calculus studies the different relations that can be defined between regions in some topological space; these relations are based on the primitive relation of connection. As an example of a

relation of the Region Connection Calculus, the relation *disconnected* between two regions x and y suggests that none of the points of region x connects with a point of region y, and vice versa. Two of its fragments, namely, RCC-8 and RCC-5 (a sublanguage of RCC-8 where no significance is attached to boundaries of regions), have been used in several real-life applications. In particular, Bouzy in [Bouzy, 2001] used the RCC-8 qualitative constraint language in programming the Go game, and Andreas et al. in [Lattner *et al.*, 2005] used RCC-5 to set up assistance systems in intelligent vehicles. Other typical applications of the Region Connection Calculus involve GIS, robot navigation, high level vision, and natural language processing [Bhatt *et al.*, 2011].

In the literature, there have also been efforts towards combining space and time in an interrelated manner and, consequently, forming spatio-temporal formalisms. With such qualitative spatio-temporal frameworks, we can represent for example the fact that a given region was contained in another region at one point in time and externally connected to that region at a next point of time, or even the fact that a point will always move towards a particular direction over time. Towards constraint-based qualitative spatio-temporal reasoning, most of the work has relied on formalisms based on the propositional temporal logic (PTL), also known as linear temporal logic, and some qualitative spatial constraint language, like the RCC-8 language we referenced earlier (cf. [Wolter and Zakharyaschev, 2003; Wolter and Zakharyaschev, 2000b]). PTL [Huth and Ryan, 2004] is the well known temporal logic comprising operators $\mathcal{U}$ (until), $\bigcirc$ (next point in time), $\Box$ (always), and $\Diamond$ (eventually) over various flows in time, such as $\langle \mathbb{N}, < \rangle$. Other spatio-temporal reasoning frameworks consider temporal sequences of spatial QCNs. These sequences allow one to describe a spatial configuration that evolves and changes over time. Indeed, solving the spatial QCNs in such a given sequence will in turn yield a sequence of scenarios constituting a timeline, upon which the different states of a qualitative spatial configuration that evolves over time can be viewed. An example of a published work on spatio-temporal sequences is the work of Westphal et al. in [Westphal *et al.*, 2013]. Another natural approach to obtain a spatio-temporal formalism is to temporalize the RCC-8 language using the Interval Algebra language. The computational complexity of that approach has been studied in [Gerevini and Nebel, 2002]. Dealing with a qualitative spatio-temporal formalism gives rise to various interesing problems, such as the ones related to periodicity and recurring patterns that can appear in temporalized QCNs.

In the context of our thesis, we push the envelope in the field of qualitative spatial and temporal reasoning by making contributions with respect to several of its key aspects. In particular, given a knowledge base of qualitative spatial or temporal information, we define novel local consistency conditions and related techniques to efficiently solve the fundamental reasoning problems that are associated with such knowledge bases. These reasoning problems consist of the *satisfiability problem*, which is the problem of deciding whether there exists a quantitative interpretation of all the entities of a knowledge base such that all of its qualitative relations are satisfied by that interpretation (such an interpretation being called a *solution*), the *minimal labeling problem*, which is the problem of determining all the atoms for each of the qualitative relations of a knowledge base that participate in at least one of its solutions, and the *redundancy problem*, which is the problem of obtaining all the non-redundant qualitative relations of a knowledge base, i.e., those qualitative relations that do not contain at least one atom participating in a solution of the modified knowledge base that results by removing these qualitative relations. Further, we enrich the field of spatio-temporal formalisms that combine space and time in an interrelated manner by making contributions with respect to a qualitative spatio-temporal logic that results by combining the propositional temporal logic (PTL) with a qualitative spatial constraint language, and by investigating the task of ordering a temporal sequence of qualitative spatial configurations to meet certain transition constraints. Regarding the spatio-temporal logic, we also present a first semantic tableau method that given a formula ϕ of that logic systematically searches for a model

for ϕ.

Organization of the Thesis

In Chapter 2, we make an overview of some well-known qualitative constraint languages for reasoning about time and space and we describe in detail the different elements that constitute such formalisms. In particular, we discuss the notion of the base relations, which are used to represent particular qualitative configurations between spatial or temporal entities and are therefore in the heart of any qualitative constraint-based formalism. Further, we delve into the relational operations that are defined in order to be able to reason with a qualitative constraint language. Moreover, we focus on qualitative constraint languages based on points, intervals, blocks, or regions, and we also go through some aspects of orientation and the means by which it can be handled.

In Chapter 3, we formally introduce the notion of a qualitative constraint network (QCN), and draw the connection between the base relations and the relational operations we presented in Chapter 2 and some useful local consistency conditions for characterizing QCNs. In short, a QCN comprises a set of variables corresponding to a set of spatial or temporal entities and a set of relations that constrain the possible qualitative configurations between the different entities. What is more, and with regard to a given QCN, we present the fundamental reasoning problems that are associated with it, as well as the state of the art algorithms for dealing with those reasoning problems. Specifically, these problems consist of the *satisfiability problem*, that is, the problem of deciding whether there exists an interpretation of all the variables of the QCN such that all of its constraints are satisfied by this interpretation (such an interpretation being called a *solution*), and the closely related *minimal labeling problem* and *redundancy problem*. The *minimal labeling problem* is the problem of determining all the base relations for each of the constraints of a QCN that participate in at least one of its solutions, whilst the redundancy problem is the problem of obtaining all the constraints of a QCN that do not contain at least one base relation participating in a solution of the modified QCN that results by removing these constraints. Further, we explain some constraint properties of QCNs, and finally make a discussion on some decomposability aspects of QCNs considered in the literature.

In Chapter 4, we review the state of the art frameworks that combine space and time in an interrelated manner. Towards constraint-based qualitative spatio-temporal reasoning, most of the work has relied on formalisms based on the propositional temporal logic (PTL), also known as linear temporal logic, and some qualitative spatial constraint language, like RCC-8 [Wolter and Zakharyaschev, 2003; Wolter and Zakharyaschev, 2000b]. PTL [Huth and Ryan, 2004] is the well known temporal logic comprising operators $\mathcal{U}$ (until), $\bigcirc$ (next point in time), $\Box$ (always), and $\Diamond$ (eventually) over various flows in time, such as $\langle \mathbb{N}, < \rangle$. Other spatio-temporal reasoning frameworks consider temporal sequences of spatial QCNs. These sequences allow one to describe a spatial configuration that evolves and changes over time. Indeed, solving the spatial QCNs in such a given sequence will in turn yield a sequence of scenarios constituting a timeline, upon which the different states of a qualitative spatial configuration that evolves over time can be viewed. We also explore a qualitative constraint-based spatio-temporal formalism that results from combining the temporal language of Interval Algebra with the spatial one of RCC-8, and we close that chapter with a discussion on the notions of periodicity and recurring patterns that can appear in temporalized QCNs.

In Chapter 5, we present our contributions in the context of qualitative constraint-based spatial and temporal reasoning. These contributions involve novel and efficient algorithms that go beyond the state of the art algorithms for reasoning with qualitative constraint networks (QCNs).

In particular, we define new local consistency conditions and new algorithms for enforcing those conditions, which we compare both theoretically and experimentally to the local consistency conditions and their respective algorithms that were presented in Section 3.5 of Chapter 3. Our contributions range over the entire spectrum of fundamental reasoning problems in qualitative constraint-based spatial and temporal reasoning. Specifically, we demonstrate both in theory and in practice how the *satisfiability problem*, the *minimal labeling problem*, and the *redundancy problem* of a given QCN can be dealt with efficiently through the use of our novel techniques. Furthermore, we address an issue in the literature regarding a non-sound approach that utilizes parallelism to check the satisfiability of RCC-8 networks. To this end, we provide the appropriate fixes for that approach, but also present our own approach of a simple decomposition scheme that exploits the sparse and loosely connected structure of the constraint graphs of very large real-world QCNs and paves the way for efficient utilization of parallelism to solve all the aforementioned fundamental reasoning tasks. The contributions to be presented in that chapter draw from the published works in [Sioutis, 2014; Amaneddine *et al.*, 2013; Sioutis and Condotta, 2014b; Sioutis *et al.*, 2015i; Sioutis *et al.*, 2015h; Sioutis *et al.*, 2015f; Sioutis and Condotta, 2014c; Sioutis and Condotta, 2014a; Sioutis *et al.*, 2015g; Sioutis *et al.*, 2016b; Sioutis *et al.*, 2016a; Sioutis *et al.*, 2016c]. Finally, we conclude that chapter and give some directions for future work both in the field of qualitative constraint-based spatial and temporal reasoning and in the field of quantitative constraint-based spatial and temporal reasoning. With respect to the latter field, and inspired from our contribution in the former field that handles redundancy in a QCN [Sioutis *et al.*, 2015f], we briefly discuss a contribution of ours towards dealing with redundant information in the Simple Temporal Problem (STP) [Dechter *et al.*, 1991], presented in detail in [Lee *et al.*, 2016].

In Chapter 6, we present our contributions with respect to formalisms that combine spatial and temporal reasoning in an interrelated manner. In particular, we study the qualitative spatio-temporal logic that results by combining the propositional temporal logic (PTL) with a qualitative spatial constraint language, namely, the $\mathcal{L}_\mathbf{1}$ logic that was presented in Section 4.2 of this thesis, and investigate the implication of certain flexible constraint properties in qualitative spatio-temporal reasoning. We use these properties to strengthen results regarding the complexity of the satisfiability problem in $\mathcal{L}_\mathbf{1}$, by replacing a stricter condition used in literature and, consequently, generalizing to more qualitative spatial constraint languages. Further, we identify fragments of the $\mathcal{L}_\mathbf{1}$ logic that capture significant aspects of spatio-temporal change. In particular, we address the issue of periodical, and smoothness and continuity constraints between spatial configurations, and obtain results on their computational aspects. Regarding periodicity, we strengthen related results that exist in the literature, by re-establishing conditions that allow for tractability and, again, generalizing to a larger class of qualitative spatial constraint languages. Further, we present a first semantic tableau method that given a $\mathcal{L}_\mathbf{1}$ formula ϕ systematically searches for a model for ϕ. Our approach builds on Wolper's tableau method for PTL, while the ideas provided can be carried to other tableau methods for PTL as well. We prove the correctness of our tableau method for $\mathcal{L}_\mathbf{1}$ using the aforementioned strengthened results regarding the satisfiability problem in $\mathcal{L}_\mathbf{1}$. Moreover, we investigate the task of ordering a temporal sequence of qualitative spatial configurations to meet certain transition constraints. This ordering is constrained by the use of conceptual neighbourhood graphs defined on qualitative spatial constraint languages. Specifically, we show that the problem of ordering a sequence of qualitative spatial configurations to meet such transition constraints is NP-complete for several well-known qualitative constraint languages, such as RCC-8 and Interval Algebra. Based on this result, we also propose a framework where the temporal aspect of a sequence of qualitative spatial configurations is constrained by a Point Algebra network, and again show that the enhanced problem is in NP when considering

the aforementioned languages. These results lie within the area of Graph Traversal and allow for many practical and diverse applications, such as identifying optimal routes in mobile robot navigation, modelling changes of topology in biological processes, and computing sequences of segmentation steps used in image processing algorithms. The contributions to be presented in that chapter draw from the published works in [Sioutis *et al.*, 2015b; Sioutis *et al.*, 2015e; Sioutis *et al.*, 2014; Sioutis *et al.*, 2015c; Sioutis *et al.*, 2015a; Sioutis *et al.*, 2015d]. Finally, we conclude that chapter and give some directions for future work. In particular, we discuss the implication of using determined entities (constants) for a given qualitative spatial constraint language (cf. [Li *et al.*, 2013; Liu *et al.*, 2011]) and whether qualitative spatio-temporal reasoning can benefit from a recent advancement regarding the modal logic S5 [Salhi and Sioutis, 2015].

In Chapter 7, we conclude our thesis and give some directions for future research. With respect to future research in particular, we provide directions that are in relation to the work we presented in this thesis, but also directions that lie outside the field of qualitative spatial and temporal reasoning and close to the field of graph theory.

Part I

State of the Art

Chapter 2

Qualitative Spatial and Temporal Constraint Languages

2.1 Introduction

A qualitative constraint-based formalism for reasoning about time and space considers a set of base relations that represent particular qualitative configurations between spatial or temporal entities. Using these relations, we can abstract certain reasoning tasks involving spatial or temporal constraints of numerical values, which would otherwise be very difficult or even impossible to deal with if quantitative information were to be considered in the first place. Indeed, as we will see later on, a qualitative representation of spatial or temporal information allows implementing efficient methods of reasoning.

The first qualitative constraint language was proposed by Allen in [Allen, 1981; Allen, 1983], called Interval Algebra, mostly used for reasoning about time. Allen wanted to define a framework for reasoning about time in the context of natural language processing that would be reliable and efficient enough for reasoning about extracted and newly indrocuded temporal information in a qualitative manner. Interval Algebra uses intervals on the timeline to represent entities corresponding to actions, events, or tasks. A set of particular relations, called base relations, that correspond to the possible qualitative configurations between two intervals can then be considered. For instance, the relation *precedes* is a base relation of Interval Algebra and it is typically used to denote the fact that a task is executed strictly before an other. With respect to space, the most well-known and studied spatial formalism is that of the Region Connection Calculus (RCC) [Randell *et al.*, 1992], proposed by Randell, Cui, and Cohn. As its name suggests, the Region Connection Calculus can be used to represent and reason about spatial relations between regions, which can be interpreted as subsets of some topological space. As such, the spatial relations considered are topological relations. An example of such a relation is the relation *partially overlapping*, which suggests that two regions share some interior points.

In this chapter, we make an overview of some well-known qualitative constraint languages and we describe in detail the different elements that constitute such a qualitative constraint-based formalism, namely, the notion of the base relations, which is in the heart of such formalisms, as well as the relational operations used in order to reason with them.

2.2 Base Relations of Qualitative Constraint Languages

A qualitative temporal or spatial constraint language is based on a finite set B of relations, called the set of *base relations*. These relations are defined on a domain D and have the same arity ϵ, for some integer $\epsilon > 1$. The domain D itself contains elements that correspond to spatial or temporal entities with respect to a given level of granularity. For instance, there can be defined infinitely many time points or temporal intervals in the timeline and infinitely many spatial regions in a two or three dimensional space. The base relations of the set B of a particular qualitative constraint language can be used to represent the definite knowledge between any two or more entities with respect to the given level of granularity. Every tuple of ϵ elements of D can satisfy at most one single base relation $b \in \mathsf{B}$. In fact, the base relations of set B are *jointly exhaustive and pairwise disjoint* (JEPD) relations. Formally, the base relations of set B satisfy the following properties:

- $\bigcup_{b \in \mathsf{B}} b = \overbrace{\mathsf{D} \times \ldots \times \mathsf{D}}^{\epsilon \text{ times}}$ (jointly exhaustive);
- $\forall b, b' \in \mathsf{B}$ such that $b \neq b'$, we have that $b \cap b' = \emptyset$ (pairwise disjoint).

The indefinite knowledge between any two or more entities can be represented by unions of possible base relations. We represent a union of base relations $b_1 \cup \ldots \cup b_j$, with $j \leq |\mathsf{B}|$, by the set $\{b_1, \ldots, b_j\}$ containing them. Hence, 2^{B} represents the total set of relations of the considered qualitative constraint language including the empty relation, denoted by $\emptyset$, that corresponds to a relation that does not contain any of the available base relations. Without any ambiguity, B will also denote the *universal relation*, which is the union of all the base relations of the considered qualitative constraint language. Further, we identify a relation of 2^{B} that correponds to the identity relation for D^{ϵ}, denoted by Id. For most of the qualitative constraint languages, the identity relation Id corresponds to a single base relation, i.e., $\mathsf{Id} = b$ for some $b \in \mathsf{B}$.

It will be said that ϵ elements $x_1, \ldots, x_\epsilon \in \mathsf{D}$ satisfy a base relation $b \in \mathsf{B}$, denoted by $b(x_1, \ldots, x_\epsilon)$, if and only if $(x_1, \ldots, x_\epsilon) \in b$. Likewise, for a relation $r \in 2^{\mathsf{B}}$ and a set of ϵ elements $x_1, \ldots, x_\epsilon \in \mathsf{D}$, it will be said that elements $x_1, \ldots, x_\epsilon$ satisfy relation r, denoted by $r(x_1, \ldots, x_\epsilon)$, if and only if there exists a base relation $b \in r$ such that $b(x_1, \ldots, x_\epsilon)$. In the case where an arity of $\epsilon = 2$ is considered, the infix notation might also be used. For instance, given two elements $x, y \in \mathsf{D}$, a base relation $b \in \mathsf{B}$, and a relation $r \in 2^{\mathsf{B}}$, $x\ b\ y$ and $x\ r\ y$ correspond to $b(x, y)$ and $r(x, y)$ respectively. Further, in the entirety of this thesis, we assume that $\mathsf{B} \supset \{\mathsf{Id}\}$.

In what follows in our thesis, we will mostly be concerned with binary qualitative constraint languages [Ladkin and Maddux, 1994], i.e., qualitative constraint languages based on a set of base relations with an arity $\epsilon = 2$. We will explicitly mention the arity whenever we refer to a qualitative constraint language of greater arity.

2.3 Cases of Qualitative Spatial and Temporal Constraint Languages

In this section we review some of the most well-known and dominant qualitative spatial and temporal constraint languages, such as Point Algebra, Cardinal Direction Calculus, Interval Algebra, Block Algebra, and RCC-8, that will be the focus of our thesis.

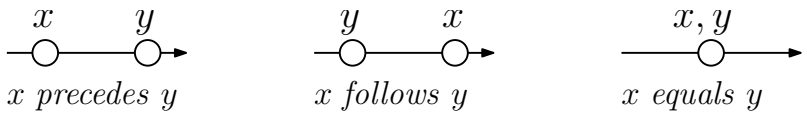


Figure 2.1: The base relations of Point Algebra

2.3.1 Point Algebra

The qualitative temporal constraint language of Point Algebra (PA) [Vilain *et al.*, 1990; van Beek and Cohen, 1990; van Beek, 1992] uses points to represent temporal entities (e.g., events) and the following three base relations to reason about the relative position of those temporal entities in the timeline: *precedes* ($<$), *equals* ($=$), and *follows* ($>$) (see Figure 2.1). These three base relations considered by Point Algebra are interpreted on a set with a linear ordering relation. In particular, considering the points on the line of rational numbers and the usual ordering relation $<$, the three base relations of Point Algebra are defined in the following manner: *precedes* $= \{(x, y) \in \mathbb{Q}\times\mathbb{Q} \mid x < y\}$, *follows* $= \{(x, y) \in \mathbb{Q}\times\mathbb{Q} \mid y < x\}$, and *equals* $= \{(x, y) \in \mathbb{Q}\times\mathbb{Q} \mid x = y\}$. Based on these three base relations, we can define eight relations of Point Algebra in total that correspond to the set $2^{\mathsf{B}} = \{\{<, =, >\}, \{<, >\}, \{<, =\}, \{=, >\}, \{<\}, \{>\}, \{=\}, \emptyset\}$. As an example, relation $\{<, >\}$ allows us to represent the knowledge that an event occurs before or after another event, but not at the same time. Further, two events $x, y \in \mathbb{Q}$ satisfy relation $\{<, >\}$ if and only if $x \neq y$.

2.3.2 Cardinal Direction Calculus

The Cardinal Direction Calculus (CDC) [Ligozat, 1998; Frank, 1991] is a qualitative constraint language with a spatial aspect and can be seen as an extension of the qualitative constraint language of Point Algebra discussed earlier. The entities of the domain D are points in the Euclidean plane and are equipped with an orthogonal reference. In particular, the relative position between two entities is determined by the Point Algebra base relations that are derived from projections of those points on the two axes. As such, we obtain nine possible base relations between two given entities x and y in Cardinal Direction Calculus, namely, east (E), north (N), south (S), west (W), northeast (NE), northwest (NW), southwest (SW), southeast (SE), and equals (EQ). These base relations can be viewed in Figure 2.2.

2.3.3 Interval Algebra

The qualitative constraint language of Interval Algebra (IA) [Allen, 1981; Allen, 1983] is one of the most well-known qualitative constraint formalisms, primarily because it is the first one to be

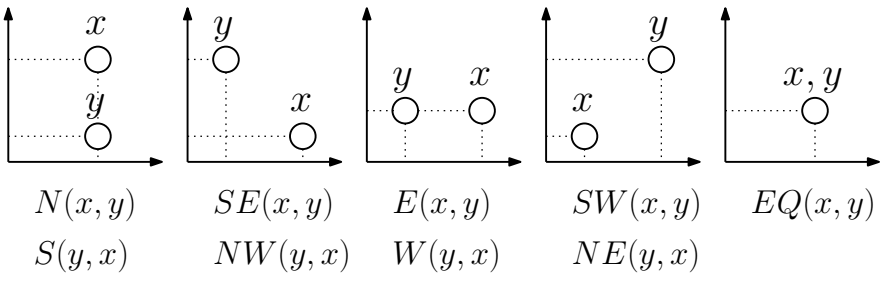


Figure 2.2: The base relations of Cardinal Direction Calculus

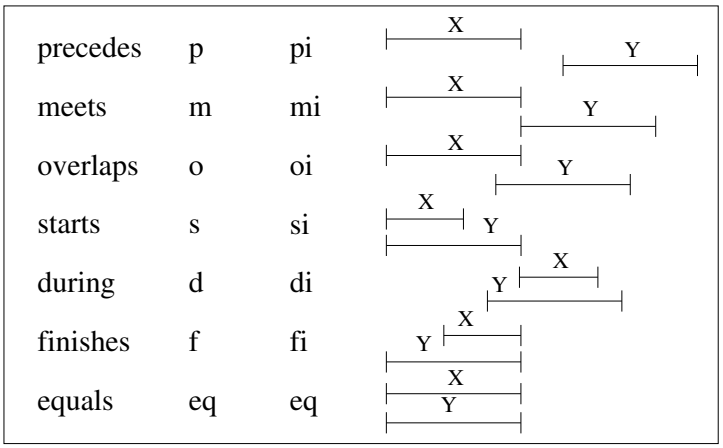


Figure 2.3: The base relations of Interval Algebra

proposed and studied in the literature and has been used to represent and reason with temporal information in applications in planning and scheduling [Allen and Koomen, 1983; Allen, 1991; Pelavin and Allen, 1987; Dorn, 1995], in natural language processing [Song and Cohen, 1988], in temporal databases [Snodgrass, 1987; Chen and Zaniolo, 1998], in multimedia databases [Little and Ghafoor, 1993], in molecular biology [Golumbic and Shamir, 1993] (e.g., arrangement of DNA segments/intervals along a linear chain involves particular temporal-like problems [Benzer, 1959]), and in workflow [Lu *et al.*, 2006]. Entities in Interval Algebra correspond to events or actions and are represented by intervals in the timeline. In particular, Interval Algebra considers the following thirteen base relations: *equals* ($=$), *precedes* (p), *precedes inverse* (pi), *meets* (m), *meets inverse* (mi), *overlaps* (o), *overlaps inverse* (oi), *starts* (s), *starts inverse* (si), *during* (d), *during inverse* (di), *finishes* (f), and *finishes inverse* (fi) (see Figure 2.3). Each of the aforementioned base relations corresponds to a particular configuration of the four endpoints of the two intervals (we have two endpoints for each interval) and allow representing a relative position between two given temporal entities. As an example, given two temporal intervals (or, equivalently, two Interval Algebra entities) x and y such that the base relation x d y holds, we have that both endpoints of the temporal interval x are properly contained in the interval that is defined by the two endpoints of the temporal interval y. In this particular case, the base relation *during* (d) can describe a situation where a temporal activity is executed within the execution time window of another temporal activity. Note that Interval Algebra can be interpreted as a spatial calculus as well. For instance, the base relation *during* can describe a situation where a line segment is part of a bigger line segment, hence, certain topological relationships can be represented as well.

At this point, let us formally introduce the domain and the base relations of Interval Algebra. The domain D of Interval Algebra is defined to be the set of intervals on the line of rational numbers (with the usual ordering relation $<$ being used for ordering the corresponding endpoints), i.e., $\mathsf{D} = \{x = (x^-, x^+) \in \mathbb{Q} \times \mathbb{Q} \mid x^- < x^+\}$. Then, each base relation can be defined by appropriately constraining the endpoints of the two temporal intervals at hand. Thus, the base relation *during* that we saw earlier, is defined as $during = \{(x, y) \in \mathsf{D} \times \mathsf{D} \mid x^- > y^- \text{ and } x^+ < y^+\}$. As another example, the relations *starts* is defined as $starts = = \{(x, y) \in \mathsf{D} \times \mathsf{D} \mid x^- = y^- \text{ and } x^+ < y^+\}$. Note that in the first related published work with respect to Interval Algebra, Allen considers only nine base relations, as he groups the base relations d, s, f (and the inverses of those base relations di, si, fi) together into a unique base relation (a unique inverse base relation

respectively).

A particular extension of Interval Algebra considers the case where the entities can be either intervals (as defined earlier) or points. As such, we can have qualitative relations between two intervals as in Interval Algebra, qualitative relations between two points as in Point Algebra, but also qualitative relations between an interval and a point in a unique calculus [Vilain, 1982]. Concerning this extension, Vilain comments: *"We should state that including points along with intervals in the domain of our system only minimally complicates the deduction algorithms. The polynomial complexity results and the consistency maintenance remain unaffected"*. In other words, allowing basic time objects to be either time intervals or time points does not alter in a significant way the framework of Interval Algebra.

2.3.4 Block Algebra

The qualitative constraint language of Block Algebra (BA) [Balbiani *et al.*, 1998; Balbiani *et al.*, 2002] is a natural generalization of the Interval Algebra constraint language to the n-dimensional Euclidean space with an orthogonal basis. The entities of Block Algebra are called *n-blocks*, or simply *blocks* when there is no confusion regarding the considered dimension. These blocks have their sides parallel to the reference axes. Every base relation of this qualitative constaint language is binary and characterized by an n-tuple of Interval Algebra base relations. The i^{th} element of such a tuple corresponds to the base relation that is satisfied by the intervals that result from the orthogonal projections of two given blocks on the i^{th} axis. As an example, let us consider the base relation (o, mi, s) that is satisfied by the two 3-blocks depicted in Figure 2.4. Indeed, if we project the two blocks on the x axis we get two intervals that satisfy the Interval Algebra base relation *overlaps*, if we project the two blocks on the y axis we get two intervals that satisfy the Interval Algebra base relation *meets inverse*, and, finally, if we project the two blocks on the z axis we get two intervals that satisfy the Interval Algebra base relation *starts*. Clearly, if we consider n-blocks, we will obtain 13^n base relations of Block Algebra. It is worth noting that the particular Block Algebra calculus of 1-blocks corresponds to the Interval Algebra calculus, and the particular Block Algebra calculus of 2-blocks to the 2-Block Algebra calculus, studied in [Balbiani *et al.*, 1998].

Another natural generalization of the Interval Algebra constraint language can be obtained by considering a different, more general, type of entities. In particular, an interval in the context

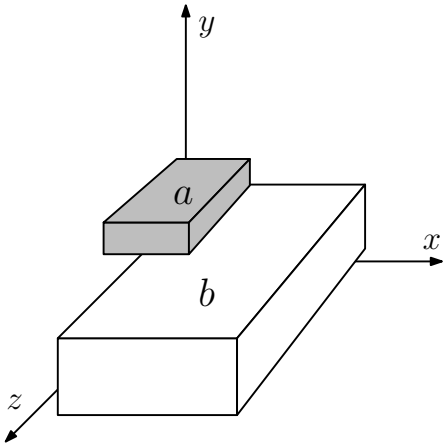


Figure 2.4: The base relation (o, mi, s) of Block Algebra

of Interval Algebra is just a pair of ordered time points. We can extend somewhat further this notion and define a *generalized interval* as an ordered, finite sequence of points in a linear order. We also call a generalized interval with n-points an n-interval [Ligozat, 1991]. More generally, for any subset Z of the set of integers $\mathbb{Z}$, a $\mathbb{Z}$-interval is a n-interval where n belongs to Z. In this way, Interval Algebra is the calculus of 2-intervals. This notion of generalized intervals is close to the notion of non-convex intervals studied in [Ladkin, 1986]. In particular, the calculus of non-convex intervals presented in [Ladkin, 1986] is the calculus of e-intervals, where e is the set of even integers [Ligozat, 1991]. The qualitative relation between two generalized intervals is determined by the Interval Algebra base relations that are satisfied between each pair of intervals comprising the respective generalized intervals. The interested reader may also see the works presented in [Balbiani *et al.*, 2000; Condotta, 2004], where a complete study of Block Algebra and the calculus of generalized intervals, as well as a comparison between those two formalisms, takes place.

2.3.5 RCC-8

Topology is perhaps the most fundamental aspect of space and certainly one that has been studied extensively within the mathematical literature. The Region Connection Calculus (RCC) is the dominant topological approach to represent and reason about topological relations [Randell *et al.*, 1992]. RCC abstractly describes regions that are non-empty regular closed subsets of some topological space that do not have to be internally connected and do not have a particular dimension, by their possible relations to each other. In particular, for any spatial region X in RCC we have that $X = c(i(X))$, where $i(\cdot)$ specifies the topological interior of a spatial region and $c(\cdot)$ the topological closure [Renz, 2002a]. RCC is based on a single primitive relation between spatial regions, relation C [Clarke, 1981]. The intended topological interpretation of

Table 2.1: Definition of the various relations of RCC; relations in bold are included in RCC-8

Relation	**Description**	**Definition**
$C(x, y)$	connects with	primitive relation
$\mathbf{DC}(x, y)$	disconnected	$\neg C(x, y)$
$P(x, y)$	part	$\forall z[C(z, x) \rightarrow C(z, y)]$
$PP(x, y)$	proper part	$P(x, y) \wedge \neg P(y, x)$
$\mathbf{EQ}(x, y)$	equals	$P(x, y) \wedge P(y, x)$
$O(x, y)$	overlaps	$\exists z[P(z, x) \wedge P(z, y)]$
$\mathbf{PO}(x, y)$	partially overlaps	$O(x, y) \wedge \neg P(x, y) \wedge \neg P(y, x)$
$DR(x, y)$	discrete	$\neg O(x, y)$
$\mathbf{TPP}(x, y)$	tangential proper part	$PP(x, y) \wedge \exists z[EC(z, x) \wedge EC(z, y)]$
$\mathbf{EC}(x, y)$	externally connected	$C(x, y) \wedge \neg O(x, y)$
$\mathbf{NTPP}(x, y)$	non-tangential proper part	$PP(x, y) \wedge \neg\exists z[EC(z, x) \wedge EC(z, y)]$
$Pi(x, y)$	part inverse	$P(y, x)$
$PPi(x, y)$	proper part inverse	$PP(y, x)$
$\mathbf{TPPi}(x, y)$	tangential proper part inverse	$TPP(y, x)$
$\mathbf{NTPPi}(x, y)$	non-tangential proper part inverse	$NTPP(y, x)$

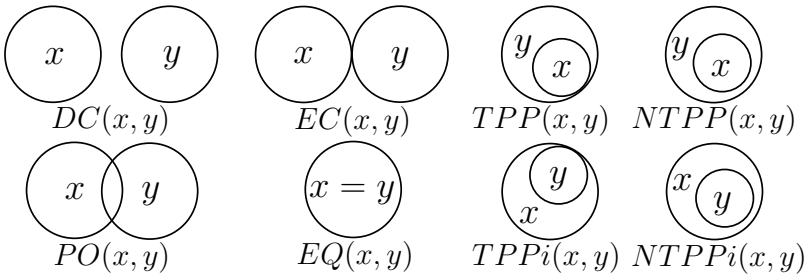


Figure 2.5: The base relations of RCC-8

$C(a, b)$, where a and b are spatial regions, is that a and b are connected if and only if their topological closures share a common point. This primitive can be used to define many predicates and functions which capture interesting and useful topological distinctions [Gotts, 1994; Gotts, 1996]. Table 2.1 shows the definitions of the RCC relations with respect to the primitive relation C. Of particular importance are those relations that form a set of jointly exhaustive and pairwise disjoint relations, viz., a set of base relations as specified in Section 2.2. The set of base relations of RCC are the following: *disconnected* (DC), *externally connected* (EC), *equals* (EQ), *partially overlapping* (PO), *tangential proper part* (TPP), *tangential proper part inverse* ($TPPi$), *non-tangential proper part* ($NTPP$), and *non-tangential proper part inverse* ($NTPPi$). These eight base relations form the RCC-8 constraint language and are depicted in Figure 2.5 (for the 2-dimensional case).

2.3.6 9-intersection model (9-IM)

Another approach to represent and reason about topological relations is based on the 9-*intersection* model. In the 9-intersection model each spatial region is characterized by 3 sets of points, each of which holds information about its interior ($\circ$), its boundary (∂), and its exterior ($-$). Thus, all relations between two spatial regions are defined by a 3×3 matrix, in which every entry takes one of two values, denoting whether the intersection of two point sets for two spatial regions is empty or not. The 3×3 matrix for a relation R between two regions A and B is shown below:

$$R(A, B) = \begin{pmatrix} A^\circ \cap B^\circ & A^\circ \cap \partial B & A^\circ \cap B^- \\ \partial A \cap B^\circ & \partial A \cap \partial B & \partial A \cap B^- \\ A^- \cap B^\circ & A^- \cap \partial B & A^- \cap B^- \end{pmatrix}$$

The 3×3 matrix would normally yield exactly 2^9 spatial relations, but taking into account the physical reality of 2-dimensional space and some specific assumptions about the nature of regions, viz., that they are 2-dimensional, internally connected (one-piece), without holes, and only emptiness or non-emptiness of the intersection is taken into account, it turns out that the there are exactly 8 remaining matrices, which correspond to the RCC-8 relations. A geometric interpretation of the 8 relations between two regions with connected boundaries is shown in Figure 2.6. These relations are also often mentioned as Egenhofer relations [Egenhofer and Herring, 1991]. By definition, Egenhofer places stronger constraints on the domain of regions than RCC-8, as we saw earlier that RCC-8 considers a more general domain, viz., regions that are non-empty regular closed subsets of some topological space that do not have to be internally connected and do not have a particular dimension. This fact makes reasoning in the 9-intersection model more complex than reasoning with RCC-8 [Renz and Nebel, 2007; Grigni *et al.*, 1995]. Further, to reason about regions with holes, one would have to define relations not only between

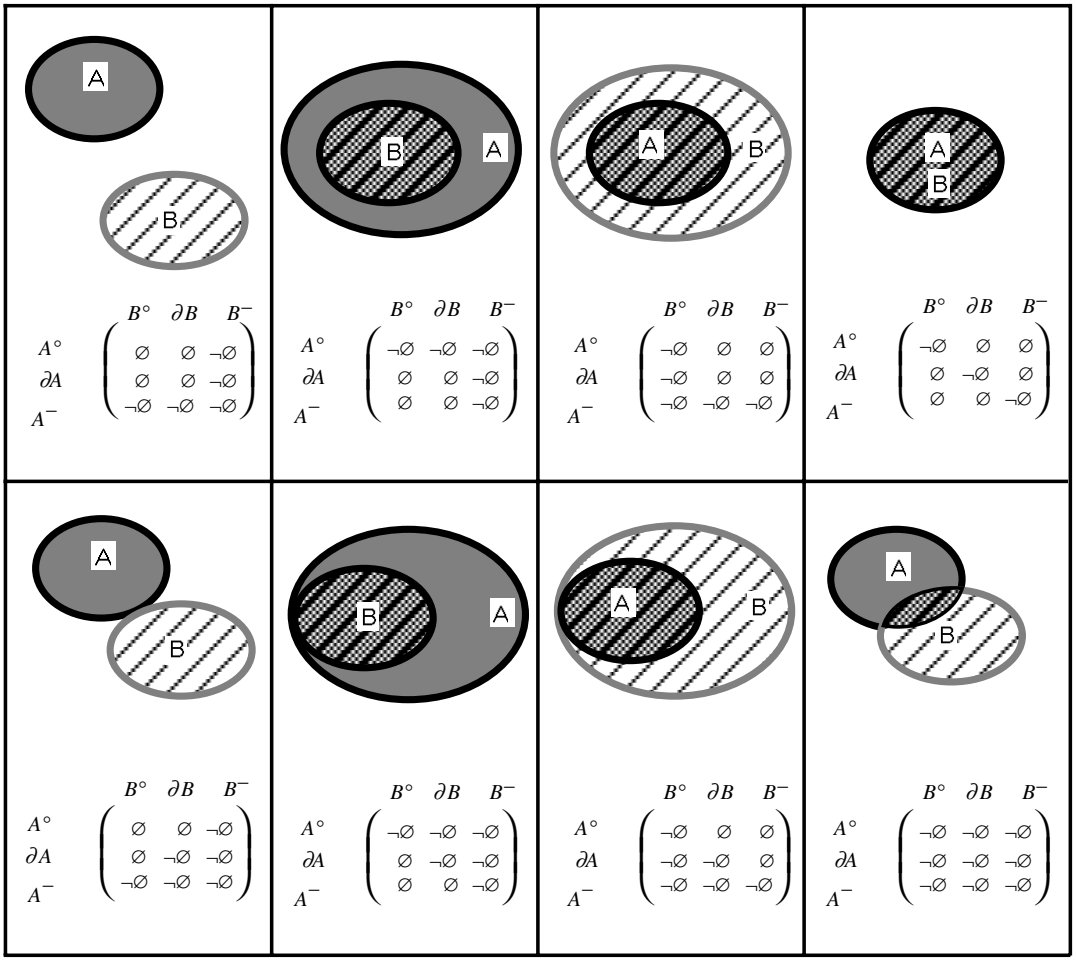


Figure 2.6: A geometric interpretation of the 8 relations between two regions with connected boundaries (symbol ¬ stands for *not*, hence, $\neg\varnothing$ is a non-empty set)

each pair of regions, but also between each hole of each region and the other region and each of its holes, adding to the computational complexity of the reasoning proccess [Egenhofer *et al.*, 1994].

As we explained earlier, Egenhofer relations are the derived relations of the 9-intersection model when considering specific assumptions about the nature of a spatial region. Other assumptions can be made to derive new sets of relations. Clementini took into account the codimension of intersection, extending the representation in each matrix cell by the dimension of the intersection rather than simply specifying whether the intersection is empty or not [Clementini and Di Felice, 1995]. The dimensionally extended 9-intersection model (DE-9IM), is commonly refered to as the Clementini intersection pattern matrix [Clementini *et al.*, 1994]. The Clementini intersection pattern matrix for a relation R between two regions A and B is shown below:

$$R(A,B) = \begin{pmatrix} dim(A^\circ \cap B^\circ) & dim(A^\circ \cap \partial B) & dim(A^\circ \cap B^-) \\ dim(\partial A \cap B^\circ) & dim(\partial A \cap \partial B) & dim(\partial A \cap B^-) \\ dim(A^- \cap B^\circ) & dim(A^- \cap \partial B) & dim(A^- \cap B^-) \end{pmatrix}$$

where *dim* is the maximum number of dimensions of the intersection ($\cap$) of the interior ($\circ$),

boundary (∂), and exterior ($-$) of regions A and B. The maximum number of dimensions of the intersection is 0 for points, 1 for lines, and 2 for areas. The interested reader may find more information about possible relations between areas, lines and points, and their respective calculi that can be defined in the dimensionally extended 9-intersection model in [Clementini *et al.*, 1993].

2.3.7 Orientation Calculi

Orientation relations describe the orientation between two spatial entities with respect to a third object. As such, a binary relation between a primary spatial entity and a reference spatial entity is, in general, not sufficient to describe the orientation between those entities, since some kind of *frame of reference* must also be considered [Cohn, 1997]; after doing so, one is then able to define an explicit ternary relation upon the primary spatial entity, the reference spatial entity, and the frame of reference. Many orientation constraint languages are defined using this approach [Freksa, 1992; Isli and Cohn, 2000; Schlieder, 1993; Röhrig, 1994], while others presuppose an immutable extrinsic frame of reference (e.g., gravitation, a fixed coordinate system) as in the calculus of dinstances and cardinal directions presented in [Frank, 1992] (also cf. [Hernández, 1994]) or a natural intrinsic orientation that is exhbitited by certain spatial entities (e.g., humans, buildings) as in the $\mathcal{OPRA}_m$ calculus presented in [Moratz *et al.*, 2005; Mossakowski and Moratz, 2012; Moratz, 2006].

Regarding orientation constraint languages based on explicit ternary relations, of particular interest is the approach of Schlieder where triples of points are mapped to one of the three qualitative values $+$, 0, and $-$, denoting anticlockwise, colinear, and clockwise orientations respectively [Schlieder, 1993]. The approach of Schlieder can be used for reasoning about visible locations in qualitative navigation tasks [Schlieder, 1993], for shape description [Schlieder, 1996], or to develop a qualitative constraint language for reasoning about the relative orientation of pairs of line segments [Schlieder, 1995]. Another important ternary orientation constraint language is that of Röhrig [Röhrig, 1994; Röhrig, 1993] which is based on the ternary relation $CYCORD(x, y, z)$ that is true in the 2-dimensional space when x, y, and z are in clockwise orientation. Röhrig also demonstrates how some qualitative constraint languages can be interpreted using the $CYCORD(x, y, z)$ relation and, thus, exploit his reasoning system that builds on that relation [Röhrig, 1993].

Regarding orientation constraint languages based on an extrinsic frame of reference, it is most common to use some global reference direction, which allows the orientation between two objects to be represented with respect to the reference direction using just binary relations (e.g., compass directions as in [Frank, 1992]). Such approaches were later generalised to form the Star algebra [Renz and Mitra, 2004; Mitra, 2004; Mitra, 2002].

Regarding orientation constraint languages based on a natural intrinsic orientation, the $\mathcal{OPRA}_m$ calculus [Moratz *et al.*, 2005; Mossakowski and Moratz, 2012; Moratz, 2006] is among the most seasoned qualitative constraint languages for reasoning about qualitative relative direction information. In $\mathcal{OPRA}_m$, oriented points, i.e., pairs of a point and a direction on the 2-dimensional plane, serve as the basic entities since they are the simplest spatial entities that have an intrinsic orientation. Sets of base relations can have adjustable granularity levels in this calculus. Further, $\mathcal{OPRA}_m$ offers simple geometric rules for computing the calculus's compositions based on triples of oriented points.

2.4 Relational Operations

Given a set of base relations B of a qualitative constraint language, we have that B is closed under the converse operation ($^{-1}$). In particular, the converse operation associates a relation $r^{-1} \in 2^{\mathsf{B}}$ with each base relation $b \in \mathsf{B}$, which is defined by $r^{-1} = \{b' \in \mathsf{B} \mid \exists x, y \in \mathsf{D}$ with $x\ b\ y$ and $y\ b'\ x\}$. The converse operation can be generalized to the total set of relations 2^{B} as follows. For each relation $r \in 2^{\mathsf{B}}$, r^{-1} is defined by $r^{-1} = \bigcup_{b \in r} b^{-1}$. For every $x, y \in \mathsf{D}$ and $r \in 2^{\mathsf{B}}$, it also holds that $y\ r^{-1}\ x$ if $x\ r\ y$. The inverse is not necessarily true in the general case [Dylla *et al.*, 2013]. For most of the qualitative constraint languages, there exists for each base relation $b \in \mathsf{B}$ a single base relation of B that corresponds to the converse of b, viz., the relation $\{(y, x) \mid (x, y) \in b\}$. For these languages, the converse relation of any base relation $b \in \mathsf{B}$ is b^{-1}, which is a base relation of B. Moreover, for every $x, y \in \mathsf{D}$ and $r \in 2^{\mathsf{B}}$ we have that $x\ r\ y$ if and only if $y\ r^{-1}\ x$. Examples of qualitative constraint languages for which the converse of a base relation corresponds to a relation comprising more than one base relations are the Cardinal Direction (Relations) Calculus (CDR) [Skiadopoulos and Koubarakis, 2005] and its recently introduced rectangular variant (RDR) [Navarrete *et al.*, 2013]. The converse of a relation 2^{B} can be obtained from a converse table which stores the converse base relation b^{-1} for each base relation $b \in \mathsf{B}$. The converse tables of some of the most well-known qualitative constraints languages that we presented earlier are given in Table 2.2. Given for example the RCC-8 relation $\{DC, TPPi, NTPP\}$ we can easily derive its converse relation, viz., relation $\{DC, TPP, NTPPi\}$, thus, $\{DC, TPPi, NTPP\}^{-1} = \{DC, TPP, NTPPi\}$.

Appart from the converse operation ($^{-1}$), 2^{B} is also equipped with the usual set-theoretic operations, viz., union and intersection, and the weak composition operation denoted by symbol $\diamond$ [Renz and Ligozat, 2005]. Given two relations $r, r' \in 2^{\mathsf{B}}$, $r \cup r'$ corresponds to a relation of 2^{B} that comprises the base relations of B that exist in either r or r'. In a similar manner, $r \cap r'$ corresponds to a relation of 2^{B} that comprises the base relations of B that exist in both r and r'. The weak composition operation is a bit more complex to define than the relational operations we discussed so far. Let us first recall the relational composition which is defined by $b \circ b' = \{(x, y) | \exists z : (x, z) \in b \wedge (z, y) \in b'\}$ for two base relations $b, b' \in \mathsf{B}$. According to the definition of relational composition, we have to look at an infinite number of tuples in order to

Table 2.2: Converse tables for (a) Point Algebra, (b) Cardinal Direction Calculus, (c) Interval Algebra, and (d) RCC-8

(a)

b	$\mathbf{b^{-1}}$
$<$	$>$
$>$	$<$
$=$	$=$

(b)

b	$\mathbf{b^{-1}}$
N	S
NW	SE
W	E
SW	NE
S	N
SE	NW
E	W
NE	SW
EQ	EQ

(c)

b	$\mathbf{b^{-1}}$
b	bi
bi	b
o	oi
oi	o
m	mi
mi	m
d	di
di	d
si	s
s	si
f	fi
fi	f
eq	eq

(d)

b	$\mathbf{b^{-1}}$
DC	DC
EC	EC
PO	PO
TPP	$TPPi$
$TPPi$	TPP
$NTPP$	$NTPPi$
$NTPPi$	$NTPP$
EQ	EQ

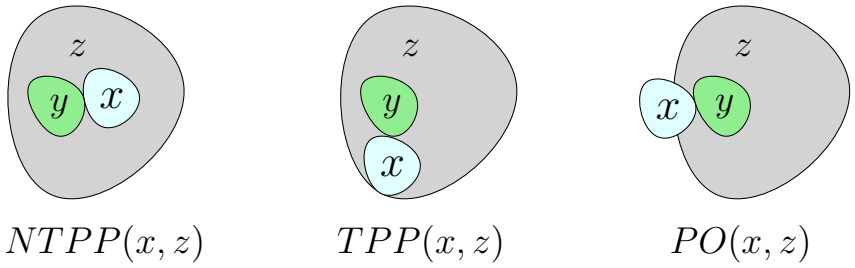


Figure 2.7: Three possible configurations for regions x, y, z when we have $EC(x, y)$ and $NTPP(y, z)$

compute the composition of base relations, which is clearly not feasible. Fortunately, many domains such as points or intervals in a timeline are ordered or, otherwise, well-structured domains and composition can be computed using the semantics of the relations. However, for domains such as arbitrary spatial regions that are more vague and where there is no common representation for the entities we consider, computing the true composition is not feasible and composition has to be approximated by using a weaker variant of it, called *weak composition* [Renz and Ligozat, 2005]. Formally, the weak composition ($\diamond$) of two base relations $b, b' \in \mathsf{B}$ is defined as the strongest relation $r \in 2^{\mathsf{B}}$ which contains $b \circ b'$, or formally, $b \diamond b' = \{b'' \in \mathsf{B} | b'' \cap (b \circ b') \neq \emptyset\}$. The advantage of weak composition is that we stay within the given set of relations 2^{B} while applying the algebraic operations, as 2^{B} is by definition closed under weak composition, union, intersection, and converse.

As an example, let us consider the RCC-8 base relations of EC and $NTPP$. The weak composition $EC \diamond NTPP$ yields the set of base relations $\{NTPP, TPP, PO\}$, as shown in Figure 2.7. The obtained set corresponds to all the base relations that can be possible between regions x and z when we have that $EC(x, y)$ and $NTPP(y, z)$. Let us now also see why RCC-8, for instance, has only weak composition and not relational composition. Consider the RCC-8 set of constraints $\{TPP(x, y), EC(x, z), TPP(z, y)\}$. It can be verified that $TPP \in EC \diamond TPP$, $EC \in TPP \diamond TPP^{-1}$, and $TPP \in EC^{-1} \diamond TPP$. As such, assuming that relational composition holds, we should be able to extract a valid configuration using and starting with any of the infinite available tuples that satisfy relation TPP or EC. However, if we pick a tuple (x, y) for satisfying relation TPP such that y is instantiated as a region with two disconnected pieces and x completely fills one piece, then z cannot be instiantiated. So, $TPP \notin EC \circ TPP$ and, consequently, relational composition does not hold for RCC-8.

As noted earlier, well-structured domains such as points or intervals in a timeline are ordered and composition can be computed using the semantics of the relations. For example, for Point Algebra and Interval Algebra, with their usual domains based on $\mathbb{Q}$ as presented earlier, weak composition is equivalent to relational composition, i.e., we have that $r \diamond r' = r \circ r'$ for every $r, r' \in 2^{\mathsf{B}}$. For qualitative constraint languages for which relational composition does not hold, we have that $r \circ r' \subset r \diamond r'$ for some $r, r' \in 2^{\mathsf{B}}$, as we also demonstrated in the case of RCC-8. A complete analysis on the relations of RCC-8 for which relational composition does not hold is provided in [Li and Wang, 2006]. The weak composition operation can be generalized to the total set of relations 2^{B} as follows. For every $r, r' \in 2^{\mathsf{B}}$, we have that $r \diamond r' = \bigcup_{b \in r, b' \in r'} b \diamond b'$. The weak composition of two relations $r, r' \in 2^{\mathsf{B}}$ can be facilitated by a weak composition table which stores the weak compositions among all base relations of the considered qualitative constraint language. The weak composition tables of Point Algebra and RCC-8 are given in Table 2.3. Given for example the RCC-8 relations $\{TPP, NTPP\}$ and $\{TPP\}$ we can easily

Table 2.3: Weak composition tables for (a) Point Algebra and (b) RCC-8

(a)

$\diamond$	$<$	$>$	$=$
$<$	$<$	B	$<$
$>$	B	$>$	$>$
$=$	$<$	$>$	$=$

(b)

$\diamond$	**DC**	**EC**	**PO**	**TPP**	**NTPP**	**TPPi**	**NTPPi**	**EQ**
DC	B	*DC*, *EC*, *PO*, *TPP*, *NTPP*	*DC*, *EC*, *PO*, *TPP*, *NTPP*	*DC*, *EC*, *PO*, *TPP*, *NTPP*	*DC*, *EC*, *PO*, *TPP*, *NTPP*	*DC*	*DC*	*DC*
EC	*DC*, *EC*, *PO*, *TPPi*, *NTPPi*	*DC*, *EC*, *PO*, *TPP*, *TPPi*, *EQ*	*DC*, *EC*, *PO*, *TPP*, *NTPP*	*EC*, *PO*, *TPP*, *NTPP*	*PO*, *TPP*, *NTPP*	*DC*, *EC*	*DC*	*EC*
PO	*DC*, *EC*, *PO*, *TPPi*, *NTPPi*	*DC*, *EC*, *PO*, *TPPi*, *NTPPi*	B	*PO*, *TPP*, *NTPP*	*PO*, *TPP*, *NTPP*	*DC*, *EC*, *PO*, *TPPi*, *NTPPi*	*DC*, *EC*, *PO*, *TPPi*, *NTPPi*	*PO*
TPP	*DC*	*DC*, *EC*	*DC*, *EC*, *PO*, *TPP*, *NTPP*	*TPP*, *NTPP*	*NTPP*	*DC*, *EC*, *PO*, *TPP*, *TPPi*, *EQ*	*DC*, *EC*, *PO*, *TPPi*, *NTPPi*	*TPP*
NTPP	*DC*	*DC*	*DC*, *EC*, *PO*, *TPP*, *NTPP*	*NTPP*	*NTPP*	*DC*, *EC*, *PO*, *TPP*, *NTPP*	B	*NTPP*
TPPi	*DC*, *EC*, *PO*, *TPPi*, *NTPPi*	*EC*, *PO*, *TPPi*, *NTPPi*	*PO*, *TPPi*, *NTPPi*	*PO*, *TPP*, *TPPi*, *EQ*	*PO*, *TPP*, *NTPP*	*TPPi*, *NTPPi*	*NTPPi*	*TPPi*
NTPPi	*DC*, *EC*, *PO*, *TPPi*, *NTPPi*	*PO*, *TPPi*, *NTPPi*	*PO*, *TPPi*, *NTPPi*	*PO*, *TPPi*, *NTPPi*	*PO*, *TPP*, *NTPP*, *TPPi*, *NTPPi*, *EQ*	*NTPPi*	*NTPPi*	*NTPPi*
EQ	*DC*	*EC*	*PO*	*TPP*	*NTPP*	*TPPi*	*NTPPi*	*EQ*

derive relation $\{TPP, NTPP\}$ by performing the following operation: $\{TPP, NTPP\} \diamond \{TPP\} = (TPP \diamond TPP) \cup (NTPP \diamond TPP) = \{TPP, NTPP\} \cup \{NTPP\} = \{TPP, NTPP\}$.

The weak composition operation $\diamond$ along with the converse operation $^{-1}$, and the total set of relations 2^{B} along with the identity relation Id of a qualitative constraint language (where B is its set of base relations), form an algebraic structure $(2^{\mathsf{B}}, \mathsf{Id}, \diamond, ^{-1})$ which can correspond to a *relation algebra* for some qualitative constraint languages in the sense of Tarski [Tarski, 1941; Dylla *et al.*, 2013]. This topic has been extensively discussed in [Ligozat and Renz, 2004; Dylla *et al.*, 2013]. In [Dylla *et al.*, 2013] the authors thoroughly analyze existing qualitative constraint languages and provide a classification involving different notions of relation algebra. In fact, we

Table 2.4: Axioms for relation algebras, where $r, s, t \in 2^{\mathsf{B}}$

Axiom	**Definition**
$\cup$-commutativity	$r \cup s = s \cup r$
$\cup$-associativity	$r \cup (s \cup t) = (r \cup s) \cup t$
Huntington axiom	$\overline{\overline{r} \cup \overline{s}} \cup \overline{\overline{r} \cup s} = r$
$\diamond$-associativity	$r \diamond (s \diamond t) = (r \diamond s) \diamond t$
$\diamond$-distributivity	$(r \cup s) \diamond t = (r \diamond t) \cup (s \diamond t)$
identity law	$r \diamond \mathsf{Id} = r$
$^{-1}$-involution	$(r^{-1})^{-1} = r$
$^{-1}$-distributivity	$(r \cup s)^{-1} = r^{-1} \cup s^{-1}$
$^{-1}$-involutive distributivity	$(r \diamond s)^{-1} = s^{-1} \diamond r^{-1}$
Tarski/de Morgan axiom	$r^{-1} \diamond \overline{r \diamond s} \cup \overline{s} = \overline{s}$

have the following result:

Proposition 1 (cf. [Dylla *et al.*, 2013]) *Each of the qualitative constraint languages of Point Algebra, Cardinal Direction Calculus, Interval Algebra, Block Algebra, and* RCC-*8 is a relation algebra with the algebraic structure* $(2^{\mathsf{B}}, \mathsf{Id}, \diamond, ^{-1})$.

In what follows, for a qualitative constraint language that is a relation algebra with the algebraic structure $(2^{\mathsf{B}}, \mathsf{Id}, \diamond, ^{-1})$, we will simply say that it is a relation algebra as the algebraic structure will always be of the same format.

In our context, a relation algebra is nothing more than a qualitative constraint language that satisfies certain axioms. These axioms are listed in Table 2.4 and allow for several optimizations when designing algorithms for reasoning with qualitative constraint languages. For instance, $\diamond$-associativity ensures that if we need to compute $r \diamond s \diamond t$ for some relations $r, s, t \in 2^{\mathsf{B}}$, we can do so by choosing to compute either $r \diamond (s \diamond t)$ or $(r \diamond s) \diamond t$, i.e., we do not need to compute the operation both from left to right and from right to left.

Due to the fact that the most interesting and well-known spatial and temporal calculi are relation algebras (cf. Proposition 1), and for the sake of simplicity in the presentation of our algorithms in what follows in the thesis, we will focus on qualitative constraint languages that are relations algebras and, hence, satisfy the related axioms.

2.5 Classes of Relations

As mentioned earlier, 2^{B} is by definition closed under weak composition, union, intersection, and converse. In the context of the algorithms that are studied in this thesis, we deal with particular subsets of relations of 2^{B} that are closed under threee of the four aforementions relational operations, namely, weak composition, intersection, and converse. We call such subsets of relations subclasses of relations. Clearly, the entire set of relations 2^{B} is a class of relations by itself. Formally, we define a subclass of relations as follows.

Definition 1 *A* subclass of relations *is a subset* $\mathcal{A} \subseteq 2^{\mathsf{B}}$ *that contains the singleton relations of* 2^{B} *and* B *and is closed under converse, intersection, and weak composition.*

Given a relation r of 2^{B} and a subclass $\mathcal{A} \subseteq 2^{\mathsf{B}}$, $\mathcal{A}(r)$ will denote the smallest (with respect to the number of base relations) relation of $\mathcal{A}$ including r.

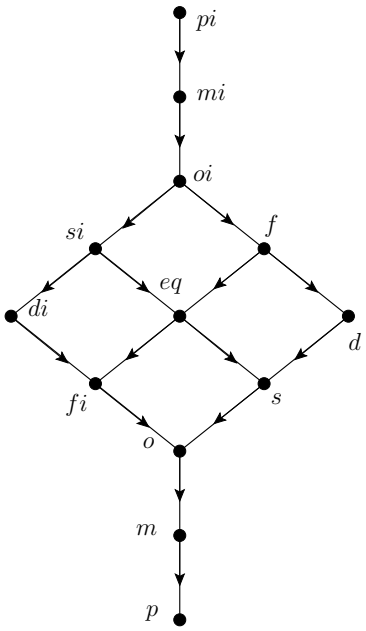


Figure 2.8: The lattice of Interval Algebra

2.5.1 Distributive Subclasses of Relations

The notion of distributive subclasses of relations will be important in what follows in the thesis, hence, we accomodate a separate section here to properly define it.

Given three relations r, r', and r'', we say that weak composition distributes over intersection if we have that $r \diamond (r' \cap r'') = (r \cap r') \diamond (r \cap r'')$ and $(r' \cap r'') \diamond r = (r' \cap r) \diamond (r'' \cap r)$.

Definition 2 *A subclass $\mathcal{A} \subseteq 2^{\mathsf{B}}$ is a* distributive subclass *if weak composition distributes over non-empty intersections for all relations r, r', $r'' \in \mathcal{A}$. A subclass $\mathcal{A} \subseteq 2^{\mathsf{B}}$ is a* maximal distributive subclass *if there exists no other distributive subclass that properly contains $\mathcal{A}$.*

The distributivity of a class of relations is a new concept introduced in [Duckham *et al.*, 2014; Li *et al.*, 2015a] and thoroughly analysed in [Long and Li, 2015]. However, there exist several examples in the literature where the authors implicitly or explicitly exploited the fact that for certain subclasses of relations for a considered qualitative constraint language weak composition distributes over non-empty intersection for all relations of that subclass. As an example, van Beek and Cohen observed this property for the subclass of *convex relations* of Point Algebra [van Beek and Cohen, 1990]. (This observation also exists in the PhD Thesis of van Beek [Van Beek, 1990].) In another example of this property, Chandra and Pujari implicitly consider the distributivity of a subclass of convex relations for RCC-8 in their proof of Theorem 8 in [Chandra and Pujari, 2005] that involves a special reasoning problem of RCC-8 networks, viz., the minimal labeling problem, which we have already mentioned and briefly explained earlier on in the introductory chapter of the thesis and will formally define later on in Section 3.3. More recently, Amaneddine and Condotta in [Amaneddine and Condotta, 2012] identified maximal distributive subclasses of relations for Point Algebra and Interval Algebra in their effort to guarantee a particular strong characterization for networks of the aforementioned qualitative constraint languages through a polynomial algorithm. What is more interesting, is the fact that all subclasses of convex relations that have been defined in the literature for the most well-known qualitative constraint languages,

viz., Point Algebra, Interval Algebra, Cardinal Direction Calculus, Block Algebra, and RCC-8, turn out to be maximal distributive sublasses of relations for these languages [Long and Li, 2015]. This is due to the fact that subclasses of convex relations and distributive subclasses of relations share a very important common property, viz., they exhibit convexity in Helly's sense [Danzer *et al.*, 1963]. As such, distributive subclasses of relations generalize the notion of subclasses of convex relations.

However, for the sake of completeness, let us introduce the notion of *convexity* for relations of qualitative constraint languages and its relationship with distributive subclasses of relations. In general, to define a subclass of convex relations for a given qualitative constraint language, one has to take into account the semantics of the base relations of the set B of that language and assume a geometrical characterization of the language as well, and first obtain a *partial ordering* [Dushnik and Miller, 1941] on its set of base relations $(\mathsf{B}, \preceq)$. Such a partially ordered set is most often represented by a Hasse diagram [Birkhoff, 1948]. As an example, Ligozat in [Ligozat, 1994] defines a partially ordered set $(\mathsf{B}, \preceq)$ for Interval Algebra, which is represented by the Hasse diagram shown in Figure 2.8 (referred to as a *lattice* due to the form of the Hasse diagram depicting it). The interpretation of the lattice of Interval Algebra with respect to convexity is as follows. For $b_1, b_2 \in \mathsf{B}$ with $b_1 \preceq b_2$, we write $[b_1, b_2]$ as the set of base relations b such that $b_1 \preceq b \preceq b_2$, and call such a relation a *convex relation*. Hence, the total set of convex relations can be obtained by enumerating the intervals in the lattice. For example, relation $\{d, s, o, m\}$ in Interval Algebra is convex as it correponds to interval $[d, m]$ in the lattice. As we will stick with distributive subclasses of relations in the context of this thesis, we will not delve more into the details of convexity and convex relations. The interested reader is kindly asked to refer to [Ligozat, 2011] for a complete analysis of the aforementioned notions.

An observation is that subclasses of convex relations for Point Algebra, Interval Algebra, Cardinal Direction Calculus, Block Algebra, and RCC-8 are Helly [Long and Li, 2015]. A Helly subclass of relations is defined as follows.

Definition 3 *A subclass $\mathcal{A} \subseteq 2^{\mathsf{B}}$ is Helly if and only if for any n relations $r_1, r_2, \ldots r_n \in \mathcal{A}$ we have:*

$$\bigcap_{i=1}^{n} r_i \neq \emptyset \text{ iff } (\forall 1 \leq i, j \leq n)\ r_i \cap r_j \neq \emptyset$$

Then, we have the following result by Long and Li in [Long and Li, 2015]:

Theorem 1 ([Long and Li, 2015]) *A subclass $\mathcal{A} \subseteq 2^{\mathsf{B}}$ for a qualitative constraint language that is a relation algebra is distributive if and only if it is Helly.*

Maximal distributive subclasses of qualitative spatial and temporal constraint languages can be identified through an automatic procedure as described in [Long and Li, 2015]. In the simplest case, identifying a maximal distributive subclass for some qualitative constrtaint language can be done with a brute-force procedure that checks if $\widehat{\hat{\mathsf{B}} \cup Z}$ satisfies distributivity for some subset $Z \subset 2^{\mathsf{B}}$, where $\hat{X}$ denotes the closure under converse, intersection, and weak composition for a subset $X \subseteq 2^{\mathsf{B}}$.

2.6 Conclusion

In this chapter, we introduced various qualitative constraint-based formalisms for reasoning about time and space. More precisely, we introduced the algebraic structure upon which such formalisms are defined, which comprises a set of base relations and certain standard relational

operations. In particular, the set of base relations of a given qualitative constraint language allows us to represent the definite knowledge between any two or more entities with respect to the given level of granularity. Moreover, the indefinite knowledge between any two or more entities can be represented by unions of possible base relations. Regarding relational operations, we defined the converse operation and the weak composition operation, both of which are essential in being able to reason with the relations of a qualitative constraint language. For instance, given two entities and the knowledge of how the first entity is related to the second one, the converse operation allows us to obtain the knowledge of how the second entity is related to the first one. The weak composition operation allows us to eliminate certain configurations among entities for which we already have some partial knowledge. As an example, if we know that a temporal event occurs before a second temporal event, which in turn occurs before a third temporal event, the weak composition operation allows us to infer that the first temporal event occurs befores the third temporal event. We used various representative examples to highlight the different types of relations that are considered by a spatial or temporal qualitative constraint language. As a matter of fact, we considered qualitative constraint languages based on points, intervals, blocks, and even regions, and we also focused on some aspects of orientation and the means by which it can be handled. Of course, the list of qualitative constraint languages that we presented is far from exhaustive, but it is sufficient for motivating our contributions that we will present later on.

Chapter 3

Reasoning with Qualitative Constraint Networks

3.1 Introduction

In this chapter, we formally introduce the notion of a qualitative constraint network (QCN), and draw the connection between the relational operations presented in the previous chapter and some useful local consistency conditions for characterizing QCNs. We present the fundamental reasoning problems associated with QCNs, overview the state of the art algorithms for dealing with those problems, and explain some constraint properties of QCNs. Finally, we discuss about some decomposability aspects of QCNs considered in the literature.

A QCN comprises a set of variables corresponding to a set of spatial or temporal entities and a set of relations that constrain the possible qualitative configurations between the different entities. Given a QCN over a set of variables corresponding to a set of spatial or temporal entities, we are particularly interested in its *satisfiability problem*, that is, deciding whether there exists an interpretation of all the variables of the QCN such that all of its constraints are satisfied by this interpretation; such an interpretation being called a *solution*. The satisfiability problem is closely related to the *minimal labeling problem* and the *redundancy problem*, in the sense that the latter two problems exhibit functions that build on the core algorithms used to obtain a solution of a QCN. In particular, the *minimal labeling problem* is the problem of determining all the base relations for each of the constraints of a QCN that participate in at least one of its solutions, whilst the redundancy problem is the problem of obtaining all the non-redundant constraints of a QCN, i.e., those constraints that do not contain at least one base relation participating in a solution of the modified QCN that results by removing these constraints.

Depending on the subset of relations of a qualitative constraint language over which a given QCN is defined, certain characterizations of that QCN can be established that allow us to deal with the aforementioned reasoning problems in polynomial time; this is typically achieved through the use of specialized local consistency conditions and related algorithms. Moreover, and always in relation to the discussed reasoning problems, we will see how certain constraint properties in the context of qualitative constraint-based reasoning allow us to exploit the structure of some special cases of QCNs and deal with them in a more efficient manner.

3.2 Qualitative Constraint Networks (QCNs)

As we already discussed, qualitative constraint languages use relations to encode spatial or temporal knowledge between entities. Hence, it comes natural to use constraints to capture these relations. In what follows, sometimes we will refer to constraints as relations when it is clear from the context that a particular relation along with some entities forms a constraint upon those entities. The problem of reasoning about qualitative spatial or temporal information can be modelled as an infinite-domain variant of a Constraint Satisfaction Problem (CSP) [Montanari, 1974], for which we use the term Qualitative Constraint Network (QCN). For instance, there are infinitely many time points or temporal intervals in the timeline and infinitely many regions in a two or three dimensional space. Formally, a QCN is defined as follows.

Definition 4 *A* qualitative constraint network *(*QCN*) is a pair (V, C) where:*

- $V = \{v_1, \ldots, v_n\}$ *is a non-empty finite set of variables each of which corresponds to a set of spatial or temporal entities;*
- C *is a mapping that associates a relation* $r \in 2^{\mathsf{B}}$ *with each pair* (v, v') *of* $V \times V$*, that relation being denoted by* $C(v, v')$*. Mapping* C *is such that* $C(v, v) = \{\mathsf{Id}\}$ *and* $C(v, v') = (C(v', v))^{-1}$ *for every* $v, v' \in V$*.*

An example of a QCN of RCC-8 is shown in Figure 3.1 along with a corresponding spatial configuration. In particular, the QCN comprises the set of variables $\{x, y, z\}$ and the constraints $C(x, y) = C(y, z) = C(z, x) = \{EC\}$. Converse relations as well as Id loops are not mentioned or shown in the figure for simplicity. Note that we always regard a QCN as a complete network. In what follows, given a QCN $\mathcal{N} = (V, C)$ and $v, v' \in V$, relation $C(v, v')$ will also be denoted by $\mathcal{N}[v, v']$. We have the following definitions regarding QCNs:

Definition 5 *Let* $\mathcal{N} = (V, C)$ *be a* QCN*, then:*

- *a* partial solution *of* $\mathcal{N}$ *on* $V' \subseteq V$ *is a valuation* σ *of the variables of* V' *such that for each pair of variables* $(u, v) \in V' \times V'$*, we have that* $\sigma(u)$ *and* $\sigma(v)$ *satisfy* $C(u, v)$*, i.e., there exists a base relation* $b \in C(u, v)$ *such that* $(\sigma(u), \sigma(v)) \in b$*;*
- *a* solution *of* $\mathcal{N}$ *is a partial solution on* V*;*
- *a* QCN $\mathcal{N}'$ *is* equivalent *to* $\mathcal{N}$ *if and only if it admits the same set of solutions with* $\mathcal{N}$*;*
- *a* sub-QCN[1] $\mathcal{N}'$ *of* $\mathcal{N}$*, denoted by* $\mathcal{N}' \subseteq \mathcal{N}$*, is a* QCN (V', C') *such that* $V' = V$ *and* $C'(v, v') \subseteq C(v, v')$ $\forall v, v' \in V$*;*
- $\mathcal{N}$ *is* atomic *if it comprises only singleton relations, where a singleton relation is a relation* $\{b\}$ *for some base relation* $b \in B$*;*
- *a* scenario $\mathcal{S}$ *of* $\mathcal{N}$ *is an atomic satisfiable sub-*QCN *of* $\mathcal{N}$*;*
- *a* partial scenario *of* $\mathcal{N}$ *on* $V' \subseteq V$ *is a scenario restricted to constraints involving only variables of* V'*;*

[1] This term is also found by the name "*refined* QCN" throughout the literature.

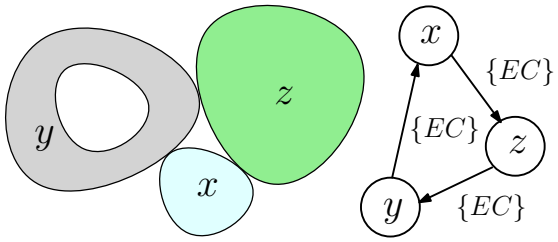


Figure 3.1: A QCN of RCC-8 along with its spatial configuration (note that region y has a hole)

- *a base relation* $b \in C(v, v')$*, with* $v, v' \in V$*, is* feasible *(resp.* unfeasible*) iff there exists (resp. there does not exist) a scenario* $\mathcal{S} = (V, C')$ *of* $\mathcal{N}$ *such that* $C'(v, v') = \{b\}$*;*

- *the* constraint graph *of* $\mathcal{N}$ *is the graph* (V, E)*, denoted by* $G(\mathcal{N})$*, for which we have that* $\{v, v'\} \in E$ *iff* $C(v, v') \neq \mathsf{B}$*;*[2]

- $\mathcal{N}$ *is said to be* trivially inconsistent *iff* $\exists v, v' \in V$ *with* $C(v, v') = \emptyset$*;*

- $\perp^V$ *will denote the* QCN $\mathcal{N}$ *whose each constraint between each pair of variables* $(v, v') \in V \times V$ *is defined by the empty relation* $\emptyset$*.*

Let us further introduce some operations with respect to QCNs.
Given a QCN $\mathcal{N} = (V, C)$ and $v, v' \in V$, we have that:

- $\mathcal{N}_{[v,v']/r}$ with $r \in 2^{\mathsf{B}}$ yields the QCN $\mathcal{N}' = (V, C')$ defined by $C'(v, v') = r$, $C'(v', v) = r^{-1}$ and $C'(y, w) = C(y, w)$ $\forall (y, w) \in (V \times V) \setminus \{(v, v'), (v', v)\}$;

- $\mathcal{N}{\downarrow_{V'}}$, with $V' \subseteq V$, yields the QCN $\mathcal{N}$ restricted to V'.

Given two QCNs $\mathcal{N} = (V, C)$ and $\mathcal{N}' = (V, C')$ of the same set of variables V, we have that:

- $\mathcal{N} + \mathcal{N}'$ yields the QCN $\mathcal{N}'' = (V, C'')$, where $C''(v, v') = C(v, v') \cup C'(v, v')$ for all $v, v' \in V$.

Given two QCNs $\mathcal{N} = (V, C)$ and $\mathcal{N}' = (V', C')$ such that $C(u, v) = C'(u, v)$ for every $u, v \in V \cap V'$, we have that:

- $\mathcal{N} \cup \mathcal{N}'$ yields the QCN $\mathcal{N}'' = (V'', C'')$, where $V'' = V \cup V'$, $C''(u, v) = C''(v, u) = \mathsf{B}$ for all $(u, v) \in (V \setminus V') \times (V' \setminus V)$, $C''(u, v) = C(u, v)$ for every $u, v \in V$, and $C''(u, v) = C'(u, v)$ for every $u, v \in V'$.

Finally, given a QCN $\mathcal{N} = (V, C)$ and a subclass $\mathcal{A} \subseteq 2^{\mathsf{B}}$, we have that:

- $\mathcal{A}(\mathcal{N})$ yields the QCN $\mathcal{N}' = (V, C')$ defined by $C'(v, v') = \mathcal{A}(C(v, v'))$ $\forall v, v' \in V$.

The aforementioned binary operations on QCNs are presented in Figure 3.2 for clarity.

[2]Note that the constraint graph of a QCN does not involve the universal relation B as it is the non-restrictive relation that contains all base relations, thus, it does not really pose a constraint. (The result of the weak composition of any relation with the universal relation is the universal relation.)

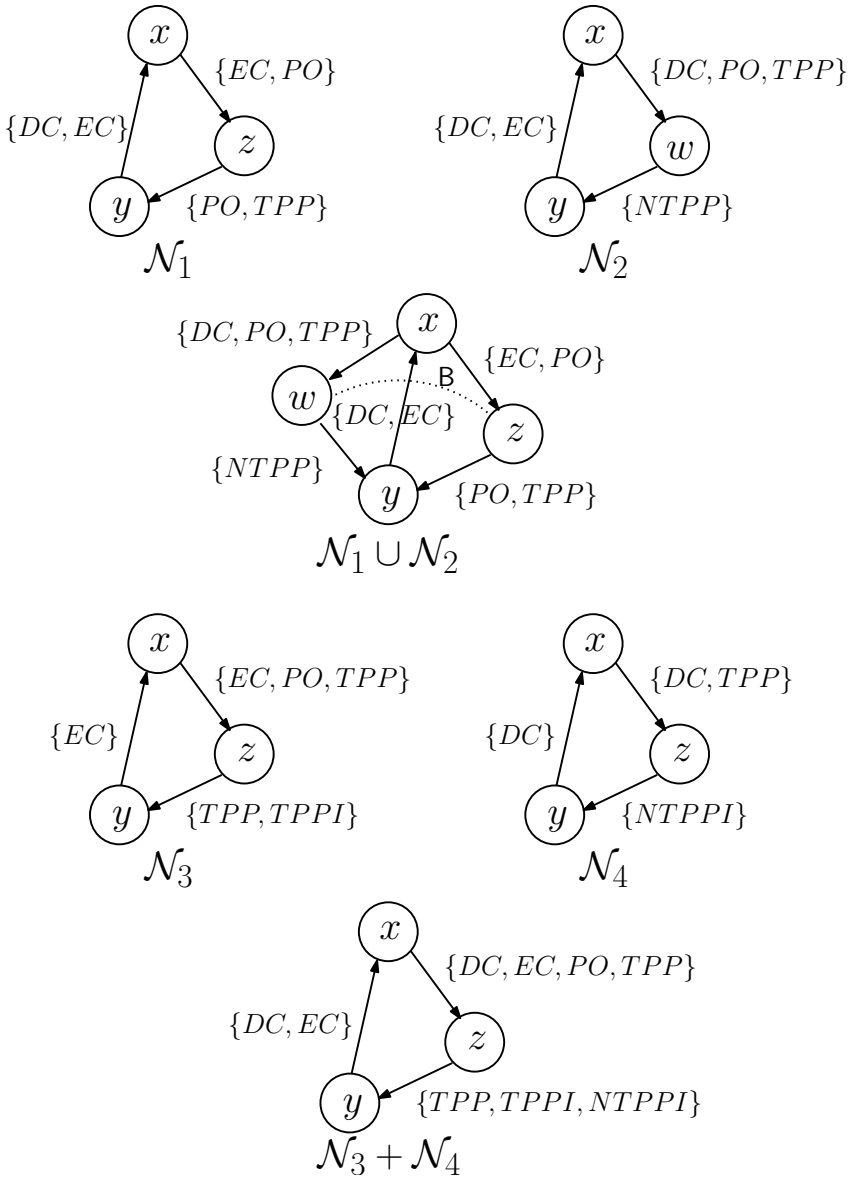


Figure 3.2: Binary operations on QCNs

3.3 Reasoning Problems Associated with QCNs

Given a QCN, we are particularly interested in its *satisfiability problem*, that is, the problem of deciding whether there exists an interpretation of all the variables of the QCN such that all its constraints are satisfied by this interpretation; such an interpretation being called a *solution* as defined earlier. The satisfiability problem is closely related to the *minimal labeling problem* (MLP) [Montanari, 1974] (cf. [Liu and Li, 2012]) and the *redundancy problem* [Duckham *et al.*, 2014; Li *et al.*, 2015a], in the sense that the latter problems exhibit functions that build on the core algorithms used to obtain a solution of a QCN. We will view the aforementioned reasoning problems in detail in what follows.

3.3.1 Satisfiability Problem

The satisfiability problem of QCN is among the most important problems associated with QCNs. We can formally define the satisfiability problem of a QCN as follows.

Definition 6 *The* satisfiability problem*, given a* QCN $\mathcal{N}$*, is the problem of deciding whether* $\mathcal{N}$ *is* satisfiable*, i.e., whether it admits a solution.*

The satisfiability problem for most of the well-known qualitative constraint languages is NP-complete. Specifically, and with the exception of Point Algebra, checking the satisfiability of an arbitrary QCN of RCC-8, Cardinal Direction Calculus, Interval Algebra, or Block Algebra is NP-complete [Renz and Nebel, 1999; Ligozat, 1998; Nebel and Bürckert, 1995; Balbiani *et al.*, 2002]. Checking the satisfiability of a QCN of Point Algebra can be done in polynomial time [van Beek, 1992]. In the literature, the usual approach for solving the satisfiability problem of a given QCN involves obtaining a scenario of that QCN. This is because we have a particular local consistency condition for deciding the satisfiability of atomic QCNs for many qualitative constraint languages, that we will discuss in a later section. Once the scenario is obtained, a solution of the QCN can be constructed in polynomial time using some *canonical model*, i.e., a structure that allows to model any satisfiable sentence of the qualitative constraint language. Of particular interest is the case of RCC-8, for which several canonical models have been defined in order to obtain interesting solutions, such as a domain of regular closed sets of the set of real numbers [Challita, 2012], a domain of countably many homeomorphic disjoint components of some topological space [Li and Wang, 2006; Li, 2006], and the usual domain of regions corresponding to regular closed subsets of some topological space that do not have to be internally connected and do not have a particular dimension [Renz, 2002a]. The canonical model of Renz, allows a simple representation of regions with respect to a set of RCC-8 constraints, and, further, enables one to generate realizations in any dimension $d \geq 1$. As an example, let us view the QCN of RCC-8 in Figure 3.1. This QCN is atomic and also, clearly, satisfiable. As such, the QCN is also a scenario of itself. A solution σ of the QCN is shown to its left, where $\sigma(x)$, $\sigma(y)$, and $\sigma(z)$ correspond to the 2-dimensional shapes tagged with x, y, and z respectively in the figure. As our domain is infinite, note that we can have an infinite number of solutions for any given scenario.

There have been many works in the literature that try to solve the satisfiability problem of a non-tractable QCN[3] in an efficient manner. All these works consider some kind of backtracking search (either iterative or recursive), a preprocessing operation to prune off certain unfeasible base relations that is based on a polynomial algorithm that makes use of weak composition, and, most importantly, a subclass of relations of 2^{B} with the property that QCNs defined over that subclass become tractable [Nebel and Bürckert, 1995; Nebel, 1997; Renz and Nebel, 2001; Renz, 1999; Renz and Nebel, 1999; Balbiani *et al.*, 2002; Balbiani *et al.*, 1999; Ligozat, 1998]. Of course, for tractable QCNs, dedicated polynomial algorithms have been defined, as is the case for QCNs of Point Algebra where a polynomial method based on *topological sort* [Knuth, 1973] is defined in [van Beek, 1992, chap. 3]. We will review the approaches for solving the satisfiability problem of QCNs in more detail in the following sections, and we will also present our contributions for solving the satisfiability problem of QCNs later on in a separate chapter.

[3] In what follows, as a convention, by saying that a QCN $\mathcal{N}$ is tractable (resp. non-tractable), we mean that the satisfiability problem for the class of QCNs that is defined by the restrictions imposed on $\mathcal{N}$ (if any) is tractable (resp. non-tractable), i.e., solvable (resp. not solvable) by a deterministic Turing machine in polynomial time.

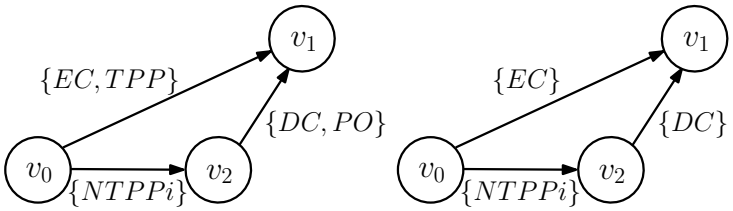


Figure 3.3: A RCC-8 network (left) and its minimal network (right)

3.3.2 Minimal Labeling Problem

The minimal labeling problem (MLP), also known as the deductive closure problem, is a fundamental problem in qualitative spatial and temporal reasoning which involves making all the constraints of a QCN minimal, i.e., obtaining the base relations participating in at least one solution for each of the constraints of that network.

Definition 7 *Given a* QCN $\mathcal{N} = (V, C)$*, we say that a relation* $C(v, v')$*, with* $v, v' \in V$*, is* minimal *if and only if every base relation* $b \in C(v, v')$ *is feasible.*

Notably, the MLP of a QCN is equivalent to the corresponding satisfiability problem with respect to polynomial Turing-reductions. As such, verifying if a QCN comprises only minimal relations is a NP-hard problem for QCNs for which the satisfiability problem is NP-complete (cf. [Liu and Li, 2012]). Since their introduction in 1974 [Montanari, 1974], minimal constraint networks have been the focus of study in both the constraint programming community (cf. [Gottlob, 2012]) and the qualitative spatial and temporal reasoning community [van Beek and Cohen, 1990; Gerevini and Saetti, 2011; Liu and Li, 2012; Chandra and Pujari, 2005; Amaneddine and Condotta, 2013; Bessière *et al.*, 1996; Gerevini and Schubert, 1995; Amaneddine and Condotta, 2012]. We define the notion of a minimal QCN as follows.

Definition 8 *A* QCN $\mathcal{N} = (V, C)$ *is* minimal *if and only if* $\forall v, v' \in V$ *we have that* $C(v, v')$ *is minimal.*

Finally, the MLP of a QCN can be formally defined as follows.

Definition 9 *Given a* QCN $\mathcal{N}$*, the* minimal labeling problem (MLP) *is the problem of obtaining the largest (with respect to* $\subseteq$*) minimal sub-*QCN $\mathcal{N}'$ *of* $\mathcal{N}$*.*

Solving the MLP of a given QCN involves prunning off unfeasible base relations iteratively until a state is reached where no more unfeasible base relations exist. In that state, all the constraints of the (possibly) modified QCN will be minimal and, thus, we will have obtained a minimal QCN. The unique equivalent minimal sub-QCN of a QCN $\mathcal{N}$ is denoted by $\mathcal{N}_{\mathsf{min}}$. As an example, let us view the QCN $\mathcal{N}$ of RCC-8 in Figure 3.3 on the left hand. Network $\mathcal{N}$ is satisfiable, but not minimal, as it comprises unfeasible base relations. In particular, the base relation PO between variables v_1 and v_2 is unfeasible as $PO \notin \{NTPPi\}^{-1} \diamond \{EC, TPP\}$ and, thus, it cannot participate in a solution. Likewise, base relation $TPP(v_0, v_1)$ is unfeasible as $TPP \notin \{NTPPi\} \diamond \{DC, PO\}$ and, thus, it also cannot participate in a solution. By removing the aforementioned two unfeasible base relations we obtain the minimal network on the right hand of the figure, i.e., network $\mathcal{N}_{\mathsf{min}}$.

In the literature, the works on obtaining the equivalent minimal sub-QCN of a QCN $\mathcal{N}$ have focused on identifying particular subclasses of relations of 2^{B} for the qualitative constraint

language considered for which minimality can be guaranteed through a polynomial algorithm [van Beek and Cohen, 1990; Gerevini and Saetti, 2007; Gerevini and Saetti, 2011; Liu and Li, 2012; Chandra and Pujari, 2005; Amaneddine and Condotta, 2013; Bessière *et al.*, 1996; Gerevini and Schubert, 1995; Amaneddine and Condotta, 2012]. In the particular case of [Amaneddine and Condotta, 2013], the authors go a step further by proposing a backtracking search algorithm that iteratively prunes off unfeasible base relations of a given QCN $\mathcal{N}$ until $\mathcal{N}_{\mathsf{min}}$ is obtained. We will review the approaches for solving the MLP of QCNs in more detail in the following sections, and we will also present our contributions for solving the MLP of QCNs later on in a separate chapter.

3.3.3 Redundancy Problem

Recently, the important problem of deriving redundancy in a RCC-8 network was considered and already well established in [Duckham *et al.*, 2014; Li *et al.*, 2015a]. For a RCC-8 network $\mathcal{N}$ a constraint is *redundant*, if removing that constraint from $\mathcal{N}$ (i.e., replacing that constraint with the universal relation) does not change the solution set of $\mathcal{N}$. A *prime network* of $\mathcal{N}$ is a network which contains no redundant constraints, but has the same solution set as $\mathcal{N}$. Finding a prime network can be useful in many applications such as computing, storing, and compressing the relationships between spatial objects and hence saving space for storage and communication, facilitating comparison between different networks, merging networks [Condotta *et al.*, 2008; Condotta *et al.*, 2009], aiding quering in spatially-enhanced databases [Nikolaou and Koubarakis, 2013; Open Geospatial Consortium, 2012], unveiling the essential network structure of a network (e.g., being a tree or of bounded *treewidth* [Bodirsky and Wölfl, 2011]), and adjusting geometrical objects to meet topological constraints [Wallgrün, 2012]. We refer the reader to [Li *et al.*, 2015a] for a well depicted real motivational example and further application possibilities. In [Li *et al.*, 2015a], the notion of redundancy is also generalized to qualitative constraint languages other than just RCC-8.

Given a QCN $\mathcal{N} = (V, C)$, we say that $\mathcal{N}$ *entails* a relation $r(v, v') \in 2^{\mathsf{B}}$, with $v, v' \in V$, if for every solution σ of $\mathcal{N}$, the relation defined by $(\sigma(v), \sigma(v'))$ is a base relation b such that $b \in r(v, v')$. A relation $C(v, v')$ in $\mathcal{N}$ is *redundant* if network $\mathcal{N}_{[v,v']/\mathsf{B}}$ entails $C(v, v')$. Note that by definition every universal relation B in a QCN is redundant. Recalling the fact that the constraint graph of a QCN involves all the non-universal relations, we trivially obtain the following lemma:

Lemma 1 *Given a* QCN $\mathcal{N} = (V, C)$ *and its constraint graph* $G(\mathcal{N}) = (V, E)$*, a relation* $\mathcal{N}[v, v']$ *with* $v, v' \in V$ *is redundant if* $\{v, v'\} \notin E$*.*

We recall the definition of a reducible and a prime QCN.

Definition 10 *A* QCN $\mathcal{N} = (V, C)$ *is* reducible *if it comprises a redundant relation other than relation* B*, and* irreducible *otherwise. An equivalent to* $\mathcal{N}$ *irreducible* QCN $\mathcal{N}' = (V, C')$ *such that* $C'(v, v') \subseteq C(v, v')\ \forall v, v' \in V$ *if* $C'(v, v') \neq \mathsf{B}$ *is called a* prime QCN *of* $\mathcal{N}$*. If a prime* QCN *of* $\mathcal{N}$ *is also unique, it is denoted by* $\mathcal{N}_{\mathsf{prime}}$*.*

Given an arbitrary QCN $\mathcal{N}$, finding a prime QCN of $\mathcal{N}$ is clearly at least as hard as determining if $\mathcal{N}$ is reducible. We then have the following result for QCNs for which the satisfiability problem is NP-complete.

Proposition 2 ([Li *et al.*, 2015a]) *Let* $\mathcal{N} = (V, C)$ *be a* QCN *for which the satisfiability problem is* NP*-complete. It is co-*NP*-complete to decide if a relation* $C(v, v')$*, with* $v, v' \in V$*, is redundant in* $\mathcal{N}$*.*

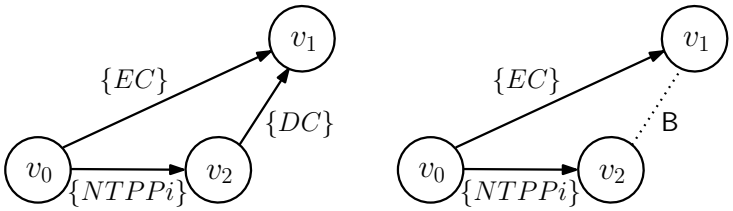


Figure 3.4: A RCC-8 network (left) and its prime network (right)

We can now formally define the redundancy problem of a QCN as follows.

Definition 11 *Given a* QCN $\mathcal{N}$*, the* redundancy problem *is the problem of determining a prime* QCN *of* $\mathcal{N}$*.*

The redundancy problem was first approached by Egenhofer and Sharma for topological constraints [Egenhofer and Sharma, 1993], where they observed that a minimal set of constraints (i.e., a prime network) contains somewhere between $(n-1)$ and $(n^2-n)/2$ non-univesral relations, but without providing any efficient algorithms for deriving such a minimal set even for atomic networks. In a recent paper, Wallgrün proposed two algorithms to approximately find the prime network [Wallgrün, 2012]. As observed in [Wallgrün, 2012], and later explored in more detail in [Li *et al.*, 2015a], neither of these two algorithms is guaranteed to provide the optimal simplification. The redundancy problem is also related to the minimal labeling problem that we defined earlier, in the sense that to obtain a minimal network we remove "unnecessary" base relations from each constraint. In a similar manner, to obtain a prime network we remove "redundant" constraints from the qualitative constraint network. As an example, let us view the QCN $\mathcal{N}$ of RCC-8 in Figure 3.4 on the left hand. Network $\mathcal{N}$ satisfiable, but not irreducible, as it comprises redundant relations. In particular, the relation $\{DC\}$ between variables v_1 and v_2 is redundant as it can be entailed by $\mathcal{N}_{[v_1,v_2]/\mathsf{B}}$ and, thus, can be replaced with relation B (that denotes the lack of a constraint between two entities in a QCN). By removing that redundant relation we obtain the equivalent to $\mathcal{N}$ and irreducible network on the right hand of the figure, i.e., network $\mathcal{N}_{\mathsf{prime}}$.

As already noted, the redundancy problem of QCNs has only recently been formally established in terms of providing efficient algorithms for obtaining optimal solutions [Duckham *et al.*, 2014; Li *et al.*, 2015a]. In particular, in [Duckham *et al.*, 2014; Li *et al.*, 2015a] it is proven that QCNs defined over particular subclasses of relations of 2^{B} can yield their unique prime networks with the aid of a polynomial algorithm. We will review the approaches for solving the redundancy problem of QCNs in more detail in the following sections, and we will also present our contributions for solving the redundancy problem of QCNs later on in a separate chapter.

3.4 Tractability of QCNs

Earlier in this chapter, there has been some mentioning of particular subclasses of relations of 2^{B} with the property that QCNs defined over that subclass become tractable. We will simply refer to such subclasses of relations as tractable subclasses of relations. A tractable subclass of relations is defined as follows.

Definition 12 *A subclass* $\mathcal{A} \subseteq 2^{\mathsf{B}}$ *is a* tractable subclass *if a* QCN $\mathcal{N}$ *comprising only relations from* $\mathcal{A}$ *is tractable. A subclass* $\mathcal{A} \subseteq 2^{\mathsf{B}}$ *is a* maximal tractable subclass *if there exists no other tractable subclass that properly contains* $\mathcal{A}$*.*

For many qualitative constraint languages, the essence of the notion of tractability for a given subclass $\mathcal{A} \subseteq 2^{\mathsf{B}}$ lies in the fact that a satisfiable QCN defined over such a subclass can be polynomially refined to an atomic satisfiable sub-QCN, viz., a scenario, for which a particular local consistency condition can be used to verify its satisfiability [Huang, 2012; Renz, 2007]. If a QCN $\mathcal{N} = (V, C)$ over a tractable subclass of relations is unsatisfiable, it will be polynomially refined to $\bot^V$. For example, the maximal tractable subclasses for RCC-8 and IA are the classes $\hat{\mathcal{H}}_8, \mathcal{C}_8$, and $\mathcal{Q}_8$ [Renz, 1999; Renz and Nebel, 2001], and $\mathcal{H}_{\mathsf{IA}}$ [Nebel and Bürckert, 1995; Nebel, 1997], respectively. Classes $\hat{\mathcal{H}}_8$ and $\mathcal{H}_{\mathsf{IA}}$ contain exactly those relations that are transformed to propositional Horn formulas when using the propositional encodings of RCC-8 and IA respectively. Further, and for RCC-8 in particular, let us denote by $\mathcal{NP}_8$ the set of relations that by themselves result in NP-completeness when combined with the set of singleton relations, defined as follows in [Renz, 1999].

$$\begin{aligned}\mathcal{NP}_8 = \quad & \{\, r \mid (\{PO\} \not\subseteq r \text{ and } (\{NTPP\} \subseteq r \text{ or } \{TPP\} \subseteq r) \\ & \text{and } (\{NTPPi\} \subseteq r \text{ or } \{TPPi\} \subseteq r)\} \\ & \cup \{\{EC, NTPP, EQ\}, \{DC, EC, NTPP, EQ\}, \\ & \{EC, NTPPi, EQ\}, \{DC, EC, NTPPi, EQ\}\}\end{aligned}$$

Then, the maximal tractable subclasses for RCC-8 are defined using the $\mathcal{NP}_8$ set of relations as follows ([Renz, 1999]).

$$\begin{aligned}\hat{\mathcal{H}}_8 = \quad & (2^{\mathsf{B}} \setminus \mathcal{NP}_8) \setminus \{\, r \mid (\{EQ, NTPP\} \subseteq r \text{ and } \{TPP\} \not\subseteq r) \\ & \text{or } (\{EQ, NTPPi\} \subseteq r \text{ and } \{TPPi\} \not\subseteq r)\} \\ \mathcal{C}_8 = \quad & (2^{\mathsf{B}} \setminus \mathcal{NP}_8) \setminus \{\, r \mid (\{EC\} \subset r \text{ and } \{PO\} \not\subseteq r) \text{ and} \\ & r \cap \{TPP, NTPP, TPPi, NTPPi, EQ\} \neq \emptyset\} \\ \mathcal{Q}_8 = \quad & (2^{\mathsf{B}} \setminus \mathcal{NP}_8) \setminus \{\, r \mid (\{EQ\} \subset r \text{ and } \{PO\} \not\subseteq r) \text{ and} \\ & r \cap \{TPP, NTPP, TPPi, NTPPi\} \neq \emptyset\}\end{aligned}$$

If we denote by $\mathcal{P}_8$ the set of relations $2^{\mathsf{B}} \setminus \mathcal{NP}_8$ (i.e., the set of relations where every relation belongs to at least one of the classes $\hat{\mathcal{H}}_8, \mathcal{C}_8$, and $\mathcal{Q}_8$), then all relations of $\mathcal{P}_8$ not contained in $\mathcal{C}_8$ contain $E\underline{C}$ and all relations of $\mathcal{P}_8$ not contained in $\mathcal{Q}_8$ contain $E\underline{Q}$ [Renz, 1999]. The propositional encoding of either $\mathcal{C}_8$ or $\mathcal{Q}_8$ is neither a Horn formula nor a Krom formula, but classes $\mathcal{C}_8$ and $\mathcal{Q}_8$ themselves are directly related to class $\hat{\mathcal{H}}_8$ in the sense that any QCN defined over either $\mathcal{C}_8$ or $\mathcal{Q}_8$ can be polynomially refined to a sub-QCN defined over $\hat{\mathcal{H}}_8$ [Renz, 1999].

In general, large tractable subclasses of qualitative spatial and temporal constraint languages can be identified without any manual intervention and without the need for additional NP-hardness proofs, through an automatic procedure as described in [Renz, 2007]. However, maximality of the tractable subclasses of relations obtained through the aforementioned procedure is not guaranteed for the qualitative constraint languages and, thus, a formal analysis of the qualitative constraint language at hand is still required [Renz, 2007]. As a final note, we consider maximal tractable subclasses of relations that by definition include all the singleton relations of the considered qualitative constraint language. For certain qualitative constraint languages, such as Interval Algebra, other subsets of their relations that do not contain all of the singleton relations can also be maximally tractable [Krokhin *et al.*, 2001; Krokhin *et al.*, 2003].

3.5 Algorithms for Reasoning with QCNs

We present here the state of the art algorithms that are used for qualitative spatial and temporal reasoning. Further, we make the connection with the reasoning problems presented in Section 3.3

along with certain properties of subclasses of relations by referring to related results where applicable.

3.5.1 Algebraic Closure and $\diamond$-consistency

Given a QCN $\mathcal{N} = (V, C)$, the method of *algebraic closure* [Ligozat and Renz, 2004] (also called *closure under weak composition*) removes certain base relations that are guaranteed to not participate in any solution of $\mathcal{N}$. The algebraic closure applies the following iterative operation until a fixed state is reached:

$$\forall v_i, v_k, v_j \in V, \quad C(v_i, v_j) \leftarrow C(v_i, v_j) \cap (C(v_i, v_k) \diamond C(v_k, v_j))$$

Due to the definition of the weak composition operation denoted by symbol $\diamond$, the algebraic closure method is sound as it only removes base relations that do not participate in any solution of a given qualitative constraint network. Every time the algebraic closure method results in the assignment of the empty relation $\emptyset$ to a constraint of a given qualitative constraint network, we can conclude that the given network is unsatisfiable. However, in general, it is not complete for deciding the satisfiability of any qualitative constraint network, i.e., we cannot conclude the satisfiability of an arbitrary qualitative constraint network if the algebraic closure method does not result in the assignment of the empty relation $\emptyset$ to a constraint of the network at hand. We will now present a local consistency, called $\diamond$-consistency, that is directly related to the algebraic closure method.

Definition 13 *A* QCN $\mathcal{N} = (V, C)$ *is said to be* $\diamond$-consistent *if and only if we have that* $C(v_i, v_j) \subseteq C(v_i, v_k) \diamond C(v_k, v_j)$, $\forall v_i, v_k, v_j \in V$.

The $\diamond$-consistent QCN obtained after the application of the algebraic closure method on a QCN $\mathcal{N}$ is unique and equivalent to $\mathcal{N}$. Further, it is called the closure of $\mathcal{N}$ under $\diamond$-consistency and it is denoted by $\diamond(\mathcal{N})$. Network $\diamond(\mathcal{N})$ corresponds to the largest (with respect to $\subseteq$) $\diamond$-consistent sub-QCN of $\mathcal{N}$. Given two QCNs $\mathcal{N} = (V, C)$ and $\mathcal{N}' = (V, C')$ we have the following properties with respect to $\diamond$-consistency:

- $\diamond(\mathcal{N}) \subseteq \mathcal{N}$ (Dominance);
- $\diamond(\mathcal{N})$ is equivalent to $\mathcal{N}$ (Equivalence);
- $\diamond(\diamond(\mathcal{N})) = \diamond(\mathcal{N})$ (Idempotence);
- if $\mathcal{N}' \subseteq \mathcal{N}$ then $\diamond(\mathcal{N}') \subseteq \diamond(\mathcal{N})$ (Monotonicity).

Interestingly, $\diamond$-consistency is able to decide the satisfiability of atomic QCNs for many qualitative constraint languages. In fact, $\diamond$-consistency provides the particular local consistency condition we were referring to in Sections 3.3.1 and 3.4. In particular, we have the following result by reviewing all the related literature for the satisfiability problem of a QCN of Point Algebra, Cardinal Direction Calculus, Interval Algebra, Block Algebra, or RCC-8 [van Beek, 1992; Ligozat, 1996; Nebel, 1995; Nebel and Bürckert, 1995; Nebel, 1997; Renz and Nebel, 2001; Renz, 1999; Renz and Nebel, 1999; Balbiani *et al.*, 2002; Balbiani *et al.*, 1999; Ligozat, 1998], or by simply consulting [Dylla *et al.*, 2013] which summarises the related contributions:

Proposition 3 (cf. [Dylla *et al.*, 2013]) *Let* $\mathcal{N} = (V, C)$ *be an atomic* QCN *of Point Algebra, Cardinal Direction Calculus, Interval Algebra, Block Algebra, or* RCC-8*. Then,* $\mathcal{N}$ *is satisfiable if and only if it is* $\diamond$*-consistent.*

With respect to the discussion about tractable subclasses of relations in Section 3.4, and the related literature, we can obtain the following result:

Proposition 4 ([Huang, 2012; Renz, 2007]) *Let $\mathcal{N} = (V, C)$ be a* QCN *of Point Algebra, Cardinal Direction Calculus, Interval Algebra, Block Algebra, or* RCC*-8, defined over one of the classes $\mathcal{P}_{\mathsf{PA}}$, $\mathcal{P}_{\mathsf{CDC}}$, $\mathcal{H}_{\mathsf{IA}}$, $\mathcal{H}^n_{\mathsf{IA}}$, or $\hat{\mathcal{H}}_8$, $\mathcal{C}_8$, and $\mathcal{Q}_8$ respectively. If $\mathcal{N}$ is not trivially inconsistent and $\diamond$-consistent, then $\mathcal{N}$ can be polynomially refined to an atomic $\diamond$-consistent sub-*QCN.

Due to Propositions 3 and 4, we can obtain the following result:

Proposition 5 *Let $\mathcal{N} = (V, C)$ be a* QCN *of Point Algebra, Cardinal Direction Calculus, Interval Algebra, Block Algebra, or* RCC*-8, defined over one of the classes $\mathcal{P}_{\mathsf{PA}}$, $\mathcal{P}_{\mathsf{CDC}}$, $\mathcal{H}_{\mathsf{IA}}$, $\mathcal{H}^n_{\mathsf{IA}}$, or $\hat{\mathcal{H}}_8$, $\mathcal{C}_8$, and $\mathcal{Q}_8$ respectively. If $\mathcal{N}$ is not trivially inconsistent and $\diamond$-consistent, then $\mathcal{N}$ is satisfiable.*

Note that for Point Algebra the maximal tractable subclass of relations $\mathcal{P}_{\mathsf{PA}}$[4] coincides with the total set of relations 2^{B}. Also, we discussed in Section 3.3.1 that a QCN of Point Algebra can be solved with a polynomial method based on topological sort as defined in [van Beek, 1992, chap. 3]. This method takes $O(n^2)$ time for a given QCN of Point Algebra over n variables; on the other hand, $\diamond$-consistency, although more time consuming to enforce as we will see later on, defines a more general approach and, hence, is still able to decide the satisfiability of that QCN.

It is important to remark that there exist qualitative constraint languages for which the satisfiability of atomic QCNs cannot be decided by $\diamond$-consistency. Such a qualitative constraint language is the Star algebra $\mathcal{STAR}^r_m$, for $m = 4$, which is also a relation algebra [Renz and Mitra, 2004]. The satisfiability problem of an atomic QCN of $\mathcal{STAR}^r_4$ is tractable, i.e., it can be solved in polynomial time, but it cannot be done so with the use of $\diamond$-consistency. In fact, atomic QCNs of $\mathcal{STAR}^r_4$ are solvable through a polynomial reduction to a particular instance of *linear programming* [Schrijver, 1986].

Let us list the following two properties for a given qualitative constraint language $\mathcal{L}$ that we will refer to from here on where necessary:

- $\mathcal{L}$ is a relation algebra;
- every $\diamond$-consistent atomic QCN of $\mathcal{L}$ is satisfiable.

Regarding the minimal labeling problem, the use of $\diamond$-consistency in the literature provides us with the following result:

Theorem 2 ([Long and Li, 2015]) *Let $\mathcal{N} = (V, C)$ be a* QCN *defined over a distributive subclass of relations of a qualitative constraint language that is a relation algebra and for which every $\diamond$-consistent atomic* QCN *is satisfiable. If $\mathcal{N}$ is not trivially inconsistent and $\diamond$-consistent, then $\mathcal{N}$ is minimal.*

Due to Propositions 1 (at page 21) and 3 (which are also used in [Long and Li, 2015]) and Theorem 2, we can obtain the following corollary:

Corollary 1 *Let $\mathcal{N} = (V, C)$ be a* QCN *defined over a distributive subclass of relations of Point Algebra, Cardinal Direction Calculus, Interval Algebra, Block Algebra, or* RCC*-8. If $\mathcal{N}$ is not trivially inconsistent and $\diamond$-consistent, then $\mathcal{N}$ is minimal.*

[4] $\mathcal{P}$ stands for *pre-convex* [Ligozat, 2011].

Before moving on, let us recall the following definition of an *all-different* qualitative constraint network:

Definition 14 *Let $\mathcal{N} = (V, C)$ be a satisfiable* QCN *of a qualitative constraint language that is a relation algebra and for which every $\diamond$-consistent atomic* QCN *is satisfiable. Then, $\mathcal{N}$ is said to be* all-different *iff $\forall u, v \in V$, with $u \neq v$, we have that $\mathcal{N}$ does not entail relation $(u \{EQ\} v)$.*

For the redundancy problem, Li et al. in [Li *et al.*, 2015a] exploit $\diamond$-consistency in the following manner:

Lemma 2 ([Li *et al.*, 2015a]) *Let $\mathcal{N} = (V, C)$ be an all-different* QCN *defined over a distributive subclass of relations of a qualitative constraint language that is a relation algebra and for which every $\diamond$-consistent atomic* QCN *is satisfiable. If $\mathcal{N}$ is not trivially inconsistent and $\diamond$-consistent, then a relation $\mathcal{N}[v, v']$, with $v, v' \in V$, is non-redundant in $\mathcal{N}$ iff we have that $\mathcal{N}[v, v'] \neq \bigcap\{\mathcal{N}[v, v''] \diamond \mathcal{N}[v'', v'] \mid v'' \in V \setminus \{v, v'\}\}$.*

Then, due to Propositions 1 (at page 21) and 3 and Lemma 2, we can obtain the following corollary:

Corollary 2 *Let $\mathcal{N} = (V, C)$ be an all-different* QCN *defined over a distributive subclass of relations of Point Algebra, Cardinal Direction Calculus, Interval Algebra, Block Algebra, or* RCC*-8. If $\mathcal{N}$ is not trivially inconsistent and $\diamond$-consistent, then a relation $\mathcal{N}[v, v']$, with $v, v' \in V$, is non-redundant in $\mathcal{N}$ iff we have that $\mathcal{N}[v, v'] \neq \bigcap\{\mathcal{N}[v, v''] \diamond \mathcal{N}[v'', v'] \mid v'' \in V \setminus \{v, v'\}\}$.*

The aforementioned results regarding the redundancy problem of a given QCN $\mathcal{N}$, allows one to extract the total set of non-redundant constraints in $\mathcal{N}$ with a polynomial algorithm (in particular, cubic in the number of variables of $\mathcal{N}$) and construct $\mathcal{N}_{\mathsf{prime}}$ as described in [Li *et al.*, 2015a]. We will present that algorithm in detail in Section 3.5.4.

Let us now discuss some complexity results with respect to $\diamond$-consistency. Clearly, given a QCN $\mathcal{N} = (V, C)$, $\diamond$-consistency for $\mathcal{N}$ can be decided or determined in $O(n^3)$ time, where $n = |V|$, as there exist at most $n(n-1)(n-2)$ possible triples of variables in V and a single check $C(v_i, v_j) \subseteq C(v_i, v_k) \diamond C(v_k, v_j)$ is performed for every such triple of variables v_i, v_j, $v_k \in V$. Applying or enforcing $\diamond$-consistency on $\mathcal{N}$, i.e., making $\mathcal{N}$ $\diamond$-consistent if it is not, requires the implementation of the algebraic closure method through an algorithm. As $\diamond$-consistency is closely related to path consistency in the sense that $\diamond$-consistency is a weaker notion of path consistency where the relational composition operation $\circ$ is replaced by the weak composition operation $\diamond$ [Renz and Ligozat, 2005], the algorithms that efficiently implement the algebraic closure method are inspired by algorithms for enforcing path consistency in the CSP framework, in particular, algorithms PC1 and PC2 [Mackworth and Freuder, 1985; Mackworth, 1977; Montanari, 1974].

Algorithms that are based on PC1 are rather naive in their approach. In particular, they operate as follows. Given a QCN $\mathcal{N} = (V, C)$, PC1-based algorithms contain an inner loop that closes $\mathcal{N}$ under weak composition by performing the $C(v_i, v_j) \leftarrow C(v_i, v_j) \cap (C(v_i, v_k) \diamond C(v_k, v_j))$ consistency operation $\forall v_i, v_k, v_j \in V$. Clearly, and as noted earlier, since there are $O(n^3)$ many such triples, where $n = |V|$, this inner loop costs $O(n^3)$ time. However, there also exists an outer loop that is resumed every time there has been a removal of at least one base relation from some constraint of $\mathcal{N}$ during the iteration of the inner loop. Since there can be at most $O(n^2|\mathsf{B}|)$ base relations in $\mathcal{N}$, the outer loop can be executed a total of $O(n^2)$ times. As such, the worst-case runtime of PC1-based algorithms is $O(n^5)$, and the best-case runtime is $\Omega(n^3)$ which occurs when the QCN is already $\diamond$-consistent and only one iteration through the

outer loop is needed. On the other hand, PC2-based algorithms can achieve a worst-case runtime of $O(n^3)$ for that same QCN $\mathcal{N}$. In the context of QCNs, PC2-based algorithms use a dedicated data structure (usually a queue as we will see later) that is initialized with all the constraints of a given QCN $\mathcal{N} = (V, C)$ over n variables that need to be processed, which can be at most $O(n^2)$ in number. A main loop selects and removes a constraint from the data structure and performs $O(n)$ consistency operations which consider all triples of variables that involve that constraint. As an example, if the constraint $C(v_i, v_j)$ is chosen, with v_i, $v_j \in V$, the consistency operations $C(v_i, v_k) \leftarrow C(v_i, v_k) \cap (C(v_i, v_j) \diamond C(v_j, v_k))$ and $C(v_k, v_j) \leftarrow C(v_k, v_j) \cap (C(v_k, v_i) \diamond C(v_i, v_j))$ are performed for every $v_k \in V$. Since every constraint of $\mathcal{N}$ can have at most $|\mathsf{B}|$ base relations, it follows that every constraint of $\mathcal{N}$ can be revised (i.e., one of its base relations can be prunned off during a consistency operation) a total of $O(|\mathsf{B}|)$ times (constant). When a constraint is revised, it is added to the data structure –if it does not already exist there– as it can affect other constraints through its participation in follow up consistency operations. Thus, at any point of the execution, the data structure only keeps track of the constraints that need to be processed in a follow up consistency operation. In addition, and as mentioned earlier, the total number of constraints that can exist in the data structure is bounded by $O(n^2)$, which constitutes the memory footprint of the algorithm. Hence, the worst-case runtime of PC2-based algorithms is $O(n^3)$, and the best-case runtime is the same as with PC1-based algorithms, viz., $\Omega(n^3)$.

The WC Algorithm

An algorithm that implements the algebraic closure method, and that is based on the PC2 algorithm, is provided in Algorithm 1, called WC, where a queue is assumed as the primary data structure for storing constraints to be processed. Given a QCN $\mathcal{N} = (V, C)$, the queue is initialized with all the pairs of variables in V which correspond to the constraints of $\mathcal{N}$ (line 2). The consistency operations take place in lines 6 and 13. Notice that whenever a constraint is modified by a reduction of its set of base relations (lines 7 and 14), it is appropriately added back to the queue (lines 10, 12, 17, and 19).

Proposition 6 ([Allen, 1983; Vilain *et al.*, 1990]) *Given a* QCN $\mathcal{N}$ *of a qualitative constraint language that is a relation algebra, algorithm* WC *terminates and returns* $\diamond(\mathcal{N})$.

Given a QCN $\mathcal{N} = (V, C)$, and as explained earlier for PC2-based algorithms, algorithm WC enforces $\diamond$-consistency on $\mathcal{N}$ in $O(|V|^3|\mathsf{B}|)$ time. Variations of the WC algorithm exist with respect to the selection of the next constraint to be processed, but also the choice of the data structure for storing the constraints to be processed. It also very important to define techniques that will allow for fast weak composition, intersection, and converse operations. Let us discuss some of these approaches.

Data Structures. Regarding the choice of the data structure, the most common approach, and to the best of our knowledge the solely used one nowadays, is based on opting for some queue data structure.

In the past, dedicated $n \times n$ $(0, 1)$-matrices have also been used to store the pairs of variables corresponding to the constraints of a given QCN to be processed within a $\diamond$-consistency enforcing algorithm. However, accessing the set of contraints that need to be processed alone takes $O(n^2)$ time, as a full iteration of the matrix needs to be performed in any case to identify the pairs of variables that are assigned the value of 1, i.e., that need to be processed (as opposed to the

Algorithm 1: $\mathsf{WC}(\mathcal{N})$

in : A QCN $\mathcal{N} = (V, C)$ of a qualitative constraint language that is a relation algebra.
output : $\diamond(\mathcal{N})$.

```
1  begin
2      Q ← {(v_i, v_j) | v_i, v_j ∈ V with 0 ≤ i ≤ j < |V|};
3      while Q ≠ ∅ do
4          (v_i, v_j) ← Q.pop();
5          foreach v_k ∈ V do
6              t ← C(v_i, v_k) ∩ (C(v_i, v_j) ⋄ C(v_j, v_k));
7              if t ≠ C(v_i, v_k) then
8                  C(v_i, v_k) ← t; C(v_k, v_i) ← t^{-1};
9                  if i ≤ k then
10                     Q ← Q ∪ {(v_i, v_k)};
11                 else
12                     Q ← Q ∪ {(v_k, v_i)};
13             t ← C(v_k, v_j) ∩ (C(v_k, v_i) ⋄ C(v_i, v_j));
14             if t ≠ C(v_k, v_j) then
15                 C(v_k, v_j) ← t; C(v_j, v_k) ← t^{-1};
16                 if k ≤ j then
17                     Q ← Q ∪ {(v_k, v_j)};
18                 else
19                     Q ← Q ∪ {(v_j, v_k)};
20     return N;
```

value of 0). Due to this inefficient (for the particular use case) storage design, the worst-case runtime of the $\diamond$-consistency enforcing algorithm can be increased to $O(n^4)$. Indeed, consider the case where in each iteration only a single constraint is modified by a reduction of its set of base relations. Thus, there can be at most $O(n^2|B|)$ iterations. This alone is not a problem, as we explained earlier in this section that each such modified constraint participates in $O(n)$ consistency operations which consider all triples of variables that involve that constraint. However, in each iteration we iterate $O(n^2)$ elements of the matrix and, thus, have an additional $O(n^2)$ overhead.

In the case where a queue is used, given a QCN $\mathcal{N} = (V, C)$ over n variables, we already saw that the queue is initialized with all the $O(n^2)$ pairs of variables in V which correspond to the constraints of $\mathcal{N}$. In particular, each entity of the queue correlates with a unique pair of variables corresponding to some constraint. A pair of variables can be added to the queue with the *push* operation, and removed from it with the *pop* operation. Hence, a queue allows us to both keep track of and iterate the minimum required number of constraints that need to be processed in a follow up consistency operation within a $\diamond$-consistency enforcing algorithm at any point of its execution. Howerer, a queue can also have its disadvantages. Using a queue as a data structure requires one to also maintain some additional data structure, such as a set or a $(0, 1)$-matrix, dedicated for fast membership testing of a pair of variables corresponding to a constraint. In particular, membership testing takes place when we want to check if a pair of variables corresponding to a constraint already exists in the queue, so that we do not add it again for revision. Since a queue is usually implemented as a doubly linked list or a modified dynamic array, membership testing of a single item costs $O(n^2)$ time. On the other hand, membership testing of a pair of variables will always take constant time when using a $(0, 1)$-matrix, and it

Table 3.1: Weighting scheme for Interval Algebra base relations

relation	eq	p	pi	m	mi	o	oi	s	si	d	di	f	fi
weight	1	3	3	2	2	4	4	2	2	4	3	2	2

will also take constant time on average when using a set of pairs of variables (although it can take $O(n^2)$ time in the worst case).

Ordering of Constraints. Regarding the selection of the next constraint to be processed, approaches have focused around queues and queuing strategies, and have varied from simply opting to operate a queue in a FIFO (first in first out) or LIFO (last in first out) manner, to applying some kind of specialized heuristic that selects the next constraint to be processed based on some preference value.

With respect to the simpler approach, i.e., opting to operate the queue in a FIFO or LIFO manner, none of the strategies has a clear advantage over the other. In general, a FIFO strategy is better when dealing with a queue that contains more than thousands or even millions of pairs of variables (this can easily be the case for a QCN over n variables with $n \geq 1000$), because it is important to make a first complete pass over the pairs of variables, perform all the consistency operations in which they are involved, and unveil any inconsistencies among triples of variables that might originally exist in the given QCN. On the other hand, a LIFO strategy can prove to be more beneficial for queues of smaller size. A pair of variables corresponding to a recently modified constraint which has been deemed more restrictive due to a reduction of its set of base relations, might be more prone to result in a reduction of the set of base relations of other constraint as well if used in a follow up consistency operation. Thus, pushing that pair of variables in the queue last and immediately reusing it by popping it for reprocessing first, might unveil an inconsistency faster. Due to each of the FIFO and LIFO strategies having its own benefits, it is common to combine both strategies during the execution of a $\diamond$-consistency enforcing algorithm. For example, one can start off with a FIFO strategy to make a first complete pass over the pairs of variables in the queue, and then repeatedly switch to a LIFO strategy whenever it feels necessary to focus more on some particular pairs of variables (e.g., pairs of variables for which the set of bease relations of the corresponding constraints was drastically reduced).

With respect to approaches that rely on some heuristic to select the next constraint to be processed, all of them are based on assigning a preference value to a constraint. These approaches are described in extent in [van Beek and Manchak, 1996; Renz and Nebel, 2001]. In particular, there is the weight heuristic, an estimate of how much the constraint corresponding to a pair of variables will restrict other constraints. Restrictiveness is measured for each base relation by successively composing the base relation with every possible relation, and then summing up the cardinalities of the resulting compositions. The result is then suitably scaled and a weight is assigned to each base relation. As an illustration, the weighting scheme for the base relations of Interval Algebra is shown in Table 3.1. The weight of a constraint is then given by the sum of the weights of its base relations. For example, the weight of the relation $\{p, o, f\}$ is 3 + 4 + 2 = 9. The smaller the weight of a relation, the more restrictive the relation is. The most restrictive relations are processed first; the reason for doing so is that they restrict the other relations on average most and, therefore, render them less likely to be processed repeatedly in the future. Chances are, that a relation which has lost less than half of its base relations due to some consistency operation will be processed more until the termination of the algorithm than a relation which has lost more than half of its base relations, likewise, due to some consistency

operation. In [Renz and Nebel, 2001], the authors exploit the fact that the RCC-8 constraint language has a limited number of relations, viz., 2^8 in total, and compute the exact restrictiveness by composing each relation (not just each base relation as in [van Beek and Manchak, 1996]) with every other relation and summing up the cardinalities of the resulting compositions. The result is then scaled into weights from 1 (the most restrictive relations) to 16 (the least restrictive relations). In its simplest form, the weight heuristic can correspond to the cardinality heuristic where the weight of every base relation is set to one.

Apart from the weight and cardinality heuristics, there also exists the constrainedness heuristic. The constrainedness heuristic is an estimate of how much a change in a relation will restrict other relations. It is determined as follows. Given a QCN $\mathcal{N} = (V, C)$ over n variables, suppose the pair of variables we are interested in is (v_i, v_j), with $v_i, v_j \in V$ and $0 \leq i \leq j < n$. The constrainedness of the relation corresponding to the pair of variables (v_i, v_j), viz., $C(v_i, v_j)$, is the sum of the weights of the relations $C(v_k, v_i)$ and $C(v_j, v_k)$, for every k such that $0 \leq k < n$. The intuition comes from examining the $\diamond$-consistency algorithm in Algorithm 1 which would propagate a change in relation $C(v_i, v_j)$. We see that $C(v_i, v_j)$ will be composed with $C(v_j, v_k)$ (line 6) and $C(v_k, v_i)$ (line 13), for every k such that $0 \leq k < n$. As a side note, heuristics that select the next constraint to be processed based on some preference value are usually implemented with the use of a priority queue data structure. A priority queue is like a regular queue data structure, but where additionally each element has a *priority* associated with it. In a priority queue, an entity with high priority is served before an entity with low priority. If two entities have the same priority, they are served according to their order in the queue. In the context of QCNs, a priority queue is most often implemented with a heap and coupled with a hash table data stucture that maps each pair of variables to an entry in the queue comprising the pair of variables itself and a weight denoting its priority. As such, given a QCN over n variables, using a priority queue in the $\diamond$-consistency enforcing algorithm would allow membership testing of an item to be performed in constant time and require a $O(log(n^2))$ cost in time for adding or removing an item.

Speeding Up Consistency Operations. Finally, in order to have a fast $\diamond$-consistency enforcing algorithm, we need to have fast weak composition, intersection, and converse operations.

Regarding the intersection operation, the relations of a qualitative constraint language are most efficiently implemented as bit vectors [van Beek and Manchak, 1996; Ladkin and Reinefeld, 1997]. In this way, the fast bit-wise AND operation can be used to perform the intersection operation. For example, if we assign the bit vectors 00000001 and 00000010 to base relations DC and EC of RCC-8 respectively, the bit vector 00000011 will correspond to relation $DC \cup EC$. Then, by intersecting 00000011 with 00000001, for example, we can immediately obtain base relation DC. Note that the size of the bit-vector is $|\mathsf{B}|$, with each of its bits corresponding to a base relation.

As noted in Section 2.4, the converse of a relation 2^{B} can be obtained from a converse table which stores the converse base relation b^{-1} for each base relation $b \in \mathsf{B}$. In fact, as most qualitative constraint languages have a small number of relations with respect to the available memory in modern computers that is required to store them, we do not need to only maintain the converse table for each base relation, but can maintain a larger converse table that stores the converse relation r^{-1} for each relation $r \in 2^{\mathsf{B}}$.

One can do something similar in the case of weak composition, i.e., she can opt to store the entire weak composition table considering the weak compositions between all the relations and not just the base relations. However, in this case, a 2-dimensional array would be required, which

would have a $O(2^{2|\mathsf{B}|})$ memory footprint (we consider here $|\mathsf{B}|$ to be a variable) and could pose a challenge even for todays modern computers (e.g., consider the RCC-23 calculus with 23 base relations, which in addition to RCC-8 allows reasoning about convexity [Cohn *et al.*, 1997]). In the case where storing a $2^{|\mathsf{B}|} \times 2^{|\mathsf{B}|}$ 2-dimensional array is impractical, we can consider Hogge's method [Hogge, 1987] which uses four small tables instead of a single large one, in particular, one with $2^{2\lceil|\mathsf{B}|/2\rceil}$ entries, two with $2^{|\mathsf{B}|}$ entries, and one with $2^{2\lfloor|\mathsf{B}|/2\rfloor}$ entries [Ladkin and Reinefeld, 1997]. The result of a weak composition can then be obtained by the union of the four array accesses plus 3 shift operations and some logical ANDs. At the cost of a slight increase of memory space, Hogge's method can be improved to use only two tables, one with $2^{|\mathsf{B}|+\lceil|\mathsf{B}|/2\rceil}$ entries holding the $\lceil|\mathsf{B}|/2\rceil$ low order weak compositions, and the other one with $2^{|\mathsf{B}|+\lfloor|\mathsf{B}|/2\rfloor}$ entries holding the $\lfloor|\mathsf{B}|/2\rfloor$ high order weak compositions [Ladkin and Reinefeld, 1997].

In the case where even Hogge's method is not directly applicable, caching for both weak compositions and converse results using hash tables can prove to be extremely beneficial as discussed in [Ladkin and Reinefeld, 1997; Gantner *et al.*, 2008; Westphal *et al.*, 2009]. In particular, since the number of entries in a hash table does not need to be directly dependent on the number of base relations in general, caching can be especially interesting for large calculi where full precomputations, as discussed above, are impossible. Also, as a $\diamond$-consistency enforcing algorithm is a fixed point algorithm, and therefore the actual weak compositions that have to be calculated tend to be clustered, even small hash tables can prove to be useful [Gantner *et al.*, 2008; Westphal *et al.*, 2009].

Hybrid Algorithms

A comprehensive comparative experimental study of different PC1-based and PC2-based $\diamond$-consistency enforcing algorithms, different data structures, and different heuristic implementations takes place in [Saade, 2008; Condotta *et al.*, 2006a]. This study also introduces new heuristics corresponding to a combination of the existing heuristics that we discussed earlier. In addition, a new algorithm is introduced, based on both PC1 and PC2. This algorithm firstly performs a main loop in which all triples of variables of a given QCN participate in consistency operations, in a similar way to PC1. Then, in a second phase, it operates in a similar way to PC2, with the exception that the dedicated data structure that is normally initialized with all the pairs of variables corresponding to the constraints of a given QCN that need to be processed, is initialized only with the pairs of variables corresponding to constraints that were modified during the first main PC1-like loop instead. This algorithm serves as a compromise between PC1 and PC2-based $\diamond$-consistency enforcing algorithms in terms of time and memory efficiency (e.g., PC1-based algorithms do not need a dedicated data structure for storing constraints to be processed). For more details, the reader may refer to the PhD Thesis of Saade [Saade, 2008].

3.5.2 Algorithms for the Satisfiability Problem of QCNs

In the case where a QCN is defined over a subclass of relations for which $\diamond$-consistency alone is able to decide satisfiability, and since algorithm WC is able to enforce $\diamond$-consistency on a given QCN of a relation algebra due to Proposition 6 (at page 37), we have the following result:

Proposition 7 ([Allen, 1983; Vilain *et al.*, 1990]) *Let $\mathcal{A} \in 2^{\mathsf{B}}$ be a subclass of relations of a qualitative constraint language that is a relation algebra, over which not trivially inconsistent and $\diamond$-consistent* QCN*s are satisfiable. Then, given a* QCN *$\mathcal{N} = (V, C)$ defined over $\mathcal{A}$, algorithm* WC *terminates and returns $\perp^V$ if and only if $\mathcal{N}$ is unsatisfiable.*

Algorithm 2: Consistency($\mathcal{N}$, $\mathcal{A}$)

```
in      : A QCN N = (V, C), and a subclass A.
output  : Null, or a refinement of network N over A.
1  begin
2      N ← WC(N);
3      if N = ⊥^V then
4          return Null;
5      if ∀v_i, v_j ∈ V with i < j we have that C(v_i, v_j) ∈ A then
6          return N;
7      choose a constraint C(v_i, v_j) with v_i, v_j ∈ V and i < j such that C(v_i, v_j) ∉ A;
8      split C(v_i, v_j) into r_1, ..., r_k ∈ A: r_1 ∪ ... ∪ r_k = C(v_i, v_j);
9      foreach r ∈ {r_l | 1 ≤ l ≤ k} do
10         N[v_i, v_j] ← r; N[v_j, v_i] ← r^-1;
11         result ← Consistency(N, A);
12         if result ≠ Null then
13             return result;
14     return Null;
```

From Proposition 7 we can assert the following theorem:

Theorem 3 ([Allen, 1983; Vilain *et al.*, 1990]) *Let $\mathcal{A} \in 2^{\mathsf{B}}$ be a subclass of relations of a qualitative constraint language that is a relation algebra, over which not trivially inconsistent and $\diamond$-consistent* QCN*s are satisfiable. Then, given a* QCN *$\mathcal{N} = (V, C)$ defined over $\mathcal{A}$, algorithm* WC *is sound and complete for deciding the satisfiability of $\mathcal{N}$.*

Then, due to Theorem 3, and Propositions 1 (at page 21), and 5 (at page 35), we have the following result:

Corollary 3 *Let $\mathcal{N} = (V, C)$ be a* QCN *of Point Algebra, Cardinal Direction Calculus, Interval Algebra, Block Algebra, or* RCC*-8, defined over one of the classes $\mathcal{P}_{\mathsf{PA}}$, $\mathcal{P}_{\mathsf{CDC}}$, $\mathcal{H}_{\mathsf{IA}}$, $\mathcal{H}^n_{\mathsf{IA}}$, or $\hat{\mathcal{H}}_8$, $\mathcal{C}_8$, and $\mathcal{Q}_8$ respectively. We have that algorithm* WC *is sound and complete for deciding the satisfiability of $\mathcal{N}$.*

For the general case of QCNs, i.e., for QCNs that are not strictly defined over subclasses of relations for which $\diamond$-consistency alone is able to decide satisfiability, some kind of backtracking algorihm must be used that operates on the basis of splitting the relations of a given QCN into disjunctions of relations belonging to a tractable subclass for which $\diamond$-consistency alone is indeed able to decide satisfiability. Such a backtracking-based approach has been presented in the context of qualitative spatial and temporal reasoning in [Nebel, 1997; Ladkin and Reinefeld, 1992; Renz and Nebel, 2001]. This approach was then adopted for solving problems of *simple temporal constraints* [Dechter *et al.*, 1991] in [Stergiou and Koubarakis, 1998; Stergiou and Koubarakis, 2000]. A nice theoretical evaluation of selected general backtracking algorithms is presented in [Kondrak and van Beek, 1997; Kondrak and van Beek, 1995].

The Consistency Algorithm

An algorithm that implements a backtracking search for solving general QCNs is provided in Algorithm 2, called Consistency. Algorithm Consistency receives as input a QCN $\mathcal{N} = (V, C)$,

and a subclass of relations $\mathcal{A}$. First, $\diamond$-consistency is enforced through the WC algorithm (line 2), which we described in detail in Section 3.5.1. Then, granted that enforcing $\diamond$-consistency did not result in a trivially inconsistent network, algorithm Consistency chooses a constraint $C(v_i, v_j)$ with $v_i, v_j \in V$ and $i < j$ such that $C(v_i, v_j) \notin \mathcal{A}$ (line 7). In the case where such a constraint does not exist, a not trivially inconsistent and $\diamond$-consistent refinement of network $\mathcal{N}$ defined over subclass $\mathcal{A}$ is returned as a solution. In any other case, constraint $C(v_i, v_j)$ is split into subrelations $r_1, ..., r_k \in \mathcal{A}$, for which we have that $r_1 \cup ... \cup r_k = C(v_i, v_j)$ (line 8). Each of these relations is instantiated accordingly to the constraint network $\mathcal{N}$ (line 10) and a recursive call is initiated (line 11). When the algorithm termitates, it is guaranteed to return either Null (line 14), or a not trivially inconsistent and $\diamond$-consistent refinement of network $\mathcal{N}$ defined over subclass $\mathcal{A}$ (line 13). Note that except for the first step, in all subsequent recursive calls of the Consistency algorithm, the WC algorithm only has to be run for the paths that are possibly affected by each prior instantiation, as explained in [Renz and Nebel, 2001; Gantner *et al.*, 2008], which takes $O(n^2)$ intersections and weak compositions, where $n = |V|$. (This detail is not included in the WC algorithm.) To obtain this constraint incremental variation of the WC algorithm, it suffices to replace the command $Q \leftarrow \{(v_i, v_j) \mid 0 \leq i \leq j < n\}$ in line 2 of the WC algorithm, with the command $Q \leftarrow \{(v_i, v_j)\}$, where v_i, v_j are the variables appearing in the constraint $C(v_i, v_j)$ in line 10 of the Consistency algorithm. As such, the WC algorithm can be seen as approximating consistency for general QCNs, but also as realising *forward checking* [Dechter, 2003] in the backtracking algorithm by closing all triples of variables of a given QCN under weak composition and eliminating base relations that are unfeasible.

Heuristics. It is most common, if not necessary, to implement some heuristics regarding constraint (line 7) and subrelation (line 9) selection, as they can overall improve the efficiency of the backtracking algorithm [Renz and Nebel, 2001; van Beek and Manchak, 1996]. These involve the cardinality heuristic for constraint selection, and the weight heuristic for subrelation selection. With respect to the cardinality heuristic for constraint selection, a constraint with the smallest number of subrelations (as specified in line 8 of the algorithm) is preferred among others. When selecting a constraint, the cardinality heuristic can be further influenced by the weights of the subrelations of a constraint (where the weights are assigned to the relations in a similar manner to what we described for the $\diamond$-consistency enforcing algorithm). In particular, the weights can function as a tie-breaker for constraints sharing the same cardinality. With respect to the weight heuristic for subrelation selection, the least restrictive subrelation is selected according to its weight. In fact, the cardinality heuristic for constraint selection and the weight heuristic for subrelation selection correspond exactly to the minimum remaining values (MRV) heuristic and the least-constraining value heuristic in the CSP framework respectively, both of which are described in detail in [Russell and Norvig, 2010]. Further, the tie-breaker used in the cardinality heuristic for constraint selection can be seen as the degree heuristic [Russell and Norvig, 2010], again, in the CSP framework. The cardinality heuristic for constraint selection can also be combined with a dynamic weight as a boost in the way that it is described by Boussemart et al. [Boussemart *et al.*, 2004]. In particular, in the context of QCNs, when a relation is reduced to the empty relation, dynamic weights are increased based on the depth of the search tree (exactly as in Mistral [5], a well-known CSP solver). Further, last conflict-based reasoning can be performed to reduce the amount of thrashing [Lecoutre *et al.*, 2006; Lecoutre *et al.*, 2009], and eligible and frozen constraints [Condotta *et al.*, 2007] can be used to reduce the search space. In particular, the concept of eligibility characterizes constraints that will not be used during the search, and

[5] `http://homepages.laas.fr/ehebrard/mistral.html`

the concept of frozen constraints describes exactly those constraints that can be instantiated to a fixed value (thus, those constraints that can be frozen) in order to avoid uneccessary reprocessing and updates. Finally, restart and nogood recording techniques can be used in the domain of qualitative spatial and temporal reasoning in the way of [Katsirelos and Bacchus, 2005; Lecoutre *et al.*, 2007]. In fact, such techniques are applied orthogonally to the usual backtracking-based approach for solving qualitative constraint satisfaction problem in [Westphal and Hué, 2012; Westphal *et al.*, 2010], where first evaluations show promising results.

Given a QCN $\mathcal{N} = (V, C)$, and a subclass $\mathcal{A}$, the search space for algorithm Consistency is $O(\alpha^{|V|^2})$, where α is the branching factor (see line 8 of the Consistency algorithm) provided by subclass $\mathcal{A}$ (e.g., $\alpha = 1.4375$ for subclass $\hat{\mathcal{H}}_8$ for RCC-8 [Renz and Nebel, 2001]). With respect to the Consistency algorithm, we have the following result:

Proposition 8 ([Nebel, 1997; Renz and Nebel, 2001]) *Let $\mathcal{A} \in 2^{\mathsf{B}}$ be a subclass of relations of a qualitative constraint language that is a relation algebra, over which not trivially inconsistent and $\diamond$-consistent* QCN*s are satisfiable. Then, given $\mathcal{A}$, and a* QCN *$\mathcal{N}$ defined over 2^{B}, algorithm* Consistency *terminates and returns* Null *if and only if $\mathcal{N}$ is unsatisfiable.*

From Proposition 8 we can assert the following theorem:

Theorem 4 ([Nebel, 1997; Renz and Nebel, 2001]) *Let $\mathcal{A} \in 2^{\mathsf{B}}$ be a subclass of relations of a qualitative constraint language that is a relation algebra, over which not trivially inconsistent and $\diamond$-consistent* QCN*s are satisfiable. Then, given $\mathcal{A}$, and a* QCN *$\mathcal{N}$ defined over 2^{B}, we have that algorithm* Consistency *is sound and complete for deciding the satisfiability of $\mathcal{N}$.*

Then, due to Theorem 4, and Propositions 1 (at page 21), and 5 (at page 35), we have the following result:

Corollary 4 *Given one of the classes $\mathcal{P}_{\mathsf{CDC}}$, $\mathcal{H}_{\mathsf{IA}}$, $\mathcal{H}^n_{\mathsf{IA}}$, or $\hat{\mathcal{H}}_8$, $\mathcal{C}_8$, and $\mathcal{Q}_8$, and a* QCN *$\mathcal{N}$ of Cardinal Direction Calculus, Interval Algebra, Block Algebra, or* RCC*-8 respectively, we have that algorithm* Consistency *is sound and complete for deciding the satisfiability of $\mathcal{N}$.*

The IterativeConsistency Algorithm

We also present an additional algorithm, which is the iterative counterpart of the recursive chronological backtracking algorithm Consistency. The recursive and iterative algorithms are functionally equivalent. However, the iterative algorithm does not suffer from a recursion depth or a recursion stack limit. Further, its structure, although a little more complex, makes it easier to fine-tune it, performance-wise. We call the iterative algorithm IterativeConsistency, presented in Algorithm 3. Again, we note that Consistency and IterativeConsistency are functionally equivalent. The advantage of IterativeConsistency over Consistency relies on the fact that it does not suffer from a recursion depth or a recursion stack limit and it avoids the overhead of costly recursive calls and call stack management altogether. In particular, and in the case where Python is used, we have experimentally verified that the performance of an implementation of the IterativeConsistency algorithm is in general around 5% better than the performance of an implementation of the Consistency algorithm.

Algorithm 3: IterativeConsistency($\mathcal{N}$, $\mathcal{A}$)

```
in     : A QCN N = (V, C), and a subclass A.
output : Null, or a refinement of network N over A.
1  begin
2      Stack ← list(∅) // Initialize stack;
3      N ← WC(N);
4      if N = ⊥^V then
5          return Null;
6      while True do
7          if ∀v_i, v_j ∈ V with i < j we have that C(v_i, v_j) ∈ A then
8              return N;
9          choose a constraint C(v_i, v_j) with v_i, v_j ∈ V and i < j such that C(v_i, v_j) ∉ A;
10         split C(v_i, v_j) into r_1, ..., r_k ∈ A: r_1 ∪ ... ∪ r_k = C(v_i, v_j);
11         r_values ← {r_l | 1 ≤ l ≤ k};
12         while True do
13             if r_values = ∅ then
14                 while Stack ≠ ∅ do
15                     N, r_values ← Stack.pop();
16                     if r_values then
17                         break;
18                 if r_values = ∅ and Stack = ∅ then
19                     return Null;
20             r ← r_values.pop();
21             N' ← N;
22             N[v_i, v_j] ← r; N[v_j, v_i] ← r^-1;
23             N ← WC(N);
24             if N ≠ ⊥^V then
25                 break;
26             N ← N';
27         Stack.append(N', r_values);
```

Reasoners

We close this section with a small discussion on state of the art reasoners for solving general QCNs. The first robust reasoners for QCNs of Interval Algebra and QCNs of RCC-8 appeared in [Nebel, 1997] (viz., Nebel's solver) and [Renz, 2002b; Renz and Nebel, 2001] (viz., Renz's solver), respectively. These reasoners were later superseded by GQR, a generic qualitative constraint reasoner that supports arbitrary qualitative constraint languages [Westphal *et al.*, 2009; Gantner *et al.*, 2008]. For this purpose, any qualitative constraint language can be fed to GQR through a description of its base relations, its identity relation, and the weak composition and converse operations that are associated with its base relations. Additionally, tractable subclasses of relations for a given qualitative constraint language can be specified. GQR stores a QCN over n variables using a $n \times n$ constraint matrix, much like Nebel's and Renz's solvers. With respect to this aspect, GQR handles the constraint matrix that represents a qualitative constraint network more efficiently during backtracking search, i.e., it does not create a copy of the matrix at each forward step of the backtracking algorithm (as is the case with Nebel's and Renz's solvers), but it only keeps track of the constraints that are altered at each forward step to be able to

Algorithm 4: Minimize($\mathcal{N}$)

in : A QCN $\mathcal{N} = (V, C)$ defined over a subclass of relations of a qualitative constraint language that is a relation algebra, over which not trivially inconsistent and $\diamond$-consistent QCNs are satisfiable.
output : $\mathcal{N}_{\mathsf{min}}$.

```
1 begin
2     N' ← N;
3     foreach v_i, v_j ∈ V with 0 ≤ i ≤ j < |V| do
4         foreach b ∈ N'[v_i, v_j] do
5             if b ∉ WC(N'_{[v_i,v_j]/{b}})[v_i, v_j] then
6                 N[v_i, v_j] ← N[v_i, v_j] \ {b};
7                 N[v_j, v_i] ← N[v_j, v_i] \ {b^-1};
8     return N;
```

reconstruct the matrix in the case of backtracking. This alone has a tremendous positive effect on the performance of GQR when compared with the performance of Nebel's or Renz's solver. In addition, the search algorithms and heuristics in GQR have been fine-tuned over several years of active development and are more efficient than the ones met in Nebel's or Renz's solver. In fact, given the fact that Nebel's and Renz's solvers were implemented many years ago, parts of their algorithms are simply outdated given the advances in constraint programming. GQR incorporates many of the techniques that we have discussed in this chapter, as well as several techniques from the CSP and SAT communities. More details on GQR can be found in the PhD Thesis of Matthias Westphal [Westphal, 2014].

3.5.3 Algorithms for the Minimal Labeling Problem of QCNs

In the case where a QCN is defined over a subclass of relations for which $\diamond$-consistency alone is able to guarantee minimality, and since algorithm WC is able to enforce $\diamond$-consistency on a given QCN of a relation algebra due to Proposition 6 (at page 37), we have the following result:

Proposition 9 *Let $\mathcal{A} \in 2^{\mathsf{B}}$ be a subclass of relations of a qualitative constraint language that is a relation algebra, over which not trivially inconsistent and $\diamond$-consistent* QCN*s are minimal. Then, given a satisfiable* QCN $\mathcal{N} = (V, C)$ *defined over $\mathcal{A}$, algorithm* WC *terminates and returns $\mathcal{N}_{\mathsf{min}}$ (viz., the unique equivalent minimal sub-*QCN *of $\mathcal{N}$).*

Then, due to Proposition 9 and Theorem 2 (at page 35), we have the following result:

Proposition 10 ([Long and Li, 2015]) *Let $\mathcal{N} = (V, C)$ be a satisfiable* QCN *defined over a distributive subclass of relations of a qualitative constraint language that is a relation algebra and for which every $\diamond$-consistent atomic* QCN *is satisfiable. Then, algorithm* WC *terminates and returns $\mathcal{N}_{\mathsf{min}}$.*

Finally, due to Proposition 10, and Propositions 1 (at page 21) and 3 (at page 34), we have the following result:

Corollary 5 *Let $\mathcal{N} = (V, C)$ be a satisfiable* QCN *defined over a distributive subclass of relations of Point Algebra, Cardinal Direction Calculus, Interval Algebra, Block Algebra, or* RCC*-8. We have that algorithm* WC *terminates and returns $\mathcal{N}_{\mathsf{min}}$.*

Algorithm 5: MinimizeSDCM($\mathcal{N}$,$\mathcal{A}$)

in : $\mathcal{N} = (V, C)$ a QCN on 2^{B}, $\mathcal{A}$ a subclass of 2^{B}.
output : A sub-QCN of $\mathcal{N}$.

1 **begin**
  // **Step 1:** Initialization
2 $\mathcal{N}_{\mathsf{init}} \leftarrow \mathcal{N}$;
3 $\mathcal{N}_{\mathsf{F}} \leftarrow \bot^V$; $\mathcal{N}_{\mathsf{nonF}} \leftarrow \bot^V$;
4 $\mathcal{N} \leftarrow \diamond(\mathcal{N})$;
5 $\mathcal{N}_{\mathsf{nonF}} \leftarrow \mathcal{N}_{\mathsf{init}} \setminus \mathcal{N}$;
6 **if** $\mathcal{N} = \bot^V$ **then**
7 **return** $\bot^V$;
  // **Step 2:** Minimization
8 **while** $\{v, v' \in V \mid (\mathcal{N}[v, v'] \setminus \mathcal{N}_{\mathsf{F}}[v, v']) \neq \emptyset\} \neq \emptyset$ **do**
9 Select $v, v' \in V$ such that $(\mathcal{N}[v, v'] \setminus \mathcal{N}_{\mathsf{F}}[v, v']) \neq \emptyset$;
10 $r \leftarrow \mathcal{N}[v, v'] \setminus \mathcal{N}_{\mathsf{F}}[v, v']$;
11 $\mathcal{N}' \leftarrow \mathsf{MinSubQCN}(\mathcal{N}, \mathcal{N}_{\mathsf{init}})$;
12 **if** $\mathcal{N}' = \mathsf{Null}$ **then**
13 $\mathcal{N}_{\mathsf{nonF}}[v, v'] \leftarrow \mathcal{N}_{\mathsf{nonF}} \cup r$;
14 $\mathcal{N}_{\mathsf{nonF}} \leftarrow (\mathcal{N}_{\mathsf{nonF}}[v, v'])^{-1}$;
15 **else**
16 $\mathcal{N}_{\mathsf{F}}[v, v'] \leftarrow \mathcal{N}_{\mathsf{F}} \cup \mathcal{A}(\mathcal{N}')$;
17 $\mathcal{N} \leftarrow \mathcal{N}_{\mathsf{init}} \setminus \mathcal{N}_{\mathsf{nonF}}$;
18 **return** $\mathcal{N}_{\mathsf{F}}$;// **Step 3: Return of the result**

In the case where a QCN is defined over a subclass of relations for which $\diamond$-consistency alone is able to decide satisfiability, and since algorithm WC is able to enforce $\diamond$-consistency on a given QCN of a relation algebra due to Proposition 6 (at page 37), we can devise an algorithm based on WC that will extract the unique equivalent minimal sub-QCN of that QCN by iterating all of its constraints and minimizing them. Such an algorithm is presented in Algorithm 4, called Minimize. We have the following result:

Proposition 11 ([Gerevini and Renz, 2002]) *Let $\mathcal{A} \in 2^{\mathsf{B}}$ be a subclass of relations of a qualitative constraint language that is a relation algebra, over which not trivially inconsistent and $\diamond$-consistent* QCN*s are satisfiable. Then, given a satisfiable* QCN *$\mathcal{N} = (V, C)$ defined over $\mathcal{A}$, algorithm* Minimize *terminates and returns $\mathcal{N}_{\mathsf{min}}$.*

Then, due to Proposition 11, and Propositions 1 (at page 21), and 5 (at page 35), we have the following result:

Corollary 6 *Let $\mathcal{N} = (V, C)$ be a satisfiable* QCN *of Point Algebra, Cardinal Direction Calculus, Interval Algebra, Block Algebra, or* RCC-*8, defined over one of the classes $\mathcal{P}_{\mathsf{PA}}$, $\mathcal{P}_{\mathsf{CDC}}$, $\mathcal{H}_{\mathsf{IA}}$, $\mathcal{H}^n_{\mathsf{IA}}$, or $\hat{\mathcal{H}}_8$, $\mathcal{C}_8$, and $\mathcal{Q}_8$ respectively. We have that algorithm* Minimize *terminates and returns $\mathcal{N}_{\mathsf{min}}$.*

Given a QCN $\mathcal{N} = (V, C)$ over n variables, algorithm Minimize runs in $O(n^5)$ time. Minimize iterates a number of $O(n^2)$ constraints and calls algorithm WC for each one of the $O(|\mathsf{B}|)$ base relations of a constraint; each such call requiring $O(n^3)$ time.

Algorithm 6: MinSubQCN($\mathcal{N}$,$\mathcal{N}_{\mathsf{init}}$)

in : Two QCNs $\mathcal{N}$=(V, C) and $\mathcal{N}_{\mathsf{init}}$=$(V, C_{\mathsf{init}})$ such that $\mathcal{N} \subseteq \mathcal{N}_{\mathsf{init}}$.
output : Null, or a $\diamond$-consistent sub-QCN $\mathcal{N}'$ with $\mathcal{A}(\mathcal{N}') \subseteq \mathcal{N}_{\mathsf{init}}$.

```
1  begin
2      N' ← WC(N);
3      if ∃v_i, v_j ∈ V such that N'[v_i, v_j] = ∅ then
4          return Null;
5      if ∀v_i, v_j ∈ V we have that A(N'[v_i, v_j]) ⊆ N_init[v_i, v_j] then
6          return N';
7      choose a constraint N'[v_i, v_j] with v_i, v_j ∈ V such that A(N'[v_i, v_j]) ⊈ N_init[v_i, v_j];
8      split N'[v_i, v_j] into r_1, ..., r_k ∈ A: r_1 ∪ ... ∪ r_k = N'[v_i, v_j];
9      foreach r ∈ {r_l | 1 ≤ l ≤ k} do
10         N'[v_i, v_j] ← r; N'[v_j, v_i] ← r^-1;
11         result ← MinSubQCN(N',N_init);
12         if result ≠ Null then
13             return result;
14     return Null;
```

For the general case of QCNs, i.e., for QCNs that are not strictly defined over subclasses of relations for which $\diamond$-consistency alone is able to decide satisfiability or guarantee minimality, Amaneddine and Condotta in [Amaneddine and Condotta, 2013] present an algorithm that essentially decomposes a given QCN $\mathcal{N}$ into a set of QCNs $\{\diamond(\mathcal{N}_1), \diamond(\mathcal{N}_2), \ldots, \diamond(\mathcal{N}_k)\}$, where k is some positive integer, such that $\diamond(\mathcal{N}_1) + \diamond(\mathcal{N}_2) + \ldots + \diamond(\mathcal{N}_k)$ equals a sub-QCN of $\mathcal{N}$ and $\diamond(\mathcal{N}_i)$, with $i \in \{1, 2, \ldots, k\}$, is a QCN defined over the subclass $\mathcal{A} \in 2^{\mathsf{B}}$ given as parameter. This algorithm is called MinimizeSDCM and it is presented in Algorithm 5. Every such not trivially inconsistent and $\diamond$-consistent QCN $\diamond(\mathcal{N}_i)$ that we mentioned earlier, is extracted through a call to algorithm MinSubQCN, presented in Algorithm 6. Note that this function is very similar to the one presented in Section 3.5.2 for solving the satisfiability problem of a QCN. With respect to the reasoning behind algorithm MinimizeSDCM and Proposition 6 (at page 37), we are provided with the following result:

Theorem 5 ([Amaneddine and Condotta, 2013]) *Given a satisfiable* QCN $\mathcal{N} = (V, C)$ *and a subclass* $\mathcal{A} \subseteq 2^{\mathsf{B}}$ *of relations of a qualitative constraint language that is a relation algebra and over which not trivially inconsistent and* $\diamond$*-consistent* QCN*s are minimal, we have that algorithm* MinimizeSDCM*, with* $\mathcal{N}$ *and* $\mathcal{A}$ *as parameters, returns* $\mathcal{N}_{\mathsf{min}}$.

Then, due to Theorems 5 and 2 (at page 35) and the reasoning behind algorithm MinimizeSDCM, we have the following result:

Theorem 6 ([Long and Li, 2015; Amaneddine and Condotta, 2013]) *Given a satisfiable* QCN $\mathcal{N} = (V, C)$ *and a distributive subclass* $\mathcal{A} \subseteq 2^{\mathsf{B}}$ *of relations of a qualitative constraint language that is a relation algebra and for which every* $\diamond$*-consistent atomic* QCN *is satisfiable, we have that algorithm* MinimizeSDCM*, with* $\mathcal{N}$ *and* $\mathcal{A}$ *as parameters, returns* $\mathcal{N}_{\mathsf{min}}$.

Finally, due to Theorem 6, and Propositions 1 (at page 21) and 3 (at page 34), we have the following result:

Algorithm 7: Delphys($\mathcal{N}$)

in : An all-different QCN $\mathcal{N}$ defined over a distributive subclass of relations of a qualitative constraint language that is a relation algebra and for which every $\diamond$-consistent atomic QCN is satisfiable.

output : χ, the set of non-redundant relations in $\diamond(\mathcal{N})$.

```
1  begin
2      χ ← ∅;
3      N' ← WC(N);
4      Q ← {(v_i, v_j) | v_i, v_j ∈ V with 0 ≤ i < j < |V|};
5      while Q ≠ ∅ do
6          (v_i, v_j) ← Q.pop();
7          τ ← ∅;
8          foreach v_k ∈ V \ {v_i, v_j} do
9              t ← N'[v_i, v_k] ◇ N'[v_k, v_j];
10             foreach b ∈ B do
11                 if b ∉ t then
12                     τ ← τ ∪ {b};
13         if τ ∪ N'[v_i, v_j] ≠ B then
14             χ ← χ ∪ {N'[v_i, v_j]};
15     return χ;
```

Corollary 7 *Given a satisfiable* QCN $\mathcal{N} = (V, C)$ *of Point Algebra, Cardinal Direction Calculus, Interval Algebra, Block Algebra, or* RCC-*8 and a distributive subclass* $\mathcal{A} \subseteq 2^{\mathsf{B}}$ *of relations of the respective qualitative constraint language, we have that algorithm* MinimizeSDCM*, with* $\mathcal{N}$ *and* $\mathcal{A}$ *as parameters, terminates and returns* $\mathcal{N}_{\mathsf{min}}$.

3.5.4 Algorithms for the Redundancy Problem of QCNs

In the case where a QCN $\mathcal{N}$ is defined over a distributive subclass of relations, and since algorithm WC is able to enforce $\diamond$-consistency on a given QCN of a relation algebra due to Proposition 6 (at page 37), we can devise an algorithm based on WC that will extract the set of non-redundant constraints in the closure under weak composition of $\mathcal{N}$, viz., $\diamond(\mathcal{N})$, by implementing Lemma 2 (at page 36). Such an algorithm is presented in Algorithm 7, called Delphys. We have the following result with respect to Delphys:

Proposition 12 ([Li *et al.*, 2015a]) *Let* $\mathcal{N} = (V, C)$ *be an all-different* QCN *defined over a distributive subclass of relations of a qualitative constraint language that is a relation algebra and for which every* $\diamond$*-consistent atomic* QCN *is satisfiable. Then, algorithm* Delphys *terminates and returns the set of non-redundant relations in* $\diamond(\mathcal{N})$.

Then, due to Proposition 12, and Propositions 1 (at page 21) and 3 (at page 34), we have the following result:

Corollary 8 *Let* $\mathcal{N} = (V, C)$ *be a* QCN *defined over a distributive subclass of relations of Point Algebra, Cardinal Direction Calculus, Interval Algebra, Block Algebra, or* RCC-*8. We have that algorithm* Delphys *terminates and returns the set of non-redundant relations in* $\diamond(\mathcal{N})$.

Algorithm 8: extractPrimeQCN($\mathcal{N}$)

in : A QCN $\mathcal{N} = (V, C)$ defined over a subclass of relations of a qualitative constraint language that is a relation algebra, over which not trivially inconsistent and $\diamond$-consistent QCNs are satisfiable.
output : A prime QCN of $\mathcal{N}$.

```
1  begin
2      C' ← map({((v, v') : (B if v ≠ v' else {Id})) | v, v' ∈ V});
3      Q ← {(v_i, v_j) | v_i, v_j ∈ V with 0 ≤ i < j < |V|};
4      while Q ≠ ∅ do
5          (v_i, v_j) ← Q.pop();
6          flag ← True;
7          foreach b ∈ B \ C(v_i, v_j) do
8              if WC(N_[v_i,v_j]/{b}) ≠ ⊥^V then
9                  C'(v_i, v_j) ← C(v_i, v_j); C'(v_j, v_i) ← (C(v_i, v_j))^-1;
10                 flag ← False; break;
11         if flag = True then
12             C(v_i, v_j) = C(v_j, v_i) = B;
13     return (V, C');
```

Clearly, given a QCN $\mathcal{N} = (V, C)$ over n variables, algorithm Delphys runs in $O(n^3)$ time as the WC algorithm call in line 3 dominates the overall execution time.

We can construct the prime QCN of a given QCN of Point Algebra, Cardinal Direction Calculus, or RCC-8 by exploiting the following result:

Proposition 13 ([Li *et al.*, 2015a]) *Let $\mathcal{N} = (V, C)$ be an all-different* QCN *defined over a distributive subclass of relations of Point Algebra, Cardinal Direction Calculus, or* RCC-*8. Then, we have that $\forall u, v \in V$ a constraint $\mathcal{N}[u, v]$ is redundant in $\mathcal{N}$ if and only if $\diamond(\mathcal{N})[u, v]$ is redundant in $\diamond(\mathcal{N})$.*

In particular, due to Propositions 12, 13, 1 (at page 21), and 3 (at page 34), and by taking into account the fact that given a QCN $\mathcal{N}$ we have that $\mathcal{N}_{\mathsf{min}}$ is unique, we obtain the following result that allows us to extract the unique prime QCN out of a QCN that satisfies certain conditions:

Theorem 7 ([Li *et al.*, 2015a]) *Let $\mathcal{N} = (V, C)$ be an all-different* QCN *defined over a distributive subclass of relations of Point Algebra, Cardinal Direction Calculus, or* RCC-*8. Further, let χ be the set of non-redundant relations in $\diamond(\mathcal{N})$. Then, $\forall u, v \in V$, we have that $\mathcal{N}_{\mathsf{prime}}[u, v] = \mathcal{N}[u, v]$ if $\diamond(\mathcal{N})[u, v] \in \chi$ and $\mathcal{N}_{\mathsf{prime}}[u, v] =$ (B if $u \neq v$ else {Id}) otherwise.*

However, as already noted earlier, the aforementioned construction of a unique prime QCN out of a given QCN is only possible for QCNs that satisfy certain conditions, i.e., QCNs defined over a distributive subclass of relations of Point Algebra, Cardinal Direction Calculus, or RCC-8. For QCNs of other qualitative constraint languages, or for QCNs defined over non-distributive subclasses of relations, we have to follow a different approach for extracting their (non-unique in the general case) prime networks. Let us consider the case where a QCN $\mathcal{N} = (V, C)$ is defined over a subclass of relations for which $\diamond$-consistency alone is able to decide satisfiability. To test if a constraint $C(u, v)$ for some $u, v \in V$ is non-redundant in $\mathcal{N}$, we need to check if there exists a base relation $b \in \mathsf{B} \setminus C(u, v)$ such that the QCN $\mathcal{N}_{[u,v]/\{b\}}$ is satisfiable; this satisfiability check

Algorithm 9: extractPrimeQCNSC($\mathcal{N}$,$\mathcal{A}$)

in : A QCN $\mathcal{N} = (V, C)$, and a subclass $\mathcal{A}$.
output : A QCN (V, C').

```
1  begin
2      C' ← map({((v, v') : (B if v ≠ v' else {Id})) | v, v' ∈ V});
3      Q ← {(v_i, v_j) | v_i, v_j ∈ V with 0 ≤ i < j < |V|};
4      while Q ≠ ∅ do
5          (v_i, v_j) ← Q.pop();
6          r ← B \ C(v_i, v_j);
7          if Consistency(N_{[v_i,v_j]/r}, A) ≠ Null then
8              C'(v_i, v_j) ← C(v_i, v_j); C'(v_j, v_i) ← (C(v_i, v_j))^{-1};
9          else
10             C(v_i, v_j) = C(v_j, v_i) = B;
11     return (V, C');
```

can be achieved through $\diamond$-consistency. We can apply this procedure iteratively to construct a prime QCN out of a given QCN and, thus, devise algorithm extractPrimeQCN, presented in Algorithm 8. We have the following result:

Proposition 14 ([Li *et al.*, 2015a]) *Let $\mathcal{A} \in 2^{\mathsf{B}}$ be a subclass of relations of a qualitative constraint language that is a relation algebra, over which not trivially inconsistent and $\diamond$-consistent* QCN*s are satisfiable. Then, given a* QCN $\mathcal{N} = (V, C)$ *defined over $\mathcal{A}$, algorithm* extractPrimeQCN *terminates and returns a prime* QCN *of $\mathcal{N}$.*

Then, due to Proposition 14, and Propositions 1 (at page 21), and 5 (at page 35), we have the following result:

Corollary 9 *Let $\mathcal{N} = (V, C)$ be a* QCN *of Point Algebra, Cardinal Direction Calculus, Interval Algebra, Block Algebra, or* RCC-*8, defined over one of the classes $\mathcal{P}_{\mathsf{PA}}$, $\mathcal{P}_{\mathsf{CDC}}$, $\mathcal{H}_{\mathsf{IA}}$, $\mathcal{H}^n_{\mathsf{IA}}$, or $\hat{\mathcal{H}}_8$, $\mathcal{C}_8$, and $\mathcal{Q}_8$ respectively. We have that algorithm* extractPrimeQCN *terminates and returns a prime* QCN *of $\mathcal{N}$.*

Given a QCN $\mathcal{N} = (V, C)$ over n variables, algorithm extractPrimeQCN runs in $O(n^5)$ time. The extractPrimeQCN algorithm iterates a number of $O(n^2)$ constraints and calls algorithm WC for each one of the $O(|\mathsf{B}|-1)$ base relations not belonging to a constraint; each such call requiring $O(n^3)$ time. It should be clear that the construction of a prime network depends on the order in which the constraints are processed, as we already noted that a prime network of a given QCN is in general not unique.

In the case where a QCN $\mathcal{N} = (V, C)$ is defined over 2^{B}, we have to devise a different algorithm than the one presented earlier. Such an algorithm will build on the Consistency algorithm presented in Algorithm 2 and will use a subclass of relations for which $\diamond$-consistency alone is able to decide satisfiability. To test if a constraint $C(u, v)$ for some $u, v \in V$ is non-redundant in $\mathcal{N}$, we need to check if the QCN that results by replacing relation $C(u, v)$ with relation $r = \mathsf{B} \setminus C(u, v)$ in $\mathcal{N}$ is satisfiable; this satisfiability check can be achieved through the Consistency algorithm along with a subclass of relations for which $\diamond$-consistency alone is able to decide satisfiability. We can apply this procedure iteratively to construct a prime QCN out of a given QCN and, thus, devise algorithm extractPrimeQCNSC, presented in Algorithm 9. We have the following result:

Proposition 15 ([Li *et al.*, 2015a]) *Let $\mathcal{A} \in 2^{\mathsf{B}}$ be a subclass of relations of a qualitative constraint language that is a relation algebra, over which not trivially inconsistent and $\diamond$-consistent* QCN*s are satisfiable. Then, given $\mathcal{A}$, and a* QCN $\mathcal{N}$ *defined over 2^{B}, we have that algorithm* extractPrimeQCNSC *terminates and returns a prime* QCN *of $\mathcal{N}$.*

Then, due to Propositions 15, 1 (at page 21), and 5 (at page 35), we have the following result:

Corollary 10 *Given one of the classes $\mathcal{P}_{\mathsf{CDC}}$, $\mathcal{H}_{\mathsf{IA}}$, $\mathcal{H}^n_{\mathsf{IA}}$, or $\hat{\mathcal{H}}_8$, $\mathcal{C}_8$, and $\mathcal{Q}_8$, and a* QCN $\mathcal{N}$ *of Cardinal Direction Calculus, Interval Algebra, Block Algebra, or* RCC-*8 respectively, we have that algorithm* extractPrimeQCNSC *terminates and returns a prime* QCN *of $\mathcal{N}$.*

Given a QCN $\mathcal{N} = (V, C)$ over n variables, and a subclass $\mathcal{A}$, the runtime of algorithm extractPrimeQCNSC is $O(\alpha^{|V|^2}|V|^2)$, as it iterates a number of $O(n^2)$ constraints and calls algorithm Consistency for each one of the constraints; each such call operating on a $O(\alpha^{|V|^2})$ search space. We remind the reader that α is the branching factor provided by subclass $\mathcal{A}$.

One might suggest that if a given QCN is defined over a subclass of relations for which $\diamond$-consistency alone is able to decide satisfiability, extractPrimeQCNSC can be modified by replacing the call to algorithm Consistency in line 7 with a call to algorithm WC (and appropriately modifying checks and input/output of the algorithm). This should give a faster version of the extractPrimeQCN algorithm, as we would no longer be required to perform a satisfiability check for each of the base relations not belonging to a constraint, but just one satisfiability check for the complement of that constraint (where the complement of a relation $r \in 2^{\mathsf{B}}$ is $\overline{r} = \mathsf{B} \setminus r$). However, in that case the algorithm would not be sound. The complement of a relation belonging to a subclass of relations for which $\diamond$-consistency alone is able to decide satisfiability, is not guaranteed to also belong to the same subclass; it can be a relation that results in a QCN being non-tractable in the general case.

3.6 Constraint Properties of QCNs

In this section we recall certain constraint and consistency properties that are closely related to the contributions that we will present in later chapters of the thesis.

First, we recall the definitions of the constraint properties of *patchwork* and *compactness* in the context of qualitative reasoning. The *patchwork* property was originally introduced in [Lutz and Milicic, 2007] and was shown to hold for atomic QCNs of IA and RCC-8.

Definition 15 *A qualitative temporal or spatial constraint language has* patchwork*, if for any finite satisfiable qualitative constraint networks $\mathcal{N} = (V, C)$ and $\mathcal{N}' = (V', C')$ defined in this language where for any $u, v \in V \cap V'$ we have that $C(u, v) = C'(u, v)$, the qualitative constraint network $\mathcal{N} \cup \mathcal{N}'$ is satisfiable.*

In light of patchwork, which concerns finite satisfiable qualitative constraint networks, *compactness* ensures satisfiability of an infinite sequence of finite satisfiable extensions of such a qualitative constraint network.

Definition 16 *A qualitative temporal or spatial constraint language has* compactness*, if any infinite set of constraints defined in this language is satisfiable whenever all its finite subsets are satisfiable.*

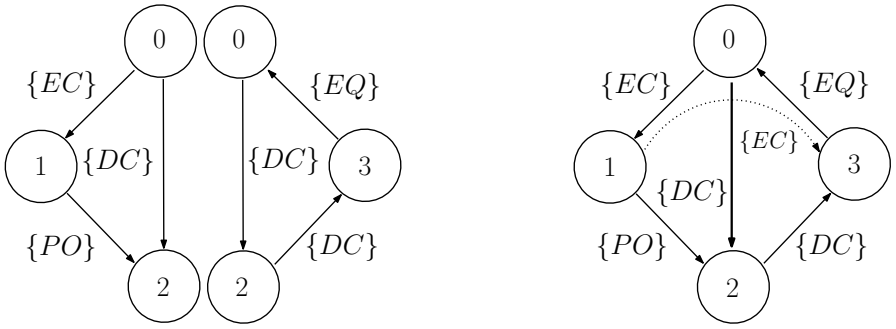


Figure 3.5: Patching two QCNs

Intuitively, patchwork ensures that unifying two satisfiable constraint networks that agree on their common part, i.e., on the constraints between their common variables, will yield a satisfiable qualitative constraint network. As an example, we can view the two QCNs of RCC-8 in Figure 3.5. (Loops corresponding to singleton relation $\{EQ\}$ and converses of constraints are not shown for simplicity.) The QCNs are atomic as they comprise singleton relations, and are also $\diamond$-consistent, therefore, by application of the patchwork property their union is satisfiable since they agree on the constraints between their common variables, namely, on $C(0,2) = \{DC\}$. Note that the universal relation that exists by definition between variables 1 and 3 in the unified QCN would result to relation $C(1,3) = \{EC\}$ if we were to calculate it, but it is not necessary to do so unless required by the specifics of a use case.

Let us now define *global consistency* and then give an example of how the former properties combined are less strict than global consistency alone.

Definition 17 *A* QCN $\mathcal{N} = (V, C)$ *is* globally consistent *if and only if, for any* $V' \subset V$*, every partial solution on* V' *can be extended to a partial solution on* $V' \cup \{v\} \subseteq V$*, for any* $v \in V \setminus V'$*.*

With respect to global consistency, we have the following result:

Theorem 8 ([Amaneddine and Condotta, 2012; Long and Li, 2015]) *Let* $\mathcal{N} = (V, C)$ *be a* QCN *defined over a distributive subclass of relations of Point Algebra, Cardinal Direction Calculus, Interval Algebra, or Block Algebra. If* $\mathcal{N}$ *is not trivially inconsistent and* $\diamond$*-consistent, then* $\mathcal{N}$ *is globally consistent.*

Patchwork is closely related to the global consistency property [Renz and Ligozat, 2005]. In particular, global consistency implies patchwork, but the opposite is not true. For example, even though RCC-8 has patchwork [Huang, 2012], it does not have global consistency [Renz and Ligozat, 2005]. For instance, let us consider the spatial configuration shown in the left of Figure 3.6. Region y is a doughnut, and region x is externally connected to it, by occupying its hole. Further, region z is externally connected to region y. For RCC-8 we know that the constraint network $\{EC(x,y), EC(y,z), EC(x,z)\}$ is satisfiable as it is $\diamond$-consistent. However, the valuation of region variables x and y is such that it is impossible to extend it with a valuation of region variable z so that $EC(x,z)$ may hold. Patchwork allows us to disregard any partial valuations and focus on the satisfiability of the network. Then, we can consider a valuation that respects the constraint network. Such a valuation is, for example, the one presented in the right of Figure 3.6. As with RCC-8, global consistency is also not available in many of the derivative constraint languages of RCC-8, such as the one that combines RCC-8 with size information about

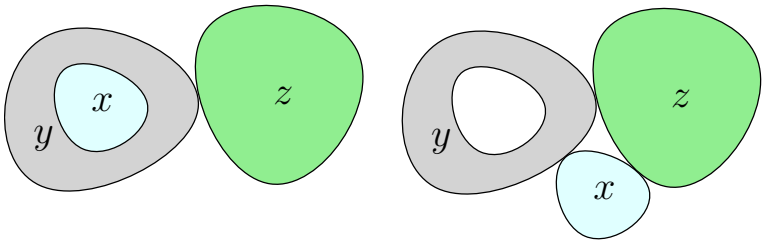


Figure 3.6: RCC-8 configurations

regions (by using Point Algebra relations to qualitative compare sizes among different regions), presented in [Gerevini and Renz, 2002].

Li et al. in [Li *et al.*, 2015a] give the following definition of the *weakly global consistency* constraint property, which aims to redefine global consistency for qualitative constraint networks in a qualitative sense:

Definition 18 *A* QCN $\mathcal{N} = (V, C)$ *is* weakly globally consistent *iff, for any* $V' \subset V$*, every partial scenario of* $\mathcal{N}$ *on* V' *can be extended to a partial scenario of* $\mathcal{N}$ *on* $V' \cup \{v\} \subseteq V$*, for any* $v \in V \setminus V'$*.*

With respect to weakly global consistency, we have the following result:

Theorem 9 ([Long and Li, 2015]) *Let* $\mathcal{N} = (V, C)$ *be a* QCN *defined over a distributive subclass of relations of a qualitative constraint language that is a relation algebra and for which every* $\diamond$*-consistent atomic* QCN *is satisfiable. If* $\mathcal{N}$ *is not trivially inconsistent and* $\diamond$*-consistent, then* $\mathcal{N}$ *is weakly globally consistent.*

Due to Propositions 1 (at page 21) and 3 (at page 34) and Theorem 9, we can obtain the following corollary:

Corollary 11 *Let* $\mathcal{N} = (V, C)$ *be a* QCN *defined over a distributive subclass of relations of Point Algebra, Cardinal Direction Calculus, Interval Algebra, Block Algebra, or* RCC*-8. If* $\mathcal{N}$ *is not trivially inconsistent and* $\diamond$*-consistent, then* $\mathcal{N}$ *is weakly globally consistent.*

Finally, Huang generalized the use of patchwork for non-atomic QCNs [Huang, 2012], providing us with the following proposition:

Proposition 16 ([Huang, 2012]) *Point Algebra, Interval Algebra, Cardinal Direction Calculus, Block Algebra, and* RCC*-8 have patchwork for not trivially inconsistent and* $\diamond$*-consistent* QCN*s defined over one of the classes* $\mathcal{P}_{\mathsf{PA}}$*,* $\mathcal{P}_{\mathsf{CDC}}$*,* $\mathcal{H}_{\mathsf{IA}}$*,* $\mathcal{H}^n_{\mathsf{IA}}$*, and* $\hat{\mathcal{H}}_8$*,* $\mathcal{C}_8$*, or* $\mathcal{Q}_8$ *respectively.*

This last proposition greatly broadens the applicability of decomposition in the context of QCNs, a practice that we will discuss in the next section, and that we adopt in most of our contributions which we will describe in detail in later chapters.

Further, with respect to compactness, we have the following result:

Proposition 17 ([Huang, 2012]) *Point Algebra, Interval Algebra, Cardinal Direction Calculus, Block Algebra, and* RCC*-8, have compactness.*

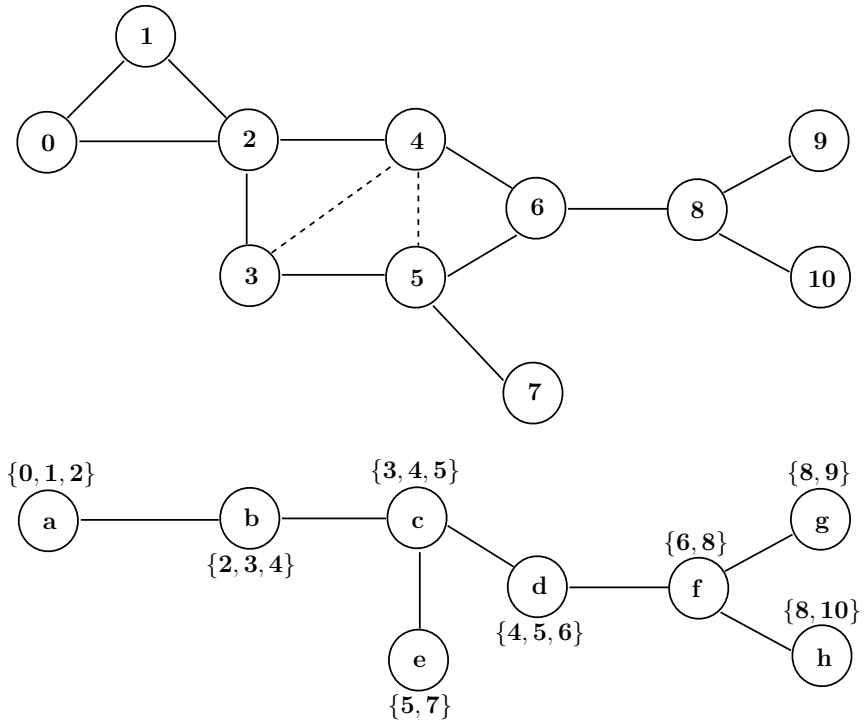


Figure 3.7: A graph (upper part) and its tree decomposition (lower part)

3.7 Decomposability of QCNs

In this section we review some theoretical and practical results regarding decomposability[6] and tractability in qualitative spatial and temporal reasoning. The related approaches perform graph decomposition to exploit some structural properties of the constraint graph of a given qualitative constraint network and apply more efficiently the algorithms for satisfiability checking that we discussed in Section 3.5. Further, they implicitly and to some extent use the patchwork property that we presented earlier in Section 3.6, in the sense that it is considered either for atomic qualitative constraint networks for which $\diamond$-consistency implies satisfiability [7], or for not trivially inconsistent qualitative constraint networks for which $\diamond$-consistency allows obtaining a solution by instantiating one variable of the qualitative constraint network at a time without backtracking (a property close to global consistency). The concept of a *tree decomposition* (originally introduced in [Halin, 1976]) will be essential in what follows. A tree decomposition is formally defined as follows.

Definition 19 *A* tree decomposition *of a graph $G = (V, E)$ is a tuple (T, X) where $T = (I, F)$ is a tree and $X = \{X_i \subseteq V \mid i \in I\}$ a collection of clusters (subsets of V) that satisfies the following conditions:*

1. *For every $v \in V$ there is at least one node $i \in I$ such that $v \in X_i$.*

[6]It is important to note that in the (older) literature the term of decomposability has been used to denote global consistency, as in [van Beek and Dechter, 1995]. We use the term of decomposability to denote the existence of any technique that involves the decomposition of the constraint graph of a given qualitative constraint network in order to enhance the efficiency and scalability of reasoning.

[7]The patchwork property is then called the *amalgamation* property. The amalgamation property becomes equivalent to the patchwork property for atomic qualitative constraint networks whenever $\diamond$-consistency implies satisfiability of these atomic qualitative constraint networks.

2. *For every* $(u, v) \in E$ *there exists a node* $i \in I$ *such that both* $u, v \in X_i$.

3. *Let* i_1, i_2, i_3 *be three nodes in* I *such that* i_2 *lies on the path between* i_1 *and* i_3 *in* T. *Then, if* $v \in V$ *belongs to both* X_{i_1} *and* X_{i_3}, v *must also belong to* X_{i_2}.

Let us view the example presented in Figure 3.7. In the upper part of the figure we can view a graph $G = (V, E)$, which can correspond to the structure of a constraint graph $G(\mathcal{N}) = (V, E)$ of a QCN $\mathcal{N}$, where $V = \{0,\ldots,10\}$ and $E = \{\{0,1\},\{1,2\},\{0,2\},\{2,3\},\{2,4\},\{3,5\},\{4,6\},\{5,6\},\{5,7\},\{6,8\},\{8,9\},\{8,10\}\}$. For the moment, we consider only the solid edges to be part of G and we disregard the dashed edges $\{3,4\}$ and $\{4,5\}$. A tree decomposition of G comprises a tree $T = (I, F)$ with the set of nodes $I = \{a,b,c,d,e,f,g,h,i\}$ and a cluster X_i for every node $i \in I$ of that tree as shown in the lower part of the figure, e.g., $X_a = \{0,1,2\}$. The first two conditions for a tuple (T, X) where $T = (I, F)$ is a tree and $X = \{X_i \subseteq V \mid i \in I\}$ a collection of clusters being a tree decomposition of a graph $G = (V, E)$, say that G is the union of the subgraphs induced by X_i, for every $i \in I$. The third condition implies that these subgraphs are organized roughly like a tree. In our previous example, it would be impossible to obtain the same tree decomposition with an additional edge $(6, 9)$, as there does not exist a node in the tree decomposition that contains both 6 and 9.

Tree decompositions have been explicitly introduced in qualitative temporal reasoning by Condotta et al. in [Condotta and D'Almeida, 2011], and implicitly in qualitative spatial and temporal reasoning by Li et al. in [Li *et al.*, 2009] and Huang et al. in [Huang *et al.*, 2013]. (The work presented in [Huang *et al.*, 2013] properly contains the work presented in [Li *et al.*, 2009], thus, we will stick to the former reference in what follows.)

In [Condotta and D'Almeida, 2011] the authors apply $\diamond$-consistency on the clusters of a tree decomposition of the constraint graph of a QCN. The graphs induced by the clusters of the tree decomposition are completed with the introduction of a new set of edges, called *fill edges*, that correspond to the universal relation for a QCN. These fill edges for the example graph of Figure 3.7 are edges $\{3,4\}$ and $\{4,5\}$. As such, the clusters of the tree decomposition are considered to be *cliques*, namely, sets of vertices such that every two vertices in a set are connected by an edge. This filling is done for two reasons: (i) by definition $\diamond$-consistency considers a complete graph to decide the satisfiability of the corresponding qualitative constraint network, and (ii) the common vertices between any two complete graphs induce a complete graph, thus, the corresponding constraint networks will completely agree on the constraints between their common variables and a patchwork-like property can be used. This property is then applied to patch together the $\diamond$-consistent QCNs of Interval Algebra defined over class $\mathcal{H}_{\mathsf{IA}}$ that correspond to the graphs induced by the clusters of the tree decomposition in a tree-like manner and construct a satisfiable qualitative constraint network. In particular, the authors in [Condotta and D'Almeida, 2011] introduce a new local consistency corresponding to the property of $\diamond$-consistency restricted to some subsets of variables of a QCN.

Definition 20 *Let* $\mathcal{N} = (V, C)$ *be* QCN *and* $\{X_0, \ldots, X_n\}$ *a family of subsets of* V. *Then,* $\mathcal{N}$ *is said to be* $\diamond_X$*-consistent if and only if for every* $X_i \in X$ *the* QCN $\mathcal{N}{\downarrow_X}$ *is* $\diamond$*-consistent.*

Then, by exploiting the notion of *pre-convexity*[8] [Ligozat, 2011] that is exhibited by the Interval Algebra relations of class $\mathcal{H}_{\mathsf{IA}}$ [Ligozat, 1996], Condotta and D'Almeida are able to prove the following result:

[8] As noted in [Ligozat, 1996], pre-convexity exemplifies an interesting patchwork-like property, which might be called generic global consistency, and is intermediate between minimality and global consistency. A solution of a not trivially inconsistent and $\diamond$-consistent QCN of Interval Algebra defined over pre-convex relations can be obtained by instantiating one variable at a time without backtracking.

Theorem 10 ([Condotta and D'Almeida, 2011]) *Let $\mathcal{N} = (V, C)$ be a* QCN *defined over class $\mathcal{H}_{\mathsf{IA}}$ of Interval Algebra, and (T, X) a tree decomposition of $G(\mathcal{N})$. We have that if $\mathcal{N}$ is not trivially inconsistent and $\overset{\diamond}{}_X$-consistent, then $\mathcal{N}$ is satisfiable.*

In [Huang *et al.*, 2013] the authors enlist a structure known as a *dtree* (decomposition tree), which, as the name suggests, is very close to a tree decomposition. Without going further into detail, a dtree is a full binary tree where the root represents a given graph and for every non-leaf node, its two children represent a partitioning of the parent graph into two subgraphs. Thus, although a dtree is not a tree decomposition, it provides a way to construct a tree decomposition out of a given graph. A dtree and a tree decomposition are therefore equivalent in the context of qualitative spatial and temporal reasoning, since omitting $\diamond$-consistency checks across children of dtree nodes (as described in [Huang *et al.*, 2013]) corresponds to omitting those checks across clusters of the tree decomposition into which the dtree is converted, as has been specifically pointed out in [Condotta and D'Almeida, 2011]. Similarly to [Condotta and D'Almeida, 2011], children of dtree nodes are treated as cliques, and the amalgamation property (patchwork for atomic QCNs) is considered to amalgamate $\diamond$-consistent atomic QCNs in a tree-like recursive manner and construct a satisfiable network. With respect to the equivalence that exists between dtrees and tree decompositions, we can infer and grant the following theorem to the authors of [Huang *et al.*, 2013]:

Theorem 11 ([Huang *et al.*, 2013]) *Let $\mathcal{N} = (V, C)$ be an atomic* QCN *defined over a qualitative constraint language that has patchwork for $\diamond$-consistent atomic* QCN*s, and (T, X) a tree decomposition of $G(\mathcal{N})$. We have that if $\mathcal{N}$ is $\overset{\diamond}{}_X$-consistent, then $\mathcal{N}$ is satisfiable.*

Consequently, by Proposition 16 (at page 54) and Theorem 11 we have the following result:

Corollary 12 *Let $\mathcal{N} = (V, C)$ be an atomic* QCN *of Point Algebra, Cardinal Direction Calculus, Interval Algebra, Block Algebra, or* RCC*-8, and (T, X) a tree decomposition of $G(\mathcal{N})$. We have that if $\mathcal{N}$ is $\overset{\diamond}{}_X$-consistent, then $\mathcal{N}$ is satisfiable.*

The aforementioned works are based on encodings of QCNs into Boolean formulas, viz., SAT encodings. However, the formulas are constructed in a way such that each solution of the formula corresponds to a not trivially inconsistent and $\diamond$-consistent QCN with relations from the class of relations at hand, and vice versa.

Before closing this section with a strong theoretical result that concerns tree decompositions and the patchwork property for atomic QCNs, let us introduce the *treewidth* of a graph. The *width* of a tree decomposition $(T, \{X_1, \ldots, X_n\})$ is $\max_{1 \leq i \leq n} |X_i| - 1$. The *treewidth* of a graph G is the minimum width possible for arbitrary tree decompositions of G. In the context of QCNs, the treewidth of a QCN $\mathcal{N}$ is simply the treewidth of its constraint graph $G(\mathcal{N})$.

Theorem 12 ([Bodirsky and Wölfl, 2011; Huang *et al.*, 2013]) *For any k, the satisfiability problem for* QCN*s of treewidth at most k that are defined on a qualitative constraint language that has patchwork for $\diamond$-consistent atomic* QCN*s can be solved in polynomial time.*

Consequently, by Proposition 16 (at page 54) and Theorem 12 we have the following result:

Corollary 13 *For any k, the satisfiability problem for* QCN*s of Point Algebra, Cardinal Direction Calculus, Interval Algebra, Block Algebra, or* RCC*-8 of treewidth at most k can be solved in polynomial time.*

A detailed algorithm for Theorem 12 that builds on the proof sketch of [Bodirsky and Wölfl, 2011] is provided in [Huang *et al.*, 2013]. In [Bodirsky and Dalmau, 2013; Bodirsky and Dalmau, 2006] the authors study CSPs over infinite domains using the concept of *ω-categoricity* [Hodges, 1997] from model theory. They use the domain D and the relations 2^{B} of a given qualitative constraint language defined on a set of base relations B to create a relational structure $\mathfrak{B}$ (called a *template*) that is ω-categorical. A relational structure $\mathfrak{B}$ is called ω-categorical if all countable models of the first-order theory of $\mathfrak{B}$ are isomorphic to each other. Interestingly, Huang shows that a qualitative constraint language over a relational structure $\mathfrak{B}$ has compactness if $\mathfrak{B}$ is ω-categorical [Huang, 2012]. For example, the relational structure $\mathfrak{B} = \{\mathbb{Q}, \{<, =, >\}\}$, which is the rational numbers $\mathbb{Q}$ with the usual comparison relations, is ω-categorical, and hence the problem of deciding whether a set of equalities and inequalities between rational variables is satisfiable has compactness. As the aforementioned problem corresponds to the well-known satisfiability checking problem of a QCN of Point Algebra, it is implied that Point Algebra has compactness. Further, in [Bodirsky and Wölfl, 2011; Bodirsky and Dalmau, 2013] Bodirsky et al. enlist the notions of treewidth and patchwork for atomic QCNs, and state that if the domain and relations of a qualitative constraint language constitute an ω-categorical relational structure, then problem instances of the qualitative constraint language of bounded treewidth are tractable and can be efficiently solved by translation into Datalog programs. So, in fact, the work presented in [Bodirsky and Wölfl, 2011] is a particular application of their more general result; a result which applies to other qualitative constraint languages as well (e.g., it applies to the qualitative constraint languages mentioned in Theorem 12). However, as the authors in [Huang *et al.*, 2013] note, it is not necessary to have compactness to prove Theorem 12 for the related qualitative constraint languages, patchwork alone is sufficient.

Another interesting notion of decomposition is employed in [Broxvall, 2002], where they decompose complex problem instances into several smaller and simpler instances that can be solved independently using the following decomposition scheme:

Definition 21 *Let $\mathcal{N} = (V, C)$ be a* QCN *of some qualitative constraint language. A* partitioning *of $\mathcal{N}$ is a* QCN *$\mathcal{N}_p = (\{\mathcal{N}_1, \ldots, \mathcal{N}_n\}, C_p)$ with variables representing* QCNs *$\mathcal{N}_1 = (V_1, C_1)$, …, $\mathcal{N}_n = (V_n, C_n)$ such that:*

- *$V_i \subseteq V$ for every $i \in \{1, \ldots, n\}$.*
- *$V = \bigcup_{i=1}^{n} V_i$ and $V_i, \ldots, V_n$ are disjoint.*
- *For every $u, v \in V$ we have that $C(u, v) \in C_i$ for some $i \in \{1, \ldots, n\}$, or $u \in V_i$, $v \in V_j$, and $C'(\mathcal{N}_i, \mathcal{N}_j) \in C_p$ with $C'(\mathcal{N}_i, \mathcal{N}_j) \subseteq C(u, v)$ and $i, j \in \{1, \ldots, n\}$ with $i \neq j$.*
- *$\mathcal{N}_p$ is satisfiable.*

Given a QCN $\mathcal{N} = (V, C)$, a partitioning $\mathcal{N}_p = (\{\mathcal{N}_1, \ldots, \mathcal{N}_n\}, C_p)$ of $\mathcal{N}$ partitions the set of variables V of $\mathcal{N}$ into exactly n sets, each of which is assigned to a distinct node of $\mathcal{N}_p$. Whenever two variables of $\mathcal{N}$ reside in different nodes of $\mathcal{N}_p$, the relation holding between the two nodes of $\mathcal{N}_p$ is a subset of the relation holding between the two variables of $\mathcal{N}$. In a sense, each node of $\mathcal{N}_p$ forms a QCN that has a subset of the set of variables of $\mathcal{N}$ as its set of variables, and original relations between variables of $\mathcal{N}$ as its set of relations. It was proven in [Broxvall, 2002] that if the set of relations of $\mathcal{N}_p$ is sufficiently restricted, then the original network $\mathcal{N}$ is satisfiable exactly when all the QCNs $\mathcal{N}_1$, …, $\mathcal{N}_n$ that correspond to the nodes of $\mathcal{N}_p$ are individually satisfiable. The relations of a given qualitative constraint language that satisfy this condition are called *partitioning* relations. For example, the set of relations $\{\{DC\}, \{PO\}, \{DC, PO\}\}$ for

RCC-8 is a set of partitioning relations. Although the aforementioned technique is very elegant in its conception, it was noted in [Broxvall, 2002] that useful instances of this kind of decomposition can be difficult to identify, especially when the size of the set of partitioning relations is small (as is the case with Interval Algebra and RCC-8), thus, deeming the technique impractical for efficient reasoning with qualitative constraint languages.

The utility of taking advantage of the structure of qualitative constraint networks has already been addressed in the context of heuristics for the $\diamond$-consistency enforcing algorithms in the works of [van Beek and Manchak, 1996; Renz and Nebel, 2001; Renz, 2002b] and discussed in Section3.5. In particular, we remind the reader that the proposed heuristics target the *denser* parts of the underlying constraint graph of a given qualitative constraint network, i.e., the qualitative relations that consist of few base relations, as an effort to propagate constraints more efficiently and also possibly resolve any local inconsistencies faster. Pruning unfeasible base relations off a qualitative relation that already comprises very few base relations can almost immediately unveil an inconsistency. However, these heuristics always consider a complete underlying constraint graph of a given network. As such, they fail to completely isolate parts of the underlying constraint graph of a given qualitative constraint network that are irrelevant to the process of satisfiability checking; such parts being universal relations that do not belong to the clusters of a tree decomposition corresponding to the constraint graph at hand. We demonstrated earlier that it is possible to omit satisfiability checks across clusters of a tree decomposition corresponding to the constraint graph of a given qualitative constraint network.

3.7.1 Decomposability in the CSP framework

A qualitative constraint network is most efficiently modelled as an infinite-domain variant of a constraint satisfaction problem through the use of a relation algebra [Ladkin and Maddux, 1994], which is also the approach we follow in our thesis. However, a qualitative constraint network can also be encoded as a finite constraint satisfaction problem instance [Renz and Nebel, 2001; Brand, 2004; Condotta *et al.*, 2006b]. In particular, given a qualitative constraint network (V, C) where $|V| = n$, we can obtain a constraint satisfaction problem instance as follows. Let X denote the set of variables containing a variable x_{ij} for each pair of variables $v_i, v_j \in V$ with $1 \leq i < j \leq n$. Then, our instance has the form $(X, \mathsf{B}, DCon \cup TCon)$, where $DCon$ is the set of domain constraints $\{(x_{ij}, C_{ij}) \mid 1 \leq i < j \leq n\}$ and $TCon$ the set of ternary constraints $\{((x_{ij}, x_{ik}, x_{kj}), R_\diamond) \mid 1 \leq i < j < k \leq n\}$ with $R_\diamond = \{(b, b', b'') \in \mathsf{B}^3 \mid b \in b' \diamond b''\}$. Namely, $DCon$ restricts the values of a variable x_{ij} to the base relations of the corresponding qualitative constraint C_{ij} and $TCon$ encodes all the consistent paths of length 2 in the network. The resulting finite network has $\frac{n(n-1)}{2}$ variables and $\binom{n}{3}$ ternary constraints. A solution of this finite instance corresponds to an atomic $\diamond$-consistent refinement of a given qualitative constraint network, and vice versa [Condotta *et al.*, 2006b]. The main disadvantage of this approach is that we are not able to make use of maximal tractable subclasses of relations, which can seriously impact the performance of satisfiability checking for calculi that heavily rely upon those subclasses, such as RCC-8 and IA. However, for large-sized qualitative calculi (viz., comprising hundreds of base relations) for which no tractable subclasses are known, a finite constraint satisfaction problem encoding can provide a considerable performance gain [Westphal and Wölfl, 2009].

In light of the strong relation that exists between qualitative and "traditional" constraint programming, it is worth mentioning some works in the latter paradigm that exploit the structure of constraint graphs in a similar manner to what we presented in this section. The interested reader may review the cited works and obtain a deeper understanding on the analogy that exists between structural characteristics of qualitative constraint networks and finite constraint

satisfaction problem instances. What is more important, the cited works may drive future research by enabling the reader to identify theoretical properties in the context of qualitative spatial and temporal reasoning, that can be used to adopt certain techniques for exploiting the structure of constraint graphs that exist in constraint programming.

In [Walsh, 2001], Walsh measures the impact that the structure of a constraint graph can have on the performance of solving a typical constraint satisfaction problem, viz., the *graph coloring problem*, which is the problem of coloring the vertices of a graph in such a way that no two adjacent vertices share the same color.

In [Baget and Tognetti, 2001], Baget et al. proposes a backtracking algorithm for solving constraint satisfaction problem instances that exploits the biconnected component subgraphs of a given constraint graph to reduce search space, permanently removing values and compiling partial solutions during exploitation.

In [Dechter and Pearl, 1989], Dechter et al. propose a constraint graph restructuring technique, based on tree decompositions, that guarantees that a large variety of queries could be answered swiftly either by sequential backtrack-free procedures, or by distributed constraint propagation methods.

Based on the work of Dechter et al. [Dechter and Pearl, 1989], Jégou et al. in [Jégou and Terrioux, 2003] propose a framework for solving constraint satisfaction problem instances that relies both on backtracking techniques and on the notion of tree decomposition of the constraint graphs. Notably, this mixed approach has been implemented and used successfully for practical constraint satisfaction problem solving [Jégou and Terrioux, 2003].

Jégou et al. in [Jégou *et al.*, 2005] study several methods for computing a rough optimal tree decomposition and assess their relevance for solving constraint satisfaction problem instances, and the same authors went on to propose dynamic heuristics for efficient backtrack search on tree decompositions of constraint graphs in [Jégou *et al.*, 2006; Jégou *et al.*, 2007].

Recently, in [Jégou and Terrioux, 2014a; Jégou and Terrioux, 2014b] Jégou et al. introduced and exploited a new graph parameter called *bag-connected tree-width* which considers tree decompositions for which each cluster induces a connected graph. It is experimentally shown in [Jégou and Terrioux, 2014b] that such bag-connected tree decompositions significantly improve the solving of constraint satisfaction problem instances by decomposition methods.

Finally, a nice uniform presentation of the major structural constraint satisfaction problem decomposition methods discussed here is given by Gottlob et al. in [Gottlob *et al.*, 2000].

3.8 Conclusion

In this chapter, we formally introduced the notion of a qualitative constraint network (QCN), drew the connection between the relational operations we presented in the previous chapter and certain useful local consistency conditions for characterizing QCNs, presented the fundamental reasoning problems that are associated with a QCN, viz., the satisfiability problem, the minimal labeling problem, and the redundancy problem, and overviewed the state of the art algorithms for dealing with those reasoning problems. Further, we explained some constraint properties of QCNs and, finally, we presented certain particular decomposability aspects of QCNs considered in the literature.

The discussion that took place in this chapter, will be relevant to the contributions that we will present in Chapter 5. In particular, the theoretical results that we will obtain in that chapter will be directly comparable to the theoretical results of this chapter, and the experimental evaluations of our novel techniques will be made against state of the art methods so that a clear

picture can be drawn, showing the technical advance in the field of qualitative constraint-based spatial and temporal reasoning.

Chapter 4

Combining Space & Time into Qualitative Spatio-Temporal Frameworks

4.1 Introduction

Time and space are fundamental cognitive concepts that have been the focus of study in many scientific disciplines, including Artificial Intelligence and, in particular, Knowledge Representation, as we have witnessed up to this point. Specifically, Knowledge Representation has been quite successful in dealing with the concepts of time and space, and has developed formalisms that range from temporal and spatial databases [Story and Worboys, 1995], to quantitative models developed in computational geometry [Preparata and Shamos, 1985] and qualitative constraint languages and logical theories developed in qualitative reasoning [Hazarika, 2012; Wolter and Zakharyaschev, 2003]. Regarding qualitative reasoning in particular, we have already reviewed in Chapter 2 some qualitative constraint-based formalisms for reasoning about time and space that consider a set of base relations for representing particular qualitative configurations between spatial or temporal entities.

In this chapter, we review the state of the art frameworks that combine space and time in an interrelated manner. With such qualitative spatio-temporal frameworks, we can represent for example the fact that a given region was contained in another region at one point in time and externally connected to that region at a next point of time, or even the fact that a point will always move towards a particular direction over time. Towards constraint-based qualitative spatio-temporal reasoning, most of the work has relied on formalisms based on the propositional temporal logic (PTL), also known as linear temporal logic, and some qualitative spatial constraint language, like RCC-8 [Wolter and Zakharyaschev, 2003; Wolter and Zakharyaschev, 2000b]. PTL [Huth and Ryan, 2004] is the well known temporal logic comprising operators $\mathcal{U}$ (until), $\bigcirc$ (next point in time), $\Box$ (always), and $\Diamond$ (eventually) over various flows in time, such as $\langle \mathbb{N}, < \rangle$. Other spatio-temporal reasoning frameworks consider temporal sequences of spatial QCNs. These sequences allow one to describe a spatial configuration that evolves and changes over time. Indeed, solving the spatial QCNs in such a given sequence will in turn yield a sequence of scenarios constituting a timeline, upon which the different states of a qualitative spatial configuration that evolves over time can be viewed. We also explore a qualitative constraint-based spatio-temporal formalism that results from combining the temporal language of Interval Algebra with the spatial one of RCC-8, and we close the chapter with a discussion on the notions of

periodicity and recurring patterns that can appear in temporalized QCNs.

4.2 Linear Point-based Time Spatio-Temporal Logics

In general, a spatial QCN, as described in Section 3.2, constitutes a static spatial configuration in some domain D, over a set of spatial variables V. To be able to describe a spatial configuration that changes over time, we can combine the *propositional temporal logic* PTL [Huth and Ryan, 2004; Pnueli, 1977] with a qualitative spatial constraint language in a unique formalism [Kontchakov *et al.*, 2007; Wolter and Zakharyaschev, 2003; Balbiani and Condotta, 2002]. The domain D of a QCN will always remain the same, but the spatial variables in it may spatially change with the passing time (e.g., in shape, size, or orientation). We can interpret formulas of such a spatio-temporal formalism using a spatio-temporal structure defined as follows.

Definition 22 *Given a qualitative spatial constraint language based on a finite set of base relation* B *defined over a domain* D *and a set of spatial variables* V*, a* ST-structure *is a tuple* $\mathcal{M}_{\mathsf{ST}} = (V, \mathsf{D}, \mathbb{N}, \alpha)$*, where* α *is a mapping that associates an element of* D *with each spatial variable of* V *at a point of time* $i \in \mathbb{N}$*, i.e.,* $\alpha : \mathbb{N} \to (V \to \mathsf{D})$*. Thus,* $\alpha(i)$ *denotes the set of elements of* D *that are associated with the set of spatial variables* V *at point of time* i*. By extending notation,* $\alpha(v, i)$*, where* $v \in V$*, denotes the element of* D *that is associated with spatial variable* v *at point of time* i*.*

For the case of RCC-8 for example, if $\mathcal{T}$ is some topological space [Munkres, 2000], let $\mathcal{R}(\mathcal{T})$ denote the set of all non-empty regular closed subsets in $\mathcal{T}$. Then, the domain D of RCC-8 is the set $\mathcal{R}(\mathcal{T})$, which can be infinite. As such, α would be a mapping associating an element of $\mathcal{R}(\mathcal{T})$ with each spatial region variable at a point of time $i \in \mathbb{N}$. In a sense, a ST-structure is a pure two-dimensional structure of space and time. We can fix a moment in time, and then move in the spatial dimension and represent the state of the spatial configuration at that moment in time. Likewise, we can focus on a single spatial variable in the spatial configuration, and then move in the temporal dimension and witness the evolution of that spatial variable through time. We can construct several linear point-based time spatio-temporal logics through the application of the temporal operators of PTL in our qualitative spatial constraint language of choice. The most important and dominant of such logics follow.

The $\mathcal{L}_0$ logic. The set of atomic propositions AP in the case of standalone PTL is replaced by the set of base relations B of the qualitative spatial constraint language considered. We will refer to such a spatio-temporal formula over B as a $\mathcal{L}_0$ formula. Thus, given a set of spatial variables V, the set of $\mathcal{L}_0$ formulas over B is inductively defined as follows. if $P \in \mathsf{B}$, and $u, v \in V$, then $P(u, v)$ is a $\mathcal{L}_0$ formula, and if ψ and ϕ are $\mathcal{L}_0$ formulas then $\neg\phi$, $\phi \vee \psi$, $\bigcirc\phi$, $\Box\phi$, $\Diamond\phi$, and $\phi \,\mathcal{U}\, \psi$ are $\mathcal{L}_0$ formulas. Formulas of the form $\Diamond\phi$ and $\Box\phi$ are abbreviations for $\top \,\mathcal{U}\, \phi$ and $\neg(\top \,\mathcal{U}\, \neg\phi)$ respectively.

A simple example of a $\mathcal{L}_0$ formula is the following:

$$\Box NTPP(Nucleus, Cell)$$

The aforementioned $\mathcal{L}_0$ formula states that a nucleus will always be located inside its cell. In fact, all $\mathcal{L}_0$ formulas are rather simple, in the sense that the expressive power of the $\mathcal{L}_0$ logic is quite limited; one cannot correlate the states of a spatial variable between different points of time. As a matter of fact, $\mathcal{L}_0$ formulas can be reduced to a sequence of spatial configurations that

are completely independent of one another, but we will see more about this in a later chapter where we will make our contributions. However, the $\mathcal{L}_\mathbf{0}$ logic is expressive enough for capturing some aspects of spatial change. As illustration, we can have the following statements using the $\mathcal{L}_\mathbf{0}$ logic:

$$\Box(DC(Virus, Cell) \rightarrow \Box DC(Virus, Cell) \vee \bigcirc\Diamond EC(Virus, Cell))$$

$$\Box(EC(Virus, Cell) \rightarrow \bigcirc\Diamond PO(Virus, Cell))$$

$$\Box(PO(Virus, Cell) \rightarrow \bigcirc\Diamond TPP(Virus, Cell))$$

$$\Box(TPP(Virus, Cell) \rightarrow \bigcirc\Diamond NTPP(Virus, Cell))$$

The aforementioned $\mathcal{L}_\mathbf{0}$ formulas taken together can describe the process of a virus latching onto a host cell and then getting inside of it. Indeed, the virus and the host cell are disconnected at some moment in time, and they will either remain disconnected for ever (the virus itself will die) or the virus will begin the process of infecting the cell by first latching onto it and then getting inside of it (at which point it will survive and multiply).

Definition 23 *Given a $\mathcal{L}_\mathbf{0}$ formula ϕ over B, we write $\langle \mathcal{M}_{\mathsf{ST}}, i\rangle \models \phi$ for the fact that $\mathcal{M}_{\mathsf{ST}}$ satisfies ϕ at point of time i, with $i \in \mathbb{N}$ (or formula ϕ is true in $\mathcal{M}_{\mathsf{ST}}$ at point of time i). The semantics is then defined as follows.*

- $\langle \mathcal{M}_{\mathsf{ST}}, i\rangle \models P(v, v')$ *iff* $P(\alpha(v, i), \alpha(v', i))$, *with* $P \in \mathsf{B}$
- $\langle \mathcal{M}_{\mathsf{ST}}, i\rangle \models \neg\phi$ *iff* $\langle \mathcal{M}_{\mathsf{ST}}, i\rangle \not\models \phi$
- $\langle \mathcal{M}_{\mathsf{ST}}, i\rangle \models \phi \vee \psi$ *iff* $\langle \mathcal{M}_{\mathsf{ST}}, i\rangle \models \phi$ *or* $\langle \mathcal{M}_{\mathsf{ST}}, i\rangle \models \psi$
- $\langle \mathcal{M}_{\mathsf{ST}}, i\rangle \models \phi \mathcal{U} \psi$ *if there exists a* $k \in \mathbb{N}$ *such that* $i \leq k$, $\langle \mathcal{M}_{\mathsf{ST}}, k\rangle \models \psi$, *and for all* $j \in \mathbb{N}$, *if* $i \leq j$ *and* $j < k$ *then* $\langle \mathcal{M}_{\mathsf{ST}}, j\rangle \models \phi$
- $\langle \mathcal{M}_{\mathsf{ST}}, i\rangle \models \bigcirc\phi$ *iff* $\langle \mathcal{M}_{\mathsf{ST}}, i+1\rangle \models \phi$

A ST-structure $\mathcal{M}_{\mathsf{ST}} = (V, \mathsf{D}, \mathbb{N}, \alpha)$, for which $\langle \mathcal{M}_{\mathsf{ST}}, 0\rangle \models \phi$, is a model for ϕ. It follows that a $\mathcal{L}_\mathbf{0}$ formula ϕ is satisfiable if there exists a model for it. The number of occurrences of symbols in a $\mathcal{L}_\mathbf{0}$ formula ϕ is denoted by $length(\phi)$ (as with every syntactical object). We then have the following result:

Theorem 13 ([Gabelaia *et al.*, 2005]) *Checking the satisfiability of a $\mathcal{L}_\mathbf{0}$ formula ϕ in a* ST*-structure is* PSPACE*-complete (in $length(\phi)$) if the satisfiability of* QCN*s defined in the considered qualitative constraint language can be decided in* PSPACE.

Due to Theorem 13 and Proposition 3 (at page 34), we can obtain the following corollary:

Corollary 14 *Checking the satisfiability of a $\mathcal{L}_\mathbf{0}$ formula in a* ST*-structure is* PSPACE*-complete for the qualitative constraint languages of Point Algebra, Cardinal Direction Calculus, Interval Algebra, Block Algebra, and* RCC*-8.*

The $\mathcal{L}_1$ logic. To increase the expressiveness of the $\mathcal{L}_0$ logic we can allow the application of operator $\bigcirc$ to spatial variables. Given a spatial variable X, $\bigcirc X$ represents the state of region X at the next point of time. For example, we can have the following statement in RCC-8:

$$\Diamond NTPP(Cell, \bigcirc Cell)$$

The aforementioned formula states that at some point of time a given cell will decrease in size, or, equivalenty, that we will eventually have two consecutive points of time such that the size of the cell in the first point of time will be larger than the size of the cell in the second point of time. The formula essentially describes the process of aging for any given human cell. The shrinking of a cell is a normal aging change that occurs in any human tissue. In a similar manner, we can specify with the following statement that a given spatial object, like an implant for instance, will never change its size:

$$\Box EQ(Implant, \bigcirc Implant)$$

As an implant is a rigid technical device, its size does not change through time; the size of an implant at any given point of time, will equal the size of the implant at the next point of time. Further, we can use the $\bigcirc$ operator to specify that a spatial object has two distinct states, allowing one for example to represent the cardiac cycle with the following simple statement:

$$\Box EQ(Heart, \bigcirc \bigcirc Heart)$$

The cardiac cycle consists of two distinct states, the *systole* (contraction), which is the part of the cardiac cycle when the ventricles contract, and the *diastole* (enlargement), which is the part of the cardiac cycle when the heart refills with blood following systole. As such, the size of the heart right before systole will equal the size of the heart right after a cardiac cycle has been completed (assuming that systole and diastole occur in two dinstinct consecutive points of time). Finally, we can use auxiliary spatial variables to compare distinct spatial objects between different time intervals. For example, we can have the following statement:

$$\Box EQ(X, \bigcirc X) \wedge EQ(X, Neoplasm) \wedge (NTPP(Tissue, X) \vee TPP(Tissue, X))$$

The aforementioned formula states that a tissue in its present state, i.e., an ensemble of similar cells as it appears in the present state, will be part of a neoplasm at some future point of time.

We call this enriched logic the $\mathcal{L}_1$ logic. The semantics of the $\mathcal{L}_1$ logic is the same with that of the $\mathcal{L}_0$ logic, provided that we appropriately handle the application of operator $\bigcirc$ to spatial variables. In particular, given a ST-structure $\mathcal{M}_{\mathsf{ST}} = (V, \mathsf{D}, \mathbb{N}, \alpha)$, a spatial variable $v \in V$, and some positive integer m, we treat an assignment $\alpha(\bigcirc^m v, i)$, where $i \in \mathbb{N}$, as follows.

$$\alpha(\bigcirc^m v, i) = \alpha(v, i + m) \text{ where } i + m \text{ is the } m^{\text{th}} \text{ successor of } i \text{ in } \langle \mathbb{N}, < \rangle$$

Like with the $\mathcal{L}_0$ logic, a $\mathcal{L}_1$ formula ϕ is satisfiable if there exists a model for it, and the number of occurrences of symbols in it is denoted by $length(\phi)$. In what follows in the thesis with respect to $\mathcal{L}_1$, we always assume that the satisfiability of atomic QCNs defined in the considered qualitative constraint language can be decided in PSPACE. We then have the following result:

Theorem 14 ([Balbiani and Condotta, 2002]) *Checking the satisfiability of a $\mathcal{L}_1$ formula ϕ in a ST-structure is PSPACE-complete if satisfiable atomic QCNs defined in the considered qualitative constraint language are globally consistent.*

Due to Theorem 14 and Propositions 3 (at page 34) and 8 (at page 53), we can obtain the following corollary:

Corollary 15 *Checking the satisfiability of a* $\mathcal{L}_1$ *formula* ϕ *in a* ST*-structure is* PSPACE*-complete for the qualitative constraint languages of Point Algebra, Cardinal Direction Calculus, Interval Algebra, and Block Algebra.*

Interestingly, Gabelaia et al. in [Gabelaia *et al.*, 2003; Gabelaia *et al.*, 2005] restricted the class of arbitrary topological spaces that serve as a domain basis for RCC-8 to the class of topological spaces generated by maximal *saws* with countably many *forks* of each type, and identified a *completion property* (as they call it), which basically translates to global consistency for atomic QCNs of RCC-8 that are defined over that particular domain interpretation.[9] As such, we also have the following result:

Theorem 15 ([Gabelaia *et al.*, 2005]) *Checking the satisfiability of a* $\mathcal{L}_1$ *formula* ϕ *in a* ST*-structure is* PSPACE*-complete for* RCC*-8 with respect to the class of topological spaces generated by maximal saws with countably many forks of each type.*

This result regarding RCC-8 was a substantial contribution, even under the aforementioned domain restrictions, as until that point of time the complexity of that particular logic concerning RCC-8 was known to be in EXPSPACE [Wolter and Zakharyaschev, 2000b].

The $\mathcal{L}_2$ logic. Let us revisit the previous example about the tissue and the neoplasm, in particular, let us review the following statement:

$$\Box EQ(X, \bigcirc X) \wedge EQ(X, Neoplasm) \wedge (NTPP(Tissue, X) \vee TPP(Tissue, X))$$

As we already explained earlier, the aforementioned formula states that a tissue in its present state, i.e., an entire ensemble of similar cells as it appears in the present state, will be part of a neoplasm at *some* future point of time. Let us assume that we want to represent the knowledge that each of the cells that constitutes the tissue will be part of the neoplasm at some possibly distinct future point of time with respect to the other cells. In other words, as the neoplasm evolves, it will engulf certain cells of the tissue at some points of time, while at other points of time some rogue cells of the tissue that are already part of the neoplasm may stop being part of the neoplasm because they will either die or return to a normal state. In the end, all of the cells of the tissue will have been a part of the neoplasm, but each of the cells at its own possibly distinct point of time with respect to the other cells. It is impossible to represent that knowledge using the $\mathcal{L}_1$ logic. However, we can further enrich the logic by allowing the application of operators $\Diamond$ and $\Box$ to spatial variables. Let us examine the following statement:

$$NTPP(Tissue, \Diamond Neoplasm) \vee TPP(Tissue, \Diamond Neoplasm)$$

We have that $\Diamond Neoplasm$ comprises all of the cells that will be part of the neoplasm at some point of time in the future, i.e., $\Diamond Neoplasm$ represents the union of the cells that are part of the neoplasm as it appears in some future point of time. In a similar manner, let us consider the following sentence:

$$NTPP(Tissue, \Box Neoplasm) \vee TPP(Tissue, \Box Neoplasm)$$

[9] A *saw* is a disjoint union of *forks*, with a fork being a three-point frame $\langle \{b, r, l\}, R \rangle$ such that bRr and bRl.

In this case, we have that $\Box Neoplasm$ comprises all of the common cells that will be part of the neoplasm at all points of time in the future, i.e., $\Box Neoplasm$ represents the intersection of the cells that are part of the neoplasm as it appears in some future point of time.

We call this further enriched logic the $\mathcal{L}_{\mathbf{2}}$ logic. The semantics of the $\mathcal{L}_{\mathbf{2}}$ logic includes the semantics of the $\mathcal{L}_{\mathbf{1}}$ logic, and considers two additions to appropriately handle the application of operators $\Diamond$ and $\Box$ to spatial variables. In particular, given a ST-structure $\mathcal{M}_{\mathsf{ST}} = (V, \mathsf{D}, \mathbb{N}, \alpha)$, a spatial variable $v \in V$, and some positive integer m, we treat the assignments $\alpha(\Box v, i)$ and $\alpha(\Diamond v, i)$, where $i \in \mathbb{N}$, as follows.

$$\alpha(\Box v, i) = c(\bigcap_{j \geq i} \alpha(v, j))$$

$$\alpha(\Diamond v, i) = c(\bigcup_{j \geq i} \alpha(v, j))$$

In the aforementioned definitions regarding α we have used $c(\cdot)$ to obtain the topological closure of an intersection or a union of closed sets. Indeed, we have already established in Section 2.3, and also mentioned earlier in this section, that the domain D of RCC-8 is the set of all non-empty regular closed subsets in some topological space $\mathcal{T}$; a domain of closed sets is also the case for the rest of the qualitative constraint languages that we consider in this thesis, viz., the qualitative constraint languages of Point Algebra, Cardinal Direction Calculus, Interval Algebra, and Block Algebra. In the case of Interval Algebra for example, a domain value is a tuple of the form (x^-, x^+), where x^- and x^+ both belong to $\mathbb{Q}$ and correspond to the starting and the ending point respectively of a temporal interval. That temporal interval contains x^- and x^+ as its limit points, i.e., the set of rational numbers defined by tuple (x^-, x^+) is closed. The reason that we have to use the closure operator $c(\cdot)$ to obtain the topological closure of an intersection or a union of closed sets, is because that intersection or union may be one of infinitely many closed sets and in that case the respective result may not be closed [Munkres, 2000]. However, the use of the $c(\cdot)$ operator does not save us from several semantic complications [Wolter and Zakharyaschev, 2003]. To simplify reasoning, we can impose a restriction to the domain at hand so that it may yield only a finite number of realizations for any given spatial variable. In this way, we will only have to deal with a finite number of intersections or unions of closed sets. To this end, we can adopt the *Finite State Assumption* that is defined as follows.

Definition 24 *Given a* ST*-structure* $\mathcal{M}_{\mathsf{ST}} = (V, \mathsf{D}, \mathbb{N}, \alpha)$*, we say that* $\mathcal{M}_{\mathsf{ST}}$ *satisfies the* Finite State Assumption*, or that* $\mathcal{M}_{\mathsf{ST}}$ *is an FSA-structure, iff for every spatial variable* $v \in V$ *there are finitely many distinct values* $\mathsf{d}_1, \mathsf{d}_2, \ldots, \mathsf{d}_m \in \mathsf{D}$*, with* m *being a positive integer, such that* $\{\alpha(v, i) \mid i \in \mathbb{N}\} = \{\mathsf{d}_1, \mathsf{d}_2, \ldots, \mathsf{d}_m\}$.

Like with the $\mathcal{L}_{\mathbf{1}}$ logic, a $\mathcal{L}_{\mathbf{2}}$ formula ϕ is satisfiable if there exists a model for it, and the number of occurrences of symbols in it is denoted by $length(\phi)$. We then have the following result:

Theorem 16 ([Wolter and Zakharyaschev, 2000b]) *Checking the satisfiability of a* $\mathcal{L}_{\mathbf{2}}$ *formula* ϕ *of* RCC*-8 in a FSA-structure is in* EXPSPACE.

Up to this point, we have presented an hierarchy of spatio-temporal logics, namely, the $\mathcal{L}_{\mathbf{0}}$, $\mathcal{L}_{\mathbf{1}}$, and $\mathcal{L}_{\mathbf{2}}$ logics, that is strictly based on a set of spatial variables where each variable respresents a unary, non-complex, term that maps to some domain value. However, we can introduce *Boolean variable terms*, which are combinations of spatial variables using the Boolean operators $\vee$, $\wedge$,

and $\neg$, and obtain, as an example, BRCC-8 [Wolter and Zakharyaschev, 2000a], which is the extension of RCC-8 that allows the use of Boolean variable terms as arguments of the RCC-8 predicates. For instance, we can have the following simple statement in BRCC-8:

$$EQ(Europe, Greece \vee Germany \vee France \vee \ldots)$$

The aforementioned BRCC-8 formula states that Europe is the *union* of all the European countries as we know them today. Note that this simple fact alone would be impossible to express in RCC-8, given that Europe and all of its separate countries (at least as we know them today) comprised our set of spatial variables. Regarding BRCC-8, it was shown to increase the expressive power of RCC-8, while retaining the same computational behavior as RCC-8 in arbitrary topological spaces [Wolter and Zakharyaschev, 2000a]. However, the combination of BRCC-8 with certain temporal logics, such as PTL, may have a different computational behavior than the combination of RCC-8 with those same logics. In the case of spatio-temporal logics based on PTL for example, most of the resulting logics that consider BRCC-8 have a higher computational complexity than the respective ones that consider RCC-8 (a generalization of which to spatial calculi other than RCC-8 we presented here). Further, things become more complicated when we consider only *connected* topological spaces to serve as a domain basis for BRCC-8. For instance, in that case, the satisfiability problem for standalone BRCC-8 becomes PSPACE-complete.

Finally, in the literature there has also been an effort to abstract and generalize the aforementioned hierarchy of spatio-temporal logics into the, so called, propositional spatio-temporal logic (PST) [Bennett *et al.*, 2002]. PST is the Cartesian product of the temporal logic PTL and the modal logic $\mathsf{S4_u}$, which is the Lewis modal system S4 [Lewis and Langford, 1932] augmented with the *universal modality* $\boxdot$ [Goranko and Passy, 1992]. The motivation behind developing the $\mathsf{S4_u}$ modal logic lay in the fact that, in the past, Tsao-Chen [Tsao-Chen, 1938] and later McKinsey and Tarski [McKinsey and Tarski, 1948] interpreted the necessity operator of the Lewis modal system S4 as the interior operator of topological spaces and proved that S4 is sound and complete with respect to this interpretation (cf. [Chagrov and Zakharyaschev, 1997]). As the satisfiability problem for S4 was then proved to be PSPACE-complete by Ladner [Ladner, 1977], the works of Tsao-Chen, and McKinsey and Tarksi, implied that there could be a computationally viable encoding for describing a general class of topological relations. Indeed, the augmented $\mathsf{S4_u}$ encoding provided a decision procedure for a spatial language that could express a large class of topological relations including all those in RCC-8 and BRCC-8 [Bennett, 1996; Bennett, 1998; Wolter and Zakharyaschev, 2003]. However, the combination of PTL and the modal logic $\mathsf{S4_u}$, namely, the PST logic, resulted in it being "too expressive". In particular, the satisfiability problem for PST formulas over the discrete flow of time $\mathbb{N}$ was shown to be undecidable in [Gabelaia *et al.*, 2003]. The same undecidability result also holds for simpler fragments of PST, such as the one where only the temporal operator $\Box$ is used and the universal modality $\boxdot$ is not used at all [Gabelaia *et al.*, 2003]. Another undecidable spatio-temporal theory is that of Muller proposed in [Muller, 1998; Muller, 2002], which is basically a first-order axiomatization of spatio-temporal entities based on the Region Connection Calculus. Delving a little deeper into modal logics, there have been multimodal logic approaches to qualitative spatio-temporal reasoning studied in the works of Burrieza et al. [Burrieza and Ojeda-Aciego, 2005; Burrieza *et al.*, 2011; Burrieza *et al.*, 2009], Muñoz-Velasco et al. [Muñoz-Velasco *et al.*, 2014], and Golinska-Pilarek et al. [Golinska-Pilarek and Muñoz-Velasco, 2012]. These approaches are based on extentions of modal logics, such as dynamic logics and variations thereof [Harel *et al.*, 2000], and are concerned with formalizing several spatio-temporal notions such as closeness, distance, and velocity.

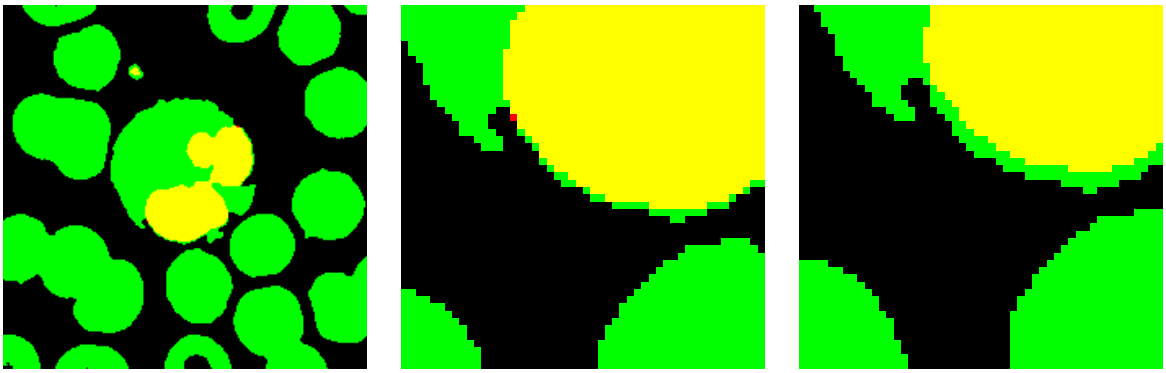


Figure 4.1: Left: segmented cell bodies (green), lobulated cell nuclei (yellow and red) and background (black), Middle: segmented cell nucleus extending outside border of host cell (red pixels), Right: the result of applying a morphological erosion operator; here the original *partially overlaps* relation changes to *proper part*

4.3 Spatio-Temporal Change based on Transition Constraints

In this section, we focus on a particular spatio-temporal reasoning framework that considers temporal sequences of spatial QCNs. These sequences allow us to describe a spatial configuration that evolves and changes over time. Indeed, solving the spatial QCNs in such a given sequence will in turn yield a sequence of scenarios constituting a timeline, upon which the different states of a qualitative spatial configuration that evolves over time can be viewed. This modelling can have many practical and diverse applications, from identifying optimal routes in mobile robot navigation, to modelling changes of topology in biological processes and computing sequences of segmentation steps used in image processing algorithms. All these application examples can be modelled as a sequence of successive spatial QCNs where we look for ways to solve the QCNs in such a manner that an assumed set of a priori constraints are satisfied by the obtained sequence of scenarios. For example, in the case of a phagocyte ingesting food, one constraint may be that the food has to be part of a food vacuole in the animal before it can be digested and absorbed.

We have already claimed that this abstraction has practical applications and now give a detailed example to better motivate the subject of this section. In [Randell *et al.*, 2013] the authors use a discrete version of the spatial logic RCC (from which the constraint language RCC-8 is derived) called DM (for Discrete Mereotopology) to model the topological organization of segmented cells and their parts and cellular structure in tissue. The domain model assumes an a priori constraint that cell nuclei form parts of their host cells, however in the example shown in Figure 4.1 the RCC-8 relation returned is *partially overlaps* and not *proper part*. There are several reasons why this scenario may happen in practice, e.g., if the regions initially segmented out as cell nuclei are being over-segmented, or variations in the histological stain density results in a less than optimal threshold level being selected. The result means the labelled regions extracted from the image cannot be a model. The task then is to repair the segmentation to restore consistency and/or optimise the sequence of segmentation steps needed. As such, a conceptual neighbourhood graph for DM is used to encode legal topological transitions, and successive states from a start to end state are generated, and then optimised. Paths through the network are then cashed out as a series of image processing segmentation steps. A single histological image may have many hundreds of cells, and the generation of symbolic models may or may not be realised in an actual image. Moreoever, some segmentation operations on regions will reduce their size

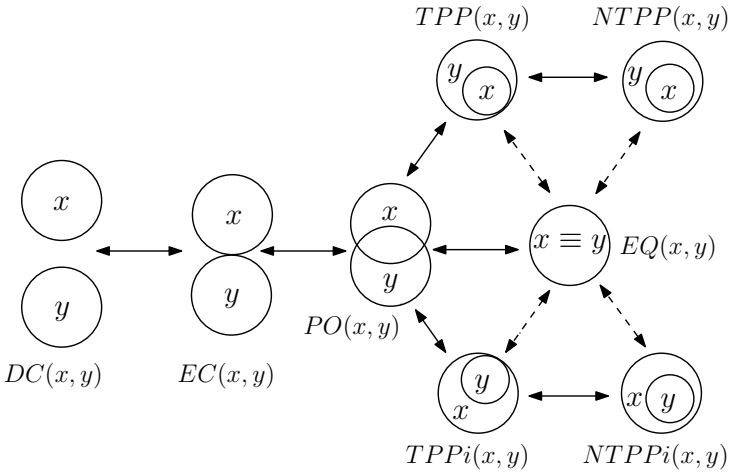


Figure 4.2: A conceptual neighbourhood graph of RCC-8

and may fragment a region into sub-parts, or separated regions that increase their size may merge, so the computational task of finding an optimal segmentation model can easily grow in complexity.

We have mentioned the notion of a conceptual neighbourhood graph in our previous brief introduction, but before formally defining it, we need to introduce the concept of conceptually neighbouring relations for a given qualitative constraint language. This concept is strongly related to the continuity and proximity that these relations might exhibit. In particular, we recall the following definition from [Freksa, 1991]:

Definition 25 ([Freksa, 1991]) *Given a qualitative constraint language based on a finite set of base relations* B *defined over a domain* D*, we have that two base relations* $b, b' \in \mathsf{B}$ *are* conceptual neighbours *with respect to a pair of entities, if they can be directly transformed into one another through continuous deformation (e.g., in shape, size, or position) of the entities.*

As an example, in RCC-8 the base relations DC and EC with respect to a pair of entities (x, y) are conceptual neighbours, since a continuous movement of the spatial entity x towards spatial entity y may cause a direct transition from relation DC to relation EC. On the other hand, and again with respect to the pair of entities (x, y), the relations DC and PO are not conceptual neighbours since a transition between those relations must go through relation EC. Another example in Interval Algebra considers relations m (meets) and o (overlaps). These relations are conceptual neighbours since an entity can directly overlap another entity after having met it first.

Clearly, by Definition 25 it follows that every base relation is a conceptual neighbour of itself, however, we do not depict any loops in our graphs in what follows for simplicity. Conceptually neighbouring relations in any given qualitative constraint language can be captured with a conceptual neighbourhood graph, which is defined as follows.

Definition 26 ([Freksa, 1991]) *Given a qualitative constraint language based on a finite set of base relations* B *defined over a domain* D*, a* conceptual neighbourhood graph[10] *of that language is a graph* $\Gamma = (\mathsf{B}, E)$ *where* $E = \{\{b, b'\} \mid b, b' \in \mathsf{B}$*, and* b *and* b' *are conceptual neighbours*$\}$.

[10] Actually, Freska in [Freksa, 1991] uses the term *neighbourhood structure* to describe what we formally define here as a graph.

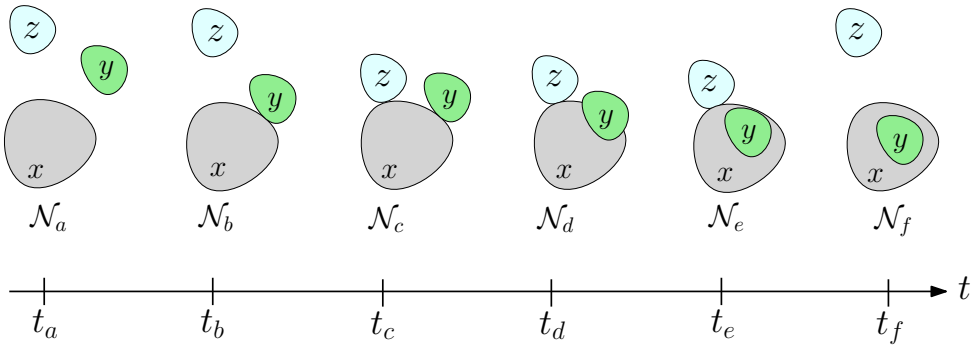


Figure 4.3: Example of a spatio-temporal sequence based on RCC-8

Conceptual neighbourhood graphs can be established for all qualitative constraint languages, a subset of which can be found in [Freksa, 1991; Santos and Moreira, 2009; Egenhofer, 2010]. It is important to note that conceptual neighbourhood graphs are not unique for a qualitative constraint language as they can be subject to further restrictions, such as constraints subject to user preference, or restrictions on deformation.

As an example, the conceptual neighbourhood graph of RCC-8 is depicted in Figure 4.2. The dashed edges represent transitions of base relations that are not allowed if we require that regions do not change size. Indeed, if we require that regions do not change size, it is impossible for a region x that is properly contained in another region y to be equal to y in a following point of time, as x cannot grow bigger and y cannot become smaller.

Next, we define the notion of a qualitative spatio-temporal sequence, which is nothing more than a sequence of spatial QCNs. The ordering of the QCNs in the aforementioned sequence constitutes a timeline that can potentially allow us to view how a spatial configuration evolves over time. We can define a qualitative spatio-temporal sequence (QSS) as follows.

Definition 27 *A qualitative spatio-temporal sequence (QSS) $\mathcal{S}$ is a sequence $(\mathcal{N}_1 = (V, C_1)$, $\mathcal{N}_2 = (V, C_2)$, ..., $\mathcal{N}_k = (V, C_k))$ of k QCNs over a set of n variables V, for some positive integers k and n.*

An atomic QSS is a QSS that comprises only atomic QCNs. Further, a solution and a scenario of a QSS is the sequence of solutions and scenarios of all its QCNs respectively. Other notions of QSSs, such as equivalence or consistency, are completely analogous to those of QCNs.

An example of an atomic spatio-temporal sequence based on RCC-8 is given in Figure 4.3. Figure 4.3 depicts the sequence $(\mathcal{N}_a = (V, C_a)$, $\mathcal{N}_b = (V, C_b)$, $\mathcal{N}_c = (V, C_c)$, $\mathcal{N}_d = (V, C_d)$, $\mathcal{N}_e = (V, C_e)$, $\mathcal{N}_f = (V, C_f))$, where $V = \{x, y, z\}$ and $\mathcal{N}_a$, $\mathcal{N}_b$, $\mathcal{N}_c$, $\mathcal{N}_d$, $\mathcal{N}_e$, and $\mathcal{N}_f$ are RCC-8 configurations over V. In particular, $\mathcal{N}_a$ defines the set of constraints $\{DC(x, y)$, $DC(y, z)$, $DC(x, z)\}$, $\mathcal{N}_b$ defines the set of constraints $\{EC(x, y)$, $DC(y, z)$, $DC(x, z)\}$, $\mathcal{N}_c$ defines the set of constraints $\{EC(x, y)$, $DC(y, z)$, $EC(x, z)\}$, $\mathcal{N}_d$ defines the set of constraints $\{PO(x, y)$, $DC(y, z)$, $EC(x,\ z)\}$, $\mathcal{N}_e$ defines the set of constraints $\{TPPi(x, y)$, $DC(y, z)$, $EC(x, z)\}$, and finally $\mathcal{N}_f$ defines the set of constraints $\{NTPPi(x, y)$, $DC(y, z)$, $DC(x,\ z)\}$. Each spatial QCN in the sequence corresponds to a unique point of time in the timeline t. For example, spatial configuration $\mathcal{N}_c$ corresponds to the point of time t_c in the timeline t. Thus, the ordering of the spatial QCNs in a given sequence yields a spatio-temporal configuration that describes how a spatial configuration evolves over time.

At this point, we can extend the notion of conceptually neighbouring relations to the notion of conceptually neighbouring atomic QCNs as follows.

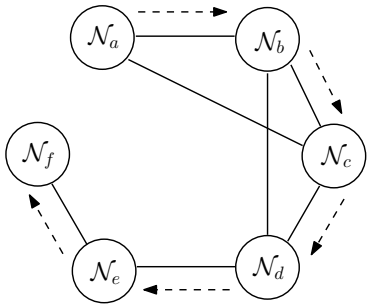


Figure 4.4: Transition graph of the spatio-temporal sequence in Figure 4.3

Definition 28 *Given a qualitative constraint language and a conceptual neighbourhood graph* Γ *of that language, we have that two atomic* QCN*s* $\mathcal{N} = (V, C)$ *and* $\mathcal{N}' = (V, C')$ *are conceptual neighbours with respect to* Γ *if* $\forall u, v \in V$ *we have that* b *and* b' *are conceptual neighbours with respect to* Γ*, where* b *and* b' *are the base relations corresponding to the singleton relations* $C(u, v)$ *and* $C'(u, v)$ *respectively.*

Intuitively, two atomic QCNs are conceptual neighbours if they can *transition* from one another through a continuous (and possible even simultaneous) transformation of their base relations to conceptually neighbouring base relations. We can also give the following definition of a conceptual neighbourhood graph for a set of atomic QCNs, but to avoid any confusion with the conceptual neighbourhood graph of the base relations of a qualitative constraint language we will refer to it as a transition graph:[11]

Definition 29 *Given a qualitative constraint language, a conceptual neighbourhood graph* Γ *of that language, and a satisfiable atomic* QSS $\mathcal{S} = (\mathcal{N}_1, \mathcal{N}_2, \ldots, \mathcal{N}_k)$*, the transition graph of* $\mathcal{S}$ *defined with respect to* Γ *is the graph* $M = (\{\mathcal{N}_1, \mathcal{N}_2, \ldots, \mathcal{N}_k\}, E)$ *where* $E = \{\{\mathcal{N}_i, \mathcal{N}_j\} \mid \mathcal{N}_i, \mathcal{N}_j \in \{\mathcal{N}_1, \mathcal{N}_2, \ldots, \mathcal{N}_k\}$*; and* $\mathcal{N}_i$ *and* $\mathcal{N}_j$ *are conceptual neighbours with respect to* $\Gamma\}$.

The transition graph of a satisfiable atomic QSS $\mathcal{S}$ of k QCNs encodes all the conceptually allowed transitions between its spatial QCNs, i.e., it encodes all the pairs of atomic QCNs that are conceptual neighbours with respect to an assumed conceptual neighbourhood graph. Clearly, if the QCNs are defined over a set of variables V, it takes $O(|V|^2)$ time to calculate if a transition is possible between the QCNs of a given pair of QCNs. As the transition graph of $\mathcal{S}$ has k nodes and, thus, $O(k^2)$ possible edges, i.e., pairs of QCNs, obtaining the entire transition graph can be done in polynomial time. It is also the case that every node, i.e., every QCN, in a transition graph is a conceptual neighbour of itself.

As an example, the transition graph of the spatio-temporal sequence depicted in Figure 4.3 defined with respect to the conceptual neighbourhood graph in Figure 4.2, is shown in Figure 4.4. Indeed, we can have continuous transitions between the spatial QCNs in the pairs $(\mathcal{N}_a, \mathcal{N}_b)$, $(\mathcal{N}_b, \mathcal{N}_c)$, $(\mathcal{N}_c, \mathcal{N}_d)$, $(\mathcal{N}_d, \mathcal{N}_e)$, $(\mathcal{N}_e, \mathcal{N}_f)$ of consecutive QCNs in the sequence $(\mathcal{N}_a, \mathcal{N}_b, \mathcal{N}_c, \mathcal{N}_d, \mathcal{N}_e, \mathcal{N}_f)$, but also continuous transitions between spatial configurations $\mathcal{N}_a$ and $\mathcal{N}_c$ (i.e., the pair $(\mathcal{N}_a, \mathcal{N}_c)$), and $\mathcal{N}_b$ and $\mathcal{N}_d$ (i.e., the pair $(\mathcal{N}_b, \mathcal{N}_d)$).

Now, we introduce the main reasoning problem of this section. Given a qualitative spatio-temporal sequence of QCNs defined over a qualitative constraint language and a conceptual

[11] In fact, the reader can easily verify that in the case where we have the set of all possible atomic QCNs over exactly two spatial entities for a given qualitative constraint language, the transition graph defined by those QCNs corresponds to the conceptual neighbourhood graph of that language.

neighbourhood graph of that language, we would like to solve the QCNs and extract scenarios of them such that the scenarios in every pair of consecutive scenarios in the obtained sequence are conceptual neighbours with respect to the conceptual neighbourhood graph. We call this problem the sequence solving problem (SSP) and define it as follows.

Definition 30 *Given a qualitative constraint language, a conceptual neighbourhood graph* Γ *of that language, and a* QSS $\mathcal{S} = (\mathcal{N}_1, \mathcal{N}_2, \ldots, \mathcal{N}_k)$*, the* SSP *for* $\mathcal{S}$ *is the problem of obtaining a scenario* $(\mathcal{S}_1, \mathcal{S}_2, \ldots, \mathcal{S}_k)$ *of* $\mathcal{S}$ *such that the scenarios* $\mathcal{S}_i$ *and* $\mathcal{S}_{i+1}$ *in every pair of consecutive scenarios* $(\mathcal{S}_i, \mathcal{S}_{i+1})$ *in the sequence are conceptual neighbours with respect to* Γ*.*

Equivalently, the SSP for a given qualitative spatio-temporal sequence $\mathcal{S} = (\mathcal{N}_1, \mathcal{N}_2, \ldots, \mathcal{N}_k)$ and a respective conceptual neighbourhood graph Γ, can be defined as the problem of obtaining a scenario $(\mathcal{S}_1, \mathcal{S}_2, \ldots, \mathcal{S}_k)$ of $\mathcal{S}$ such that $(\mathcal{S}_1, \mathcal{S}_2, \ldots, \mathcal{S}_k)$ itself defines a path in the transition graph of that scenario with respect to Γ. For example, it is easy to see that the pairs of consecutive QCNs of the satisfiable atomic spatio-temporal sequence depicted in Figure 4.3, correspond to a path illustrated with dashed arrows in the transition graph of that sequence depicted in Figure 4.4. In this particular example, as the sequence is satisfiable and atomic, the aforementioned QCNs are scenarios of themselves. As such, $(\mathcal{N}_a, \mathcal{N}_b, \mathcal{N}_c, \mathcal{N}_d, \mathcal{N}_e, \mathcal{N}_f)$, as viewed in Figure 4.4, defines a path in the transition graph of the unique scenario of the spatio-temporal sequence depicted in Figure 4.3 with respect to the conceptual neighbourhood graph of RCC-8 depicted in Figure 4.2.

We then have the following result from Westphal et al. in [Westphal *et al.*, 2013]:

Theorem 17 ([Westphal *et al.*, 2013]) *Let* $\mathcal{S}$ *be a* QSS *of Point Algebra and* Γ *the usual conceptual neighbourhood graph of Point Algebra, viz.,* $(\{<, =, >\}, \{\{<, =\}, \{=, >\}\})$ *(omitting loops). Then, deciding whether there is a solution of the* SSP *for* $\mathcal{S}$ *with respect to* Γ *is a* $\mathcal{NP}$*-complete problem.*

Westphal et al. present another interesting result that provides a sufficient condition to decide whether there is a solution of the SSP for a given qualitative spatio-temporal sequence of Point Algebra. The result follows.

Proposition 18 ([Westphal *et al.*, 2013]) *Let* $\mathcal{S} = (\mathcal{N}_1 = (V, C_1), \mathcal{N}_2 = (V, C_2), \ldots, \mathcal{N}_k = (V, C_k))$ *be a* QSS *of Point Algebra and* Γ *the usual conceptual neighbourhood graph of Point Algebra, viz.,* $(\{<, =, >\}, \{\{<, =\}, \{=, >\}\})$ *(omitting loops). Then, there is a solution of the* SSP *for* $\mathcal{S}$ *with respect to* Γ *if:*

- *for each* $1 \leq k' \leq k$*, we have that* $\mathcal{N}_{k'}$ *is not trivially inconsistent and* $\diamond$*-consistent;*
- *for each* $1 \leq k' < k$*, and for any scenario* $\mathcal{S}_{k'}{\downarrow_{V'}}$ *of* $\mathcal{N}_{k'}$ *and any scenario* $\mathcal{S}_{k'+1}{\downarrow_{V'}}$ *of* $\mathcal{N}_{k'+1}$ *both restricted to any same set of two variables* $V' \subseteq V$*, we have that* $\mathcal{S}_{k'}{\downarrow_{V'}}$ *and* $\mathcal{S}_{k'+1}{\downarrow_{V'}}$ *are conceptual neighbours with respect to* Γ*.*

As noted in [Westphal *et al.*, 2013], the result of Proposition 18 might help to define interesting heuristics and domain splitting strategies in qualitative constraint-based spatial and temporal reasoning for arbitrary QCNs, much like what was intended with the domain-based decomposition scheme in [Broxvall, 2002] that we explored in Section 3.7. Further, the aforementioned results naturally extend to other point-based formalisms such as, e.g., Cardinal Direction Calculus, Interval Algebra, and Block Algebra and are thus important.

The results of this section consider the simple notion of a conceptual neighbourhood graph as defined by Freska in [Freksa, 1991], which encodes the conceptual proximity and continuity between the base relations of a qualitative constraint language in a highly abstract manner. However, this high level of abstraction can be inadequate for many form of continuous motion, since trasnition graphs based on conceptual neighbourhood graphs only consider the spatial change of one entity without taking into account simultaneous spatial changes of other entities at a given point of time. Due to this limination, there have been extensions of conceptual neighbourhood graphs defined in the literature, such as the notion of *directed* conceptual neighbourhood graphs proposed by Galton in [Galton, 2001] and the concept of *generalized* conceptual neighbourhood graphs presented by Ragni et al. in [Ragni and Wölfl, 2005]. Directed conceptual neighbourhood graphs characterize continuity by means of *dominance* and *dominance spaces*, which formalize the notion of continuous change in the context of qualitative reasoning. As a motivational example towards defining dominance, Galton considers a quantitative entity X with real values. This entity can be described qualitatively by saying that is strictly positive (state p), strictly negative (state n), or zero (state z). Then, by considering some time interval i, if X is in state n thoughout i, and if it is in another state in one of the bounding points of i, this other state can only be z, hence, z dominates n. Galton goes on to introduce the concept of dominance between qualitative states and shows how standard conceptual neighbourhood graphs can be extended to directed ones. Ragni et al. deal with the question whether there exists a continuous transition from an initial spatial configuration to a final spatial configuration when the set of all possible transitions is restricted by certain transition graphs. They define the concept of generalized conceptual neighbourhood graphs, which extend conceptual neighbourhood graphs as we described them in this section, in the sense that they encode possible transitions between spatial configurations of a fixed number of base relations and not just between base relations. As such, generalized conceptual neighbourhood graphs can be seen as particular cases of transition graphs (in the way they are explained in Definition 29). However, an additional parameter in the definition of a generalized conceptual neighbourhood graph enables one to exactly specify how many spatial entities are allowed to change (e.g., in shape, size, or orientation) for a continuous transition between two spatial configurations to take place. This allows for reasoning about spatial objects in stricter and more user-controllable dynamic settings. It should also be clear that we can examine qualitative spatio-temporal sequences of some calculus with respect to any graph, even user defined ones that may have nothing to do with continuity and proximity between spatial configurations.

Finally, it is important to note that the notion of continuous transitions between spatial configurations is closely related to Qualitative Simulation [Kuipers, 1985; Kuipers, 1986; Kuipers, 1993; Kuipers, 1994], at least as it appears in the context of qualitative spatial and temporal reasoning. In particular, qualitative simulation starts with a qualitative differential equation model (QDE), which is an abstraction of an ordinary differential equation that consists of a set of real-valued variables and functional, algebraic, and differential constraints among them, and a qualitative description of an initial state of that QDE model. Given a qualitative description of a state, qualitative simulation predicts all the possible qualitative state descriptions that can be direct successors of the current state description. By repeating this process we get a graph of qualitative state descriptions, called a *behavior graph*, in which the paths starting from the root are the possible qualitative behaviors. The graph of qualitative states is pruned according to criteria derived from the theory of ordinary differential equations, in order to preserve the guarantee that all possible behaviors are predicted. As such, qualitative simulation predicts the set of possible behaviors consistent with a qualitative differential equation model of the world. Its value comes from the ability to express natural types of incomplete knowledge of the world, and

the ability to derive a provably complete set of possible behaviors in spite of the incompleteness of the model. In the context of qualitative spatial and temporal reasoning, we could consider our initial state to be that of a spatial or temporal configuration. Then, we would like to compute all the possible direct successors of that state description, which in this case would correspond to all the possible spatial configurations that could be obtained through continuous transitions from the initial state. By repeating this process we would get a transition graph of spatial configurations, in which the paths starting from the root would be the possible spatial configurations. In fact, this type of qualitative spatio-temporal simulation has been considered in the work of Cui et al. in [Cui *et al.*, 1992], where a qualitative simulation program for handling topological information is presented. It is interesting to note that the algorithm of [Cui *et al.*, 1992] is illustrated with a real-world example of a simulation of phagocytosis and exocytosis, two processes used by unicellular organisms for garnering food and expelling waste material respectively. Before closing off this section, it is worth mentioning the *Transition Calculus* of Gooday et al. [Gooday and Galton, 1997]. The Transition Calculus serves as a high-level approach in which transitions are directly modelled in terms of the state changes that they bring about. As such, the Transition Calculus can provide a more natural way of representing and solving many transition and change problems, and, therefore, it also naturally extends to the AI planning domain. It should be mentioned that the Transition Calculus is primarily intended as a purely high-level formalism and, thus, it does not restrict itself to the notion of continuous transitions per se. However, spatial or temporal semantics about continuity can be encoded in the calculus and enable it to operate as a framework for qualitative spatio-temporal simulation as well. With respect to the notion of continuity and continuous changes in the spatial domain, Galton makes a thorough study in [Galton, 1997; Galton, 2000] for both qualitative and quantitative spatial state representations. A general work that introduces a constraint-based framework for studying infinite qualitative simulations concerned with contingencies such as time, space, shape, size, abstracted into a finite set of qualitative relations, is presented in [Apt and Brand, 2006]. In particular, in [Apt and Brand, 2006] qualitative simulations are defined by a combination of constraints that formalize the background knowledge concerned with qualitative reasoning and appropriate inter-state constraints that are formulated using linear temporal logic.

4.4 Combining RCC-8 and Interval Algebra

In the previous sections, we have discussed combinations of qualitative spatial constraint languages with certain modal logics to create an hierarchy of qualitative spatio-temporal logics, and we have also reviewed the case where a sequence of spatial QCNs is used to describe continuous spatial change of spatial entities over the timeline that underlies the sequence. In this section we present a spatio-temporal formalism that it based entirely on constraints.

The dominant qualitative spatial and temporal constraint languages in Artificial Intelligence are RCC-8 and Interval Algebra respectively. It is then only natural to combine these two languages into a unique qualitative constraint-based spatio-temporal formalism. In [Gerevini and Nebel, 2002], the authors use Interval Algebra to temporalize RCC-8 and present the *spatio-temporal constraint calculus* (STCC). For example, given an RCC-8 formula $PO(x, y)$, they assign to it a temporal interval i, which means that the RCC-8 formula must be true at every point between the endpoints of the interval i. In a sense, we can view a STCC network as a QCN of Interval Algebra that has a set of spatial QCNs as its set of variables. Formally, a STCC is defined as follows.

Definition 31 *A* spatio-temporal constraint calculus STCC *network is a* QCN (W, R) *of Interval*

Algebra where:

- $W = \{\mathcal{N}_1 = (V, C_1), \mathcal{N}_2 = (V, C_2), \ldots, \mathcal{N}_k = (V, C_k)\}$ *is a non-empty finite set of spatial* QCN*s of* RCC-*8 defined over a non-empty finite set of variables* V *corresponding to spatial entities;*
- R *is the usual constraint mapping in a* QCN *as defined in Definition 4.*

Then, a solution of a STCC network is defined as follows.

Definition 32 *Let* $\mathcal{N} = (W, R)$ *be a* STCC *network over* n *spatial* QCN*s of* RCC-*8. A* solution *of* $\mathcal{N}$ *consists of:*

- *a solution of the underlying* QCN *of Interval Algebra where the* n *spatial* QCN*s of* RCC-*8 are viewed as variables of the Interval Algebra network;*
- *a solution for each of the* n *spatial* QCN*s of* RCC-*8 that holds at every point between the endpoints of the temporal interval upon which each spatial* QCN *of* RCC-*8 is defined.*

As illustration, we can have the simple STCC network $\mathcal{N} = (W, R)$, where W is the set of QCNs of RCC-8 $\{\mathcal{N}_1, \mathcal{N}_2\}$ defined by the sets of constraints $\{PO(x, y)\}$ and $\{EC(x, y)\}$ respectively, and R is a mapping that associates the singleton Interval Algebra relation $\{m\}$ (meets) with the previous pair of spatial QCNs (which are treated as if they were the temporal variables of the underlying QCN of Interval Algebra). As such, the underlying QCN of Interval Algebra is defined by the singleton constraint $\{m(\mathcal{N}_1, \mathcal{N}_2)\}$.

Let σ be a solution of the STCC network $\mathcal{N}$. Then σ will comprise a solution of the underlying QCN of Interval Algebra, denoted by σ_{IA}, and a solution of each of the spatial QCNs $\mathcal{N}_1$ and $\mathcal{N}_2$, denoted by σ_1 and σ_2, respectively. Let as assume that σ_{IA} is such that $\sigma_{\mathsf{IA}}(\mathcal{N}_1) = [0, 1]$ and $\sigma_{\mathsf{IA}}(\mathcal{N}_2) = [1, 2]$. Then, the solution σ_1 of $\mathcal{N}_1$ holds at every point between the endpoints of interval $[0, 1]$, i.e., it is true for interval $(0, 1)$, and, likewise, the solution σ_2 of $\mathcal{N}_2$ holds at every point between the endpoints of interval $[1, 2]$, i.e., it is true for interval $(1, 2)$.

As checking whether a STCC network has a solution involves checking whether the associated QCNs of Interval Algebra and RCC-8 have a solution, it comes rather natural that checking whether a STCC network has a solution is at least as difficult as checking whether a QCN of Interval Algebra or RCC-8 has a solution, which is a NP-complete problem (see Section 3.3.1). This bears the question of how much more the combination of Interval Algebra and RCC-8 increases the complexity of checking whether a STCC network has a solution. Hopefully, the complexity does not increase at all. In particular, we have the following result:

Theorem 18 ([Gerevini and Nebel, 2002]) *Deciding whether there is a solution of a* STCC *network* $\mathcal{N}$ *is a* NP*-complete problem.*

The result of Theorem 18 can be explained by the fact that given a STCC, one only needs polynomially many points in the real line to construct a solution of the underlying QCN of Interval Algebra, as well as polynomially many solutions of the associated QCNs of RCC-8. Moreover, each solution of a QCN of RCC-8 can be constructed using size polynomial in the size of the QCN, by employing some canonical model that allows this, such a model being for example the one defined by Renz in [Renz, 2002a]. In all fairness to Gerevini and Nebel, and as noted also by them in [Gerevini and Nebel, 2002], the result of Theorem 18 is strongly implied by the work of Bennett et al. in [Bennett *et al.*, 2002]. In particular, Bennett et al. describe a very similar

approach to the one of Gerevini et al. for spatio-temporal reasoning, by combining Interval Algebra with BRCC-8 instead of RCC-8. We remind the reader that BRCC-8 is an extension of RCC-8 that allows the use of Boolean variable terms as arguments of the RCC-8 predicates, where a Boolean variable term is a combination of spatial variables using the Boolean operators $\vee$, $\wedge$, and $\neg$. Another difference lies in the fact that Bennett et al. require that a solution of an associated QCN of BRCC-8 in a given STCC network holds not only at every point between the endpoints of the temporal interval upon which the spatial QCN of BRCC-8 is defined, but also at the endpoints of that temporal interval. Thus, the approach of Gerevini et al. is a little more flexible than the approach of Bennett et al. as it allows spatial configurations to change at endpoints of temporal intervals.

There are some obvious shortcomings with the use of a STCC network to represent and reason with spatio-temporal information. For example, given a STCC network $\mathcal{N} = (\{\mathcal{N}_1, \mathcal{N}_2\}, R)$, such that the QCNs of RCC-8 $\mathcal{N}_1$ and $\mathcal{N}_2$ do not share any common solutions, we can only have that $R(\mathcal{N}_1, \mathcal{N}_2) \subseteq \{m, mi, p, pi\}$. This is because since $\mathcal{N}_1$ and $\mathcal{N}_2$ do not share any common solutions, it would be impossible to have a common solution that would be required in the case where the two temporal intervals that are associated with the networks $\mathcal{N}_1$ and $\mathcal{N}_2$ respectively overlapped; in more detail, a common solution would be required for both networks over that overlapping part of the two intervals, as any two entities can only be given a single valuation at a specific point of time. For instance, let us consider the simple STCC network $\mathcal{N} = (W, R)$, where W is the set of QCNs of RCC-8 $\{\mathcal{N}_1, \mathcal{N}_2\}$ defined by the sets of constraints $\{DC(x, y)\}$ and $\{EQ(x, y)\}$ respectively. It is clear that these two networks do not share any common solution. Let us assume that R is a mapping that associates any of the singleton Interval Algebra relations defined by $\mathsf{B} \setminus \{m, mi, p, pi\}$, for example, $\{d\}$ (during), with the previous pair of spatial QCNs. As such, the underlying QCN of Interval Algebra is defined by the singleton constraint $\{d(\mathcal{N}_1, \mathcal{N}_2)\}$. A solution of the STCC network $\mathcal{N}$ should comprise a solution of the underlying QCN of Interval Algebra, denoted by σ_{IA}, and a solution of each of the spatial QCNs $\mathcal{N}_1$ and $\mathcal{N}_2$. Let as assume that σ_{IA} is such that $\sigma_{\mathsf{IA}}(\mathcal{N}_1) = [0, 1]$ and $\sigma_{\mathsf{IA}}(\mathcal{N}_2) = [0, 2]$. This is indeed a valid solution of the underlying QCN of Interval Algebra, as it satisfies base relation $d(\mathcal{N}_1, \mathcal{N}_2)$. Next, we need to have a common solution for each of the QCNs $\mathcal{N}_1$ and $\mathcal{N}_2$ that will hold at every point between the endpoints of interval $[0, 1]$ including the endpoint 1 itself (as it does not belong to a specified Interval Algebra temporal interval, but occurs as a byproduct), i.e., that it will be true for interval $(0, 1]$. This is not possible, as $\mathcal{N}_1$ and $\mathcal{N}_2$ do not share any common solutions. Thus, in this particular case R should be a mapping that associates only some subrelation of the Interval Algebra relation defined by $\{m, mi, p, pi\}$ with the pair of spatial QCNs $\mathcal{N}_1$ and $\mathcal{N}_2$. However, even the small Interval Algebra relation defined by $\{m, mi, p, pi\}$ itself, leads to the total set of relations of Interval Algebra when combined with the singleton relations of Interval Algebra and closing the resulting set of relations under intersection, weak composition, and converse. Due to this issue, and considering also the result of Theorem 18 as well as the fact that it is NP-hard to decide the satisfiability of an arbitrary QCN of Interval Algebra, we have the following result:

Theorem 19 ([Gerevini and Nebel, 2002]) *Let $\mathcal{N}$ be a* STCC *network defined over the set of singleton and universal relations of* RCC-*8 and Interval Algebra. Then, deciding whether there is a solution of $\mathcal{N}$ is a* NP-*complete problem.*

The result of Theorem 19 does not leave much room for identifying large tactable fragments of STCC, in contrast to the case with Interval Algebra and RCC-8 for which large tractable subclasses of their relations have been indeed identified. This result also partially addresses an open problem raised by Wolter et al. in [Wolter and Zakharyaschev, 2003], who questioned the

existense of such large tractable fragments of STCC. Nevertheless, there exists a case where the existence of a solution of an STCC network can be decided in polynomial time. In particular, we have the following result:

Theorem 20 ([Gerevini and Nebel, 2002]) *Let $\mathcal{N}$ be a* STCC *network such that the underlying* QCN *of Interval Algebra forms a scenario, and each of the associated* QCN*s of* RCC-*8 is defined over a tractable subclass of relations. Then, deciding whether there is a solution of $\mathcal{N}$ can be achieved in polynomial time.*

Another shortcoming of STCC as a spatio-temporal calculus, is that it does not permit one to state general laws of how the spatial configurations associated with a STCC network change; the spatial configurations associated with a STCC network are unrelated to one another. Thus, one cannot state for example that regions cannot change their size, or that spatial changes should occur continuously. However, Gerevini et al. address this issue by forcing an additional set of contraints into an STCC instance to the effect that changes have to be continuous and that regions do not change size. Gerevini et al. go on to show that deciding whether there is a solution of a given STCC network under the continuous change and size persistence constraints remains a NP-complete problem. This is due the fact that given a candidate solution of a STCC network, we can check if the continuous change and size persistence constraints are satisfied by this solution in polynomial time.

An analogous effort to that of Gerevini et al. [Gerevini and Nebel, 2002] for presenting a spatio-temporal formalism that it based entirely on constraints, is given by Ragni et al. in [Ragni and Wölfl, 2006]. In particular, Ragni et al. investigate a constraint formalism that temporalizes the Cardinal Direction Calculus using the Interval Algebra, much like Gerevini et al. temporalize RCC-8. Further, Ragni et al. place emphasis on how continuous change of objects in Cardinal Direction Calculus is reflected in changes of the respective qualitative relations expressing these relative positions. They show how continuous change can be represented as a set of operations to objects in grid-like structures, and based on this representation they propose a method for encoding temporalized spatial constraint satisfaction problems as deterministic planning problems.

4.5 Spatio-Temporal Periodicity

In this section, we capture the notion of an ultimately periodic qualitative constraint network (UPQCN) [Condotta *et al.*, 2005]. A UPQCN is a temporalized QCN that evolves over time with a recurrent pattern. In particular, let us consider a set of spatial entities whose spatial location may change over time. At each point of time, an entity is associated with a given location. We wish to be able to express the following three types of constraints: constraints between the locations of two entities at a given point of time, constraints between the locations of two entities at distinct points of time, and constraints between the locations of two entities that have to be satisfied at all points of time following an initial point of time.

Let us assume that each integer $t \geq 0$ corresponds to a point of time. Then, a UPQCN allows us to represent qualitative constraints like the ones described in the following example. Let us consider the qualitative constraint language of Point Algebra, which we will interpret in spatial terms. In what follows, all considered entities are points on the rational line. With that being said, we have three entities X, Y, and Z. The entities change positions over time according to the following constraints: at point of time $t = 0$, X is left of both Y and Z; regarding points of time $t = 0$ and $t = 1$, either the location of Z at point of time $t = 0$ is left of that at point of

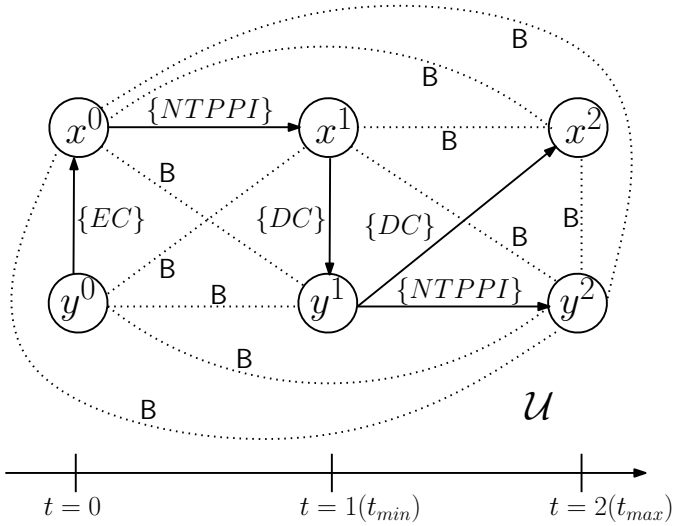


Figure 4.5: An example UPQCN $\mathcal{U}$ of RCC-8

time $t = 1$, or they coincide; regarding point of time $t = 1$ and all following points of time $t \geq 1$, the location of X is right of Y; for all points of time $t \geq 1$, X moves to the left and Y moves to the right; for all points of time $t \geq 2$, Z moves to the left and it stays to the left of X and to the right of Z.

As we saw in our aforementioned example, a UPQCN allows us to specify certain periodic constraints, which would be impossible to do with the use of a simple QCN. A UPQCN is formally defined as follows.

Definition 33 *A* UPQCN *is a structure* (V, C, t_{min}, t_{max})*, where* $V = \{v_0^0, \ldots, v_n^0, \ldots, v_0^{t_{max}}, \ldots, v_n^{t_{max}}\}$ *is a finite set of variables,* t_{min} *and* t_{max} *are two integers such that* $0 \leq t_{min} \leq t_{max}$*, and* C *is a mapping that associates a relation* $C(v, v') \in 2^{\mathsf{B}}$ *with each pair* (v, v') *of* $V \times V$*. Mapping* C *is such that* $C(v, v) = \{\mathsf{Id}\}$ *and* $C(v, v') = (C(v', v))^{-1}$ *for every* $v, v' \in V$*.*

An example UPQCN of RCC-8 is shown in Figure 4.5. Intuitively, in a spatial context, each variable $v^t \in V$ represents the occurrence of the spatial component of entity v at point of time t, with $t \in \mathbb{N}$. $C(u^t, v^{t'})$ is a constraint on the relative positions of the occurrence of u at point of time t and that of v at point of time t'. The constraints expressed by C are twofold: firstly, all constraints from 0 up to t_{max} have to be satisfied; secondly, all constraints from t_{min} to t_{max} have to be satisfied up to t_{max}, but also on all subsequent periods $\{t_{min} + t, \ldots, t_{max} + t\}$, where $t \in \mathbb{N}$. In other words, the structure defines both initial constraints (up to t_{max}), and a recurrent pattern of constraints (from t_{min} to t_{max}), the *motif*, which repeats itself indefinitely. The *motif* of a UPQCN $\mathcal{U}$ is defined as follows.

Definition 34 *Let* $\mathcal{U} = (V, C, t_{min}, t_{max})$ *be a* UPQCN *over* n *entities. The* motif *of* $\mathcal{U}$*, denoted by* $motif(\mathcal{U})$*, is the* QCN $\mathcal{N}_m = (V_m, C_m)$*, where* $V_m = \{v^t \mid v^t \in V \text{ and } t \leq lg\}$*, with* $lg = t_{max} - t_{min}$*, and* $\forall m, m' \in \{0, \ldots, n\}$ *and* $\forall k, k' \in \{0, \ldots, lg\}$*,* $C_m(v_m^k, v_{m'}^{k'}) = C(v_m^{k+t_{min}}, v_{m'}^{k'+t_{min}})$*.*

As illustration, the motif of the example UPQCN $\mathcal{U}$ of RCC-8 shown in Figure 4.5 is provided in Figure 4.6.

As in the case of a QCN, given a UPQCN $\mathcal{U}$ we are mainly interested in solving the satisfiability problem associated with it, i.e., deciding whether $\mathcal{U}$ admits a solution. A solution of a UPQCN is defined as follows.

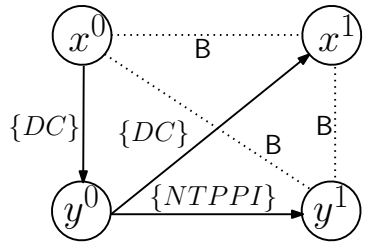


Figure 4.6: The motif of the UPQCN $\mathcal{U}$ of RCC-8 shown in Figure 4.5

Definition 35 *Let $\mathcal{U} = (V, C, t_{min}, t_{max})$ be a* UPQCN *over n entities. A* solution *of $\mathcal{U}$ is a valuation σ of the variables of V such that for each pair of variables $(u^{t_i}, v^{t_j}) \in V$ with $t_i < t_j$ we have:*

- *if $t_j \leq t_{max}$, then $\sigma(u^{t_i})$ and $\sigma(v^{t_j})$ satisfy $C(u^{t_i}, v^{t_j})$, i.e., there exists a base relation $b \in C(u^{t_i}, v^{t_j})$ such that $(\sigma(u^{t_i}), \sigma(v^{t_j})) \in b$;*
- *if $t_i \geq t_{min}$ and $t_j - t_i \leq t_{max} - t_{min}$ then for all t_i', t_j' such that $t_{min} \leq t_i' \leq \min\{t_{max}, t_i\}$, $t_{min} \leq t_j' \leq \min\{t_{max}, t_j\}$ and $t_j - t_i = t_j' - t_i'$, we have that $\sigma(u^{t_i})$ and $\sigma(v^{t_j})$ satisfy $C(u^{t_i'}, v^{t_j'})$.*

A solution of the example UPQCN $\mathcal{U}$ of RCC-8 shown in Figure 4.5 is provided in Figure 4.7. Note that the valuation of the variables of $\mathcal{U}$ satisfy all of the constraints up to t_{max} $(t = 2)$, but also all of the constraints specified by the motif of $\mathcal{U}$. For instance we have $NTPPI(y^0, y^1)$, $NTPPI(y^1, y^2)$, $NTPPI(y^2, y^3)$, and so on.

Other notions of UPQCNs, such as equivalence, consistency, or scenarios, are completely analogous to those of QCNs.

With respect to the satisfiability problem of a UPQCN, we have the following result:

Theorem 21 ([Balbiani and Condotta, 2002]) *Let $\mathcal{A} \in 2^{\mathsf{B}}$ be a subclass of relations of a qualitative constraint language over which not trivially inconsistent and $\diamond$-consistent* QCN*s are globally consistent. Then, checking the satisfiability of a* UPQCN *$\mathcal{U} = (V, C, t_{min})$ defined over $\mathcal{A}$ can be achieved in polynomial time.*

Due to Theorem 21 and Proposition 8 (at page 53), we can infer the following corollary:

Corollary 16 *Checking the satisfiability of a* UPQCN *$\mathcal{U} = (V, C, t_{min}, t_{max})$ defined over a distributive subclass of relations of Point Algebra, Cardinal Direction Calculus, Interval Algebra, or Block Algebra can be achieved in polynomial time.*

Regarding arbitrary UPQCNs, we have the following result:

Theorem 22 ([Balbiani and Condotta, 2002]) *Let $\mathcal{A} \in 2^{\mathsf{B}}$ be a subclass of relations of a qualitative constraint language over which not trivially inconsistent and $\diamond$-consistent* QCN*s are globally consistent. Then, given $\mathcal{A}$, checking the satisfiability of a* UPQCN *$\mathcal{U} = (V, C, t_{min})$ defined over 2^B is in* PSPACE*.*

Due to Theorem 22 and Proposition 8 (at page 53), we can obtain the following corollary:

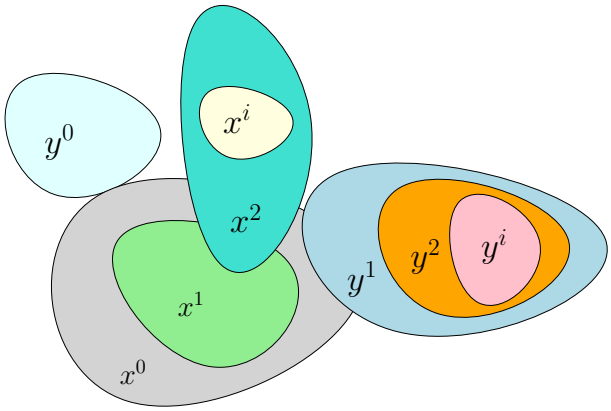


Figure 4.7: Solution of the UPQCN $\mathcal{U}$ of RCC-8 shown in Figure 4.5 (i>2)

Corollary 17 *Checking the satisfiability of a* UPQCN $\mathcal{U} = (V, C, t_{min}, t_{max})$ *is in* PSPACE *for the qualitative constraint languages of Point Algebra, Cardinal Direction Calculus, Interval Algebra, and Block Algebra.*

It is worth mentioning that the idea of ultimately periodic constraints has also been applied to simple temporal constraints found under instances of the Simple Temporal Problem (STP) [Dechter *et al.*, 1991]. In particular, Condotta et al. in [Condotta *et al.*, 2006c] extend the notion of an STP to that of an ultimately periodic STP (UPSTP) in a similar manner to what was described in this section. Interestingly, the study of temporal periodicy has also found applications in fields outside of artificial intelligence, such as software verification [Bensalem *et al.*, 2007]. In [Bensalem *et al.*, 2007], the authors study the problem of verifying that a cycle in the flow chart of a program does not terminate, i.e., it becomes ultimately periodic. In particular, they present some exact and sufficient conditions for cycle non-termination and provide an application for program verification, which allows for checking sequential and concurrent programs against temporal properties and using temporal logic as a guide to select test cases in such programs.

4.6 Conclusion

Qualitative spatio-temporal frameworks combine space and time in an interrelated manner and, hence, allow one to reason about time and space interdependently. Such qualitative spatio-temporal frameworks allow performing reasoning tasks that would otherwise be impossible through the use of simple QCNs (as they were presented in Section 3.2). For example, these frameworks allow us to represent the fact that a given region was partially overlapping another region at one point in time and became disconnected from that region at some next point of time.

In this chapter, we reviewed the state of the art qualitative spatio-temporal frameworks. In particular, we presented an hierarchy of qualitative spatio-temporal logics that builds on a combination of the propositional temporal logic (PTL) with certain spatial logics, we overviewed a spatio-temporal reasoning framework that considers temporal sequences of spatial QCNs and allows one to describe a spatial configuration that evolves and changes over time, and we also explored a qualitative constraint-based spatio-temporal formalism that results from combining the temporal language of Interval Algebra with the spatial one of RCC-8. Finally, we closed the

chapter with a discussion on the notions of periodicity and recurring patterns that can appear in temporalized QCN.

The discussion that took place in this chapter, will be relevant to the contributions that we will present in Chapter 6. In particular, the theoretical results that we obtain in that chapter are directly comparable to the theoretical results of this chapter. Further, we will explain in detail how our contributions there present a technical advance in the field of qualitative spatio-temporal reasoning.

Part II

Contributions

Chapter 5

Efficient Algorithms for tackling Qualitative Constraint Networks

5.1 Introduction

In this chapter, we present our contributions with respect to qualitative constraint-based spatial and temporal reasoning, which involve novel and efficient algorithms that go beyond the state of the art algorithms for reasoning with qualitative constraint networks (QCNs).

In particular, we define new local consistency conditions and new algorithms for enforcing those conditions, which we compare both theoretically and experimentally to the local consistency conditions and their respective algorithms that were presented in Section 3.5. Our contributions range over the entire spectrum of fundamental reasoning problems in qualitative constraint-based spatial and temporal reasoning. Specifically, we demonstrate both in theory and in practice how the *satisfiability problem*, the *minimal labeling problem*, and the *redundancy problem* of a given QCN (cf. Section 3.3) can be dealt with efficiently through the use of our novel techniques.

Furthermore, we address an issue in the literature regarding a non-sound approach that utilizes parallelism to check the satisfiability of RCC-8 networks. To this end, we provide the appropriate fixes for that approach, but also present our own approach of a simple decomposition scheme that exploits the sparse and loosely connected structure of the constraint graphs of very large real-world QCNs and paves the way for efficient utilization of parallelism to solve all the aforementioned fundamental reasoning tasks.

The contributions to be presented in this chapter draw from the published works in [Sioutis, 2014; Amaneddine *et al.*, 2013; Sioutis and Condotta, 2014b; Sioutis *et al.*, 2015i; Sioutis *et al.*, 2015h; Sioutis *et al.*, 2015f; Sioutis and Condotta, 2014c; Sioutis and Condotta, 2014a; Sioutis *et al.*, 2015g; Sioutis *et al.*, 2016b; Sioutis *et al.*, 2016a; Sioutis *et al.*, 2016c].

Finally, in Section 5.8 we conclude the chapter and give some directions for future work both in the field of qualitative constraint-based spatial and temporal reasoning and in the field of quantitative constraint-based spatial and temporal reasoning. In particular, with respect to the latter field, and inspired from our contribution in the former field that handles redundancy in a QCN [Sioutis *et al.*, 2015f], we have already made a step towards dealing with redundant information in the Simple Temporal Problem (STP) [Dechter *et al.*, 1991], presented in detail in [Lee *et al.*, 2016].

5.2 Partial Algebraic Closure and Partial $\diamond$-consistency

Given a QCN $\mathcal{N} = (V, C)$ and a graph $G = (V, E)$ the method of *partial algebraic closure* (also called *partial closure under weak composition*) removes certain base relations corresponding to the edges of graph G that are guaranteed to not participate in any solution of $\mathcal{N}$. The partial algebraic closure applies the following iterative operation until a fixed state is reached:

$$\forall\{v_i, v_k\}, \{v_i, v_j\}, \{v_j, v_k\} \in E, \quad C(v_i, v_j) \leftarrow C(v_i, v_j) \cap (C(v_i, v_k) \diamond C(v_k, v_j))$$

Clearly, when G is a complete graph, the method of partial algebraic closure becomes equivalent to the method of algebraic closure. It follows that the partial algebraic closure method is sound as it only removes base relations that do not participate in any solution of a given qualitative constraint network, but is not complete for deciding the satisfiability of any qualitative constraint network, i.e., we cannot conclude the satisfiability of an arbitrary qualitative constraint network if the partial algebraic closure method does not result in the assignment of the empty relation $\emptyset$ to a constraint of the network at hand. We will now present a local consistency, called partial $\diamond$-consistency, that is directly related to the partial algebraic closure method and results from restricting $\diamond$-consistency, as presented in Section 3.5.1, to a subset of the set of edges of the complete underlying graph of an input network.

Definition 36 *A* QCN $\mathcal{N} = (V, C)$ *is said to be* partially $\diamond$-consistent with respect to a graph $G = (V, E)$*, or, simply,* $\overset{\diamond}{}_G$-consistent*, if and only if we have that* $C(v_i, v_j) \subseteq C(v_i, v_k) \diamond C(v_k, v_j)$*,* $\forall\{v_i, v_k\}, \{v_i, v_j\}, \{v_j, v_k\} \in E$*.*

The $\overset{\diamond}{}_G$-consistent QCN obtained after the application of the partial algebraic closure method on a QCN $\mathcal{N}$ with respect to a given graph $G = (V, E)$ is equivalent to $\mathcal{N}$ and unique with respect to the particular choice of graph G. However, it should be noted that given two graphs $G_1 = (V, E_1)$ and $G_2 = (V, E_2)$ such that $E_1 \neq E_2$, the $\overset{\diamond}{}_{G_1}$-consistent and $\overset{\diamond}{}_{G_2}$-consistent QCNs of $\mathcal{N}$ may differ in some of their constraints. The $\overset{\diamond}{}_G$-consistent QCN of $\mathcal{N}$ is called the closure of $\mathcal{N}$ under $\overset{\diamond}{}_G$-consistency and it is denoted by $\overset{\diamond}{}_G(\mathcal{N})$. Network $\overset{\diamond}{}_G(\mathcal{N})$ corresponds to the largest (with respect to $\subseteq$) $\overset{\diamond}{}_G$-consistent sub-QCN of $\mathcal{N}$. Given two QCNs $\mathcal{N} = (V, C)$ and $\mathcal{N}' = (V, C')$, and an arbitrary graph $G = (V, E)$, we can prove the following properties with respect to $\overset{\diamond}{}_G$-consistency:

- $\diamond(\mathcal{N}) \subseteq \overset{\diamond}{}_G(\mathcal{N}) \subseteq \mathcal{N}$ (Dominance);
- $\overset{\diamond}{}_G(\mathcal{N})$ is equivalent to $\mathcal{N}$ (Equivalence);
- $\overset{\diamond}{}_G(\overset{\diamond}{}_G(\mathcal{N})) = \overset{\diamond}{}_G(\mathcal{N})$ (Idempotence);
- if $\mathcal{N}' \subseteq \mathcal{N}$ then $\overset{\diamond}{}_G(\mathcal{N}') \subseteq \overset{\diamond}{}_G(\mathcal{N})$ (Monotonicity).

The aforementioned properties follow directly from the fact that $\overset{\diamond}{}_G$-consistency is equivalent to $\diamond$-consistency when G is a complete graph. Next, we will show how $\overset{\diamond}{}_G$-consistency can be used to decide the satisfiability of QCNs under certain conditions. To this end, it will be necessary to introduce the notion of a *chordal (or triangulated) graph.*

Chordal graphs, triangulations, and perfect elimination orderings

We begin by introducing the definition of a chordal graph. The interested reader may find more results regarding chordal graphs, and graph theory in general, in [Diestel, 2012].

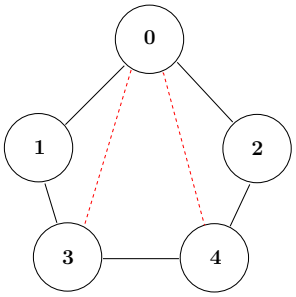


Figure 5.1: Example of a chordal graph

Algorithm 10: Triangulation(G)

in : A graph $G = (V, E)$.
output : A triangulated graph $G' = (V, E')$ with $E \subseteq E'$.
1 **begin**
2 $\alpha \leftarrow$ MaximumCardinalitySearch(G);
3 $F(\alpha) \leftarrow$ EliminationGame(G, α);
4 $G' = (V, E \bigcup F(\alpha))$;
5 **return** G';

Definition 37 *Let $G = (V, E)$ be an undirected graph. G is* chordal *(or* triangulated*) if every cycle of length greater than* 3 *has a chord, which is an edge connecting two non-adjacent vertices of the cycle.*

The graph shown in Figure 5.1 consists of a cycle which is formed by five solid edges and two dashed edges that correspond to its chords. As for this part, the graph is chordal. However, removing one dashed edge would result in a non-chordal graph. Indeed, the other dashed edge with three solid edges would form a cycle of length four with no chords. Chordality checking can be done in $O(|V| + |E|)$ time for a given graph $G = (V, E)$ with the *maximum cardinality search* (MCS) algorithm which also constructs an elimination ordering α as a byproduct [Tarjan and Yannakakis, 1984; Berry *et al.*, 2002]. In particular, MCS visits the vertices of a graph in an order such that, at any point, a vertex is visited that has the largest number of visited neighbours. If a graph is not chordal, it can be made so by the addition of a set of new edges, called *fill edges*. This process is called the *elimination game* and can run as fast as in $O(|V| + (|E \bigcup F(\alpha)|))$ time for a given graph $G = (V, E)$, where $F(\alpha)$ is the set of fill edges that result by following the elimination ordering α, eliminating the nodes one by one, and connecting all nodes in the neighbourhood of each eliminated node [Parter, 1961]. The obtained augmented graph $G' = (V, E \bigcup F(\alpha))$ is a *triangulation* of G [Fulkerson and Gross, 1965]. A triangulation algorithm that makes use of the aforementioned methods is presented in Algorithm 10, while the methods themselves are well presented in [Berry *et al.*, 2002].

Given a graph $G = (V, E)$, with $|V| = n$, and a vertex $v \in V$, $N(v)$ denotes the set of neighbours of v in G, i.e., $N(v) = \{u \mid \{u, v\} \in E\}$. A vertex $v \in V$ is said to be a *simplicial* vertex of G if the subgraph of G induced by $N(v)$ is complete. Further, let $\alpha : V \rightarrow \{0, 1, \ldots, n-1\}$ be a bijection of V onto $\{0, 1, \ldots, n-1\}$ serving as an elimination ordering of G, and let G_i denote the subgraph of G induced by $V_i = \{\alpha^{-1}(0), \alpha^{-1}(1), \ldots, \alpha^{-1}(i)\}$, with $0 \leq i < n$. (Note that $G_{n-1} = G$.) The elimination ordering $(\alpha^{-1}(n-1), \alpha^{-1}(n-2), \ldots, \alpha^{-1}(0))$ of the vertices of V is said to be a *perfect elimination ordering* of G, if for every $n > i > 0$, vertex $\alpha^{-1}(i)$ is a simplicial vertex of graph G_i. Then, we have the following theorem:

Theorem 23 ([Fulkerson and Gross, 1965]) *A graph G is chordal if and only if it admits a perfect elimination ordering.*

If the graph is already chordal, following an elimination ordering α produced by the maximum cardinality search algorithm means that no fill edges are added, i.e., α is actually a perfect elimination ordering [Tarjan and Yannakakis, 1984]. For example, a perfect elimination ordering for the chordal graph shown in Figure 5.1 would be the ordering $1 \rightarrow 3 \rightarrow 4 \rightarrow 2 \rightarrow 0$ of its set of nodes. In general, it is desirable to achieve chordality with as few fill edges as possible. However, obtaining an optimum graph triangulation is a NP-hard problem [Yannakakis, 1981].

In a QCN fill edges correspond to the universal relation B, i.e., the non-restrictive relation that contains all base relations and represents the lack of any definite knowledge between two entities. As such, the *chordal constraint graph* of a given QCN is exactly its constraint graph augmented with edges corresponding to relation B to make it chordal. In what follows, we will also use the term *chordal* QCN to emphasize that we only consider the constraints of the QCN that correspond to the edges of a triangulation of its constraint graph; this term will be particularly useful when visualizing QCNs.

After our brief introduction of chordal graphs and related notions, we demonstrate how chordal graphs become relevant in the context of qualitative spatial reasoning, by presenting a result that combines $\overset{\diamond}{}_G$-consistency and chordal graphs to decide the satisfiability of a qualitative constraint network. In particular, we prove the following result:

Proposition 19 *Let $\mathcal{N} = (V, C)$ be a* QCN *defined over a subclass of relations of a qualitative constraint language that has patchwork for not trivially inconsistent and $\diamond$-consistent* QCN*s defined over that subclass of relations, and $G = (V, E)$ a chordal graph such that $G(\mathcal{N}) \subseteq G$. If $\mathcal{N}$ is not trivially inconsistent and $\overset{\diamond}{}_G$-consistent, then $\mathcal{N}$ is satisfiable.*

Proof. Let $V = \{v_1, v_2, \ldots, v_n\}$ be the set of variables of $\mathcal{N}$. Since G is a chordal graph, by Theorem 23 we have that there exists a perfect elimination ordering of G. As such, let $\alpha : V \rightarrow \{0, 1, \ldots, n-1\}$ be a bijection of V onto $\{0, 1, \ldots, n-1\}$ serving as a perfect elimination ordering of G, i.e., α yields the perfect elimination ordering ($\alpha^{-1}(n-1)$,$\alpha^{-1}(n-2)$,...,$\alpha^{-1}(0)$) of G. We will show that $\mathcal{N}$ can be reduced to a sequence of completely overlapping, not trivially inconsistent, and $\diamond$-consistent QCNs. We remind the reader that G_i with $i \in \{0, \ldots, n-1\}$ denotes the subgraph of G induced by $\{\alpha^{-1}(0)$,$\alpha^{-1}(1)$,...,$\alpha^{-1}(i)\}$. Let S_i for some $i \in \{0, \ldots, n-1\}$ denote the subgraph of G_i induced by $\{\alpha^{-1}(i)\} \cup N_i(\alpha^{-1}(i))$, where $N_i(\alpha^{-1}(i))$ denotes the set of neighbours of $\alpha^{-1}(i)$ in G_i. Given a vertex $\alpha^{-1}(i)$ with $i \in \{1, \ldots, n-1\}$, we have that $\alpha^{-1}(i)$ is a simplicial vertex of graph G_i, as our perfect elimination ordering is ($\alpha^{-1}(n-1)$,$\alpha^{-1}(n-2)$,...,$\alpha^{-1}(0)$). As such, we have that subgraph S_i is complete for every $i \in \{0, \ldots, n-1\}$; specifically, $N_i(\alpha^{-1}(i))$ induces a complete subgraph of G_i, which when augmented with the vertex $\alpha^{-1}(i)$ (which shares edges with all of the vertices of $N_i(\alpha^{-1}(i))$) yields a complete graph. Further, due to our construction, and as G is chordal, every induced cycle of G (that is not a loop) lies in some graph S_i with $i \in \{0, \ldots, n-1\}$ and is a three-vertex cycle (triangle) by definition. In other words, any subset of the set of vertices of G cannot induce a cycle of G that is not a triangle of some graph S_i with $i \in \{0, \ldots, n-1\}$. In addition, as $\mathcal{N}$ is not trivially inconsistent and as $\overset{\diamond}{}_G$-consistency guarantees that all paths of length 2 in G are $\diamond$-consistent, we have that the sequence $(S_0, S_1, \ldots, S_{n-1})$ of complete graphs corresponds to the sequence $(\mathcal{N}_0 = (V(S_0), C_0), \mathcal{N}_1 = (V(S_1), C_1), \ldots, \mathcal{N}_{n-1} = (V(S_{n-1}), C_{n-1}))$ of completely overlapping, not trivially inconsistent, and $\diamond$-consistent QCNs. In particular, the QCNs are

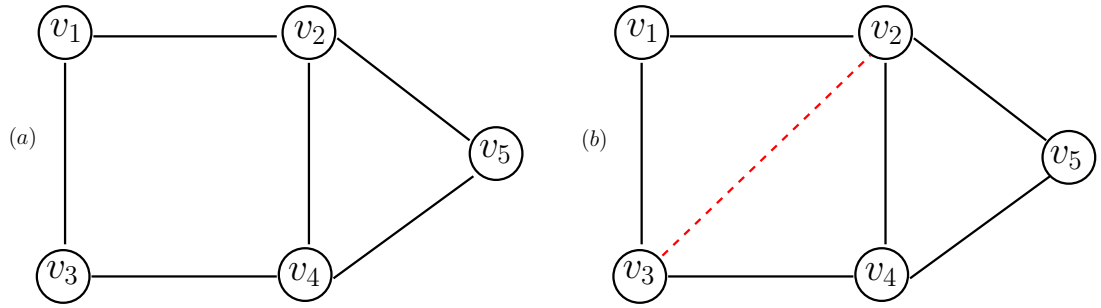


Figure 5.2: Triangulation of the underlying constraint graph of a QCN

completely overlapping in the following sense: each induced cycle of G is a triangle of the complete graph induced by $V(S_i)$ for some $i \in \{0, \ldots, n-1\}$, and for each j from 1 to $n-1$ we obtain a QCN $\mathcal{N}' = \bigcup_{j-1 \geq i \geq 0} \mathcal{N}_i$, such that $\mathcal{N}'[u,v] = \mathcal{N}_j[u,v]$ $\forall u, v \in (\bigcup_{j-1 \geq i \geq 0} V(S_i)) \cap V(S_j)$, and we have that $(\bigcup_{j-1 \geq i \geq 0} S_i) \cap S_j$ defines a complete graph (see also the notion of *pasting* for chordal graphs in [Diestel, 2012, Chapter 5]). As we have now satisfied the prerequisites for guaranteeing applicability of patchwork for our QCNs, we can derive that $\bigcup_{n-1 \geq i \geq 0} \mathcal{N}_i$ is a satisfiable QCN (by applying patchwork $n-1$ times). As $\mathcal{N} = \bigcup_{n-1 \geq i \geq 0} \mathcal{N}_i$, we can conclude that $\mathcal{N}$ is satisfiable. $\dashv$

An example of Proposition 19 is shown in Figure 5.2. The constraint graph of a given network $\mathcal{N}$ is shown in the left part of the figure (a). We triangulate this graph by adding edge v_2, v_3 that is depicted as a dashed line in the right part of the figure (b). A perfect elimination ordering for the obtained chordal graph would be the ordering $v_1 \rightarrow v_3 \rightarrow v_2 \rightarrow v_4 \rightarrow v_5$ of its set of nodes. By enforcing $\diamond$-consistency on the chordal constraint graph we can check the consistency of network $\mathcal{N}$ considering only 3 triangles of constraints, viz., (v_1, v_2, v_3), (v_2, v_3, v_4), and (v_2, v_4, v_5). If we had opted to complete the constraint graph and obtain the corresponding complete graph of order 5, enforcing $\diamond$-consistency would result in considering a total of 10 triangles of constraints. In general, the number of cycles of length 3, viz. triangles, in a complete graph G of order n, denoted by $c_3(G)$, is given by the mathematical expression $c_3(G) = n!/(6(n-3)!)$.

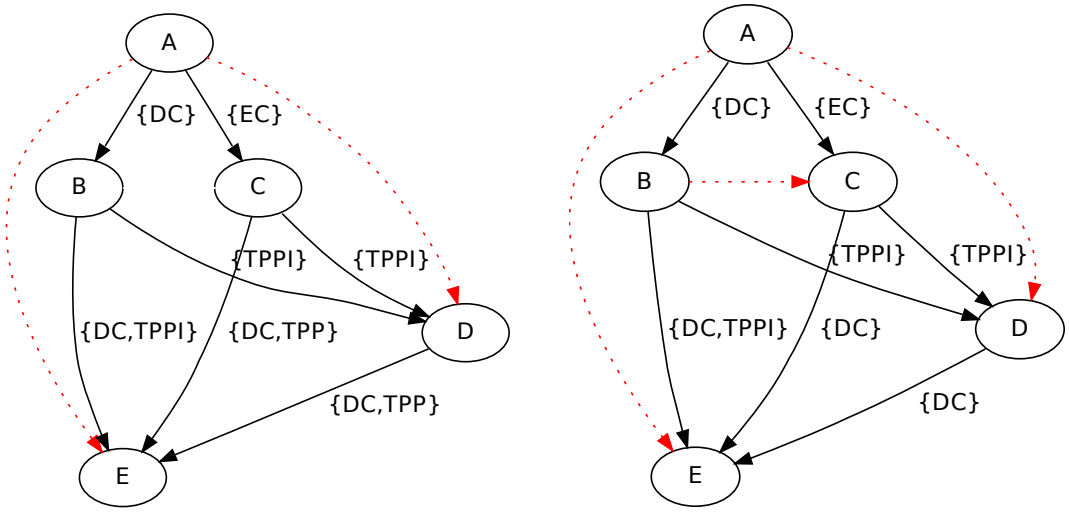
Due to Propositions 19 and 16 (at page 54), we can obtain the following result:

Proposition 20 *Let $\mathcal{N} = (V, C)$ be a QCN of Point Algebra, Cardinal Direction Calculus, Interval Algebra, Block Algebra, or RCC-8, defined over one of the classes $\mathcal{P}_{\mathsf{PA}}$, $\mathcal{P}_{\mathsf{CDC}}$, $\mathcal{H}_{\mathsf{IA}}$, $\mathcal{H}^n_{\mathsf{IA}}$, or $\hat{\mathcal{H}}_8$, $\mathcal{C}_8$, and $\mathcal{Q}_8$ respectively, and $G = (V, E)$ a chordal graph such that $G(\mathcal{N}) \subseteq G$. If $\mathcal{N}$ is not trivially inconsistent and $\diamond_G$-consistent, then $\mathcal{N}$ is satisfiable.*

The result of Proposition 19 can be directly compared with the result of Theorems 11 (at page 57) and 10 (at page 57), due to the following proposition that correlates tree decompositions and chordal graphs:

Proposition 21 ([Diestel, 2012]) *A graph G is chordal if and only if it has a tree decomposition $(T, \{X_1, \ldots, X_n\})$ where cluster X_i is a clique of G for every $i \in \{1, \ldots, n\}$.*

Proposition 21 suggests that given a QCN $\mathcal{N}$ where G is a triangulation of $G(\mathcal{N})$ and (T, X) a tree decomposition of G, $\diamond_G$-consistency and $\diamond_X$-consistency are equivalent. Of course, we note that tree decompositions in general do not necessarily yield chordal graphs by themselves, as the clusters of a tree decomposition may not correspond to cliques. However, as in the case of $\diamond_X$-consistency the clusters of a tree decomposition (T, X) are treated as cliques where $\diamond$-consistency is enforced on each and every one of them, every such tree decomposition corresponds



(a) chordal QCN (b) standard QCN

Figure 5.3: Pruning capacity of $\diamond$-consistency restricted to a triangulation of the constraint graph of a QCN (partial $\diamond$-consistency) and standard $\diamond$-consistency on that QCN

to a chordal graph. Thus, our result can be seen as a generalization of both the results of Theorems 11 and 10, since we also consider more general QCNs and not just atomic ones or QCNs restricted to Interval Algebra.

Pruning Capacity of Partial $\diamond$-consistency

Given Proposition 19 presented earlier and the property $\diamond(\mathcal{N}) \subseteq \overset{\diamond}{}_G(\mathcal{N}) \subseteq \mathcal{N}$ for a QCN $\mathcal{N}$, the question arises whether more pruning can indeed be obtained by enforcing $\diamond$-consistency on an input network rather than $\diamond$-consistency restricted to a triangulation of the constraint graph of that network (when considering the common edges of course). Further, if so, it would be interesting to pinpoint how much more pruning we can obtain, as local consistency conditions are in general used by satisfiability checking algorithms to efficiently propagate constraints during search and prune the search space (read the discussion in Section 3.5.2). In addition, it would be interesting to identify the case (if any) where $\diamond$-consistency and partial $\diamond$-consistency have the exact same pruning capacity.

First, we show that even for tractable subclasses of relations of a given qualitative constraint language, partial $\diamond$-consistency does not yield the same pruning capacity as $\diamond$-consistency in general.

In Figure 5.3 we present an example of the pruning capacity of enforcing $\diamond$-consistency restricted to a triangulation of the constraint graph of a QCN (partial $\diamond$-consistency) and standard $\diamond$-consistency on that QCN. For our example we use relations that are contained in all three maximal tractable subclasses $\hat{\mathcal{H}}_8, \mathcal{C}_8$, and $\mathcal{Q}_8$ of RCC-8. The (chordal) QCN in Figure 5.3a is partially $\diamond$-consistent with respect to its corresponding chordal constraint graph. Dashed edges represent universal relations. However, the addition of edge $\{B, C\}$ that completes the constraint graph, shown in Figure 5.3b, results in the pruning of base relation TPP on the constraints corresponding to edges $\{C, E\}$ and $\{D, E\}$ when enforcing partial $\diamond$-consistency with respect to that complete (and chordal) constraint graph of the QCN, i.e., when enforcing $\diamond$-consistency on that

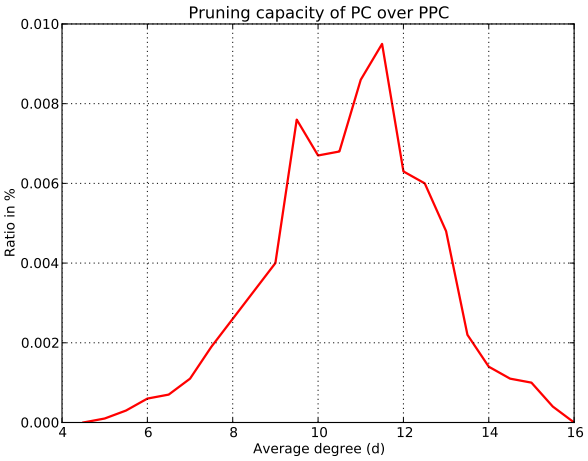


Figure 5.4: Pruning capacity of ⋄-consistency (PC) over partial ⋄-consistency (PPC)

QCN.

Thus, in the case of tractable QCNs we cannot have a result similar to the one provided by Bliek and Sam-Haroud who consider convex CSPs and show that the pruning capacity of enforcing path consistency restricted to a triangulation of the constraint graph of an input network is equivalent to the pruning capacity of enforcing path consistency on that network when considering the common edges [Bliek and Sam-Haroud, 1999].

We tried to find out approximately how much more pruning we can obtain on the common edges when opting to apply ⋄-consistency on an input network rather than ⋄-consistency restricted to a triangulation of the constraint graph of that network. Therefore, we considered networks of 100 nodes based on model $\mathsf{A}(\mathsf{n},\mathsf{d},\mathsf{l})$ [Renz and Nebel, 2001], with $l = 4$ $(= |\mathsf{B}|/2)$ relations per edge, and an average degree d between 4.5 and 16.0 with a 0.5 step. For each degree we created 10000 networks. Regarding model $\mathsf{A}(\mathsf{n},\mathsf{d},\mathsf{l})$, network instances with an average degree d between values 8.0 and 11.0 lie within the *phase transition* region, where it is equally possible for networks to be consistent or inconsistent, and, thus, are harder to solve [Renz and Nebel, 2001]. We expect more pruning to occur in that particular region for ⋄-consistency instead of partial ⋄-consistency, as more reasoning is required in general to decide the ⋄-consistency of a network. For each network we count the number of base relations that correspond to the common edges between a triangulation and the completion of its constraint graph, after ⋄-consistency has been applied restricted to each respective graph. Then, we calculate a ratio ρ, which indicates how many more base relations were pruned by ⋄-consistency in comparison with partial ⋄-consistency.

The results of our experiment are shown in Figure 5.4. Indeed, we were right on our expectation, as one can clearly see that the biggest ratios lie within the phase transition region, with the small exception of degree 11.5, that lies marginally outside 8.0 and 11.0.[12] In a sense, we obtain similar experimental results to those of Bliek and Sam-Haroud in [Bliek and Sam-Haroud, 1999], for a different kind of CSPs, viz., qualitative constraint networks over infinite domains. It is important to note that ⋄-consistency is able to prune more relations than partial ⋄-consistency, but only in a very small percentage. In the best case, the ratio is $\sim 0.009\%$, which translates to one more base relation being pruned in every 10 000. Of course, this ratio is particular to RCC-8, but qualitatively similar results can be obtained for other qualitative constraint languages as well.

[12] Some small variations in the graph are normal and expected artifacts of random instance generation.

Next, we present a result that is similar to the aforementioned main result of [Bliek and Sam-Haroud, 1999], namely, we will show that for a QCN $\mathcal{N}$ defined over a maximal distributive subclass of relations, the pruning capacity of $\overset{\diamond}{}_G$-consistency and $\diamond$-consistency is identical on the common edges of a chordal graph G with $G(\mathcal{N}) \subseteq G$ and the complete graph of $\mathcal{N}$.

First, we recall the following lemma that will allow us to make a chordal graph complete by adding a single edge at a time and keeping all intermediate graphs chordal:

Lemma 3 ([Bliek and Sam-Haroud, 1999]) *If $G = (V, E)$ is a non-complete chordal graph, then one can add a missing edge $\{u, v\}$ with $u, v \in V$ such that the graph $G' = (V, E \cup \{\{u, v\}\})$ is chordal and the graph induced by $X = \{x \mid \{u, x\}, \{x, v\} \in E\}$ is complete.*

We now proceed with proving the following result:

Proposition 22 *Let $\mathcal{N} = (V, C)$ be a* QCN *defined over a distributive subclass of relations of a qualitative constraint language that is a relation algebra and for which every $\diamond$-consistent atomic* QCN *is satisfiable, and $G = (V, E)$ a chordal graph such that $G(\mathcal{N}) \subseteq G$. Then, $\forall \{v, v'\} \in E$ we have that $\diamond(\mathcal{N})[v, v'] = \overset{\diamond}{}_G(\mathcal{N})[v, v']$.*

Proof. Suppose that $\mathcal{N}$ is $\overset{\diamond}{}_G$-consistent and not trivially inconsistent. Note that if $\mathcal{N}$ is trivially inconsistent, we will trivially have that $\diamond(\mathcal{N})[v, v'] = \overset{\diamond}{}_G(\mathcal{N})[v, v'] = \emptyset \ \forall \{v, v'\} \in E$, as desired, by the dominance property of $\overset{\diamond}{}_G$-consistency. We will add to graph G the missing edges one by one until it becomes a complete graph. We will show that the constraints that correspond to the new edges can be computed from existing constraints and without revising any existing constraints, so that for each intermediate chordal graph G' the network $\mathcal{N}$ will be $\overset{\diamond}{}_{G'}$-consistent, and for the final complete graph K_V (K_V denotes the complete graph on V) the network $\mathcal{N}$ will therefore be $\overset{\diamond}{}_{K_V}$-consistent, that is, $\diamond$-consistent. Every edge is added according to Lemma 3 to retain a chordal graph at all times. Consider graph G as shown in Figure 5.5.

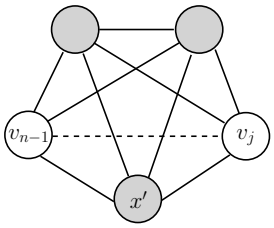


Figure 5.5: A non-complete chordal graph G

After adding edge $e = \{v_{n-1}, v_j\}$ to G we obtain graph $G' = (V, E \cup \{e\})$. As $e \notin E$ and $G(\mathcal{N}) \subseteq G$, initially $\mathcal{N}[v_{n-1}, v_j] = \mathsf{B}$. Let $X = \{x \mid \{v_{n-1}, x\}, \{x, v_j\} \in E\}$ be the set of variables that induce a complete graph, denoted in grey color in our figure. We will show that the constraint corresponding to edge e can be computed (in particular, refined from B) as follows.

$$\mathcal{N}[v_{n-1}, v_j] \leftarrow \bigcap_{x \in X} \mathcal{N}[v_{n-1}, x] \diamond \mathcal{N}[x, v_j] \tag{5.1}$$

Note that by construction graphs A induced by $\{v_{n-1}\} \cup X$ and B induced by $X \cup \{v_j\}$ are complete. Let us denote by $\mathcal{N}_A$ and $\mathcal{N}_B$ the QCNs of RCC-8 that correspond to graphs A and B respectively. We need to show that network $\mathcal{N}$ is $\overset{\diamond}{}_{G'}$-consistent. In particular, we need to show that every path π of length 2 that goes through v_{n-1} and v_j is consistent, as every other path of length 2 is by assumption consistent. We need to consider two cases (reasoning is the

same for symmetrical cases). With $x' \in X$, either $\pi = \langle v_{n-1}, v_j, x' \rangle$, or $\pi = \langle v_{n-1}, x', v_j \rangle$. If $\pi = \langle v_{n-1}, v_j, x' \rangle$, we prove that for every base relation of $\mathcal{N}[v_{n-1}, x']$, there exist base relations for both $\mathcal{N}[v_{n-1}, v_j]$ as defined in (5.1) and $\mathcal{N}[v_j, x']$, so that the path is consistent (i.e., $\mathcal{N}[v_{n-1}, x'] \subseteq \mathcal{N}[v_{n-1}, v_j] \diamond \mathcal{N}[v_j, x']$). As graph A is complete, $\mathcal{N}_A$ is $\diamond$-consistent, and therefore due to Theorems 2 (at page 35) and 9 (at page 54), $\mathcal{N}_A$ is weakly globally consistent and minimal. Thus, for every base relation of $\mathcal{N}[v_{n-1}, x']$ there exist compatible base relations for all other constraints in $\mathcal{N}_A$, i.e., there exists a scenario for $\mathcal{N}_A$, denoted by $\mathcal{N}_A^{\mathcal{S}}$. As $V(A) \cap V(B) = X$ induces a complete graph, the assignment of base relations for every constraint in $\mathcal{N}_A$ will define a partial scenario for $\mathcal{N}_B$ on X. Again, as graph B is complete, $\mathcal{N}_B$ is $\diamond$-consistent, and therefore due to Theorems 2 (at page 35) and 9 (at page 54), $\mathcal{N}_B$ is weakly globally consistent and minimal, thus, this partial scenario can be extented to a scenario for $\mathcal{N}_B$, denoted by $\mathcal{N}_B^{\mathcal{S}}$. As such, $\mathcal{N}[v_j, x']$ will be also characterized by a base relation. Finally, relations $\mathcal{N}[v_{n-1}, x]$ and $\mathcal{N}[x, v_j]$ for all $x \in X$ participating in (5.1) will be characterized by base relations, thus, there exists a base relation for $\mathcal{N}[v_{n-1}, v_j]$ too as clearly $\mathcal{N}_A^{\mathcal{S}} \cup \mathcal{N}_B^{\mathcal{S}}$ is $\overset{\diamond}{}_G$-consistent and, thus, satisfiable due to Proposition 19. If $\pi = \langle v_{n-1}, x', v_j \rangle$, by equation (5.1) we know that for every base relation of $\mathcal{N}[v_{n-1}, v_j]$ we can find base relations for $\mathcal{N}[v_{n-1}, x']$ and $\mathcal{N}[x', v_j]$, so that $\mathcal{N}[v_{n-1}, v_j] \subseteq \mathcal{N}[v_{n-1}, x'] \diamond \mathcal{N}[x', v_j]$ can hold. $\dashv$

Then, due to Propositions 1 (at page 21), 16 (at page 54), and 22 we can obtain the following corollary:

Corollary 18 *Let $\mathcal{N} = (V, C)$ be a* QCN *defined over a distributive subclass of relations of Point Algebra, Cardinal Direction Calculus, Interval Algebra, Block Algebra, or* RCC-*8, and $G = (V, E)$ a chordal graph such that $G(\mathcal{N}) \subseteq G$. Then, $\forall \{v, v'\} \in E$ we have that $\diamond(\mathcal{N})[v, v'] = \overset{\diamond}{}_G(\mathcal{N})[v, v']$.*

Regarding the minimal labeling problem, due to Theorem 2 (at page 35) and Proposition 22 we can assert the following result:

Theorem 24 *Let $\mathcal{N} = (V, C)$ be a* QCN *defined over a distributive subclass of relations of a qualitative constraint language that is a relation algebra and for which every $\diamond$-consistent atomic* QCN *is satisfiable, and $G = (V, E)$ a chordal graph such that $G(\mathcal{N}) \subseteq G$. If $\mathcal{N}$ is not trivially inconsistent and $\overset{\diamond}{}_G$-consistent, then $\forall \{v, v'\} \in E$ we have that $C(v, v')$ is minimal.*

Then, due to Theorem 24 and Propositions 1 (at page 21) and 3 (at page 34) we can obtain the following corollary:

Corollary 19 *Let $\mathcal{N} = (V, C)$ be a* QCN *defined over a distributive subclass of relations of Point Algebra, Cardinal Direction Calculus, Interval Algebra, Block Algebra, or* RCC-*8, and $G = (V, E)$ a chordal graph such that $G(\mathcal{N}) \subseteq G$. If $\mathcal{N}$ is not trivially inconsistent and $\overset{\diamond}{}_G$-consistent, then $\forall \{v, v'\} \in E$ we have that $C(v, v')$ is minimal.*

Next, we move on to study the implication of partial $\diamond$-consistency in the redundancy problem of a QCN. In particular, we will now prove a result that allows us to obtain the same set of non-redundant relations as the one provided by Lemma 2 (at page 36), more efficiently, through the use of chordal graphs and partial $\diamond$-consistency.

Proposition 23 *Let $\mathcal{N} = (V, C)$ be an all-different* QCN *defined over a distributive subclass of relations of a qualitative constraint language that is a relation algebra and for which every $\diamond$-consistent atomic* QCN *is satisfiable, and $G = (V, E)$ a chordal graph such that $G(\mathcal{N}) \subseteq G$.*

If $\mathcal{N}$ is not trivially inconsistent and $\overset{\diamond}{}_G$-consistent, then a relation $\diamond(\mathcal{N})[v,v']$ is non-redundant in $\diamond(\mathcal{N})$, iff we have that $\{v,v'\} \in E(G(\mathcal{N}))$ and $\mathcal{N}[v,v'] \neq \bigcap\{\mathcal{N}[v,v'']\diamond\mathcal{N}[v'',v'] \mid \{v,v''\},\{v'',v'\} \in E, v'' \neq v,v'\}$.

Proof. By Lemma 1 (at page 31) and the fact that for a QCN $\mathcal{M}$ we have that $\diamond(\mathcal{M})$ and $\mathcal{M}$ are equivalent, it trivially follows that if $\{v,v'\} \notin E(G(\mathcal{N}))$ then $\diamond(\mathcal{N})[v,v']$ is redundant in $\diamond(\mathcal{N})$. We will show that we can have the same set of non-redundant relations as the one suggested in Lemma 2 (at page 36). Let $\diamond(\mathcal{N})[v,v']$ with $\{v,v'\} \in E(G(\mathcal{N}))$ be a relation in $\diamond(\mathcal{N})$, then by Proposition 22 we have that $\diamond(\mathcal{N})[v,v'] = \overset{\diamond}{}_G(\mathcal{N})[v,v']$ $(= \mathcal{N}[v,v'])$. Let $G' = (V, E' = E \cup \{e\})$ be the chordal graph that results by adding edge $e = \{u,v'\}$ to G according to Lemma 3. We will only consider the general case where $|\{x \mid \{u,x\},\{x,v'\} \in E\}| \geq 2$, as in any other case the proof is trivial. It suffices to show that the next equation holds:

$$\bigcap\{\overset{\diamond}{}_{G'}(\mathcal{N})[v,v''] \diamond \overset{\diamond}{}_{G'}(\mathcal{N})[v'',v'] \mid \{v,v''\},\{v'',v'\} \in E', v'' \neq v,v'\} = \bigcap\{\overset{\diamond}{}_G(\mathcal{N})[v,v''] \diamond \overset{\diamond}{}_G(\mathcal{N})[v'',v'] \mid \{v,v''\},\{v'',v'\} \in E, v'' \neq v,v'\} \quad (5.2)$$

Equation (5.2) states that $\mathcal{N}[v,v']$ is strictly contained in the intersection of all paths of length 2 that start with v and end with v' in G iff $\mathcal{N}[v,v']$ is strictly contained in the intersection of all paths of length 2 that start with v and end with v' in G'. If (5.2) holds, it follows that we can add all edges necessary to obtain the paths of length 2 that start with v, end with v', and go through every $v'' \in V \setminus \{v,v'\}$, and have that $\bigcap\{\overset{\diamond}{}_G(\mathcal{N})[v,v''] \diamond \overset{\diamond}{}_G(\mathcal{N})[v'',v'] \mid \{v,v''\},\{v'',v'\} \in E, v'' \neq v,v'\} = \bigcap\{\diamond(\mathcal{N})[v,v''] \diamond \diamond(\mathcal{N})[v'',v'] \mid v'' \in V \setminus \{v,v'\}\}$, so that the necessary and sufficient conditions specified in Lemma 2 (at page 36) and Proposition 23 are equivalent with respect to deciding the redundancy of $\diamond(\mathcal{N})[v,v']$. By contradiction, let us assume that (5.2) does not hold. In particular, we will assume that b is a base relation that is in the intersection of all paths of length 2 that start with v and end with v' in G, but not in the intersection of all paths of length 2 that start with v and end with v' in G'. Clearly, this is only possible when $b \notin \mathcal{N}[v,u] \diamond \mathcal{N}[u,v']$, where the new relation $\mathcal{N}[u,v']$ is computed according to equation (5.1) in the proof of Proposition 22, i.e., $\mathcal{N}[u,v']$ is the intersection of all paths of length 2 that start with u and end with v' in G. Thus, a path $\pi = \langle u,x,v'\rangle$ exists in G such that $b \notin \mathcal{N}[v,u] \diamond \mathcal{N}[u,x] \diamond \mathcal{N}[x,v']$. Then, we have that $b \notin \mathcal{N}[v,x] \diamond \mathcal{N}[x,v']$. However, both $\{v,x\}$ and $\{x,v'\}$ are in G and we already know that $b \in \mathcal{N}[v,v''] \diamond \mathcal{N}[v'',v']$ for all $v'' \in V$ such that $\{v,v''\},\{v'',v'\} \in E$, thus, $b \in \mathcal{N}[v,x] \diamond \mathcal{N}[x,v']$. This is a contradiction, thus, (5.2) holds. ⊣

Then, due to Propositions 1 (at page 21), 3 (at page 34) and 23, we can obtain the following corollary:

Corollary 20 *Let $\mathcal{N} = (V,C)$ be an all-different* QCN *defined over a distributive subclass of relations of Point Algebra, Cardinal Direction Calculus, Interval Algebra, Block Algebra, or* RCC*-8, and $G = (V,E)$ a chordal graph such that $G(\mathcal{N}) \subseteq G$. If $\mathcal{N}$ is not trivially inconsistent and $\overset{\diamond}{}_G$-consistent, then a relation $\diamond(\mathcal{N})[v,v']$ is non-redundant in $\diamond(\mathcal{N})$, iff we have that $\{v,v'\} \in E(G(\mathcal{N}))$ and $\mathcal{N}[v,v'] \neq \bigcap\{\mathcal{N}[v,v''] \diamond \mathcal{N}[v'',v'] \mid \{v,v''\},\{v'',v'\} \in E, v'' \neq v,v'\}$.*

We now discuss some complexity results with respect to partial $\diamond$-consistency. Clearly, given a QCN $\mathcal{N} = (V,C)$ and a graph $G = (V,E)$, $\overset{\diamond}{}_G$-consistency for $\mathcal{N}$ can be decided or determined in $O(\delta|E|)$ time, where δ is the maximum vertex degree of G, as for each pair of variables $\{v_i,v_j\} \in E$ we can have at most δ variables $v_k \in V$ such that v_i,v_j,v_k forms a triangle in G. Applying or enforcing $\overset{\diamond}{}_G$-consistency on $\mathcal{N}$, i.e., making $\mathcal{N}$ $\overset{\diamond}{}_G$-consistent if it is not, requires the implementation of the partial algebraic closure method through an algorithm. As $\overset{\diamond}{}_G$-consistency is essentially $\diamond$-constistency restricted to a triangulation of the constraint graph of some network, the algorithms

Algorithm 11: PWC$(\mathcal{N}, G)$

```
   in     : A QCN N = (V, C) of a qualitative constraint language that is a relation algebra, and a
            graph G = (V, E).
   output : ◇^◇_G(N).
1  begin
2      Q ← {(v_i, v_j) | {v_i, v_j} ∈ E with 0 ≤ i ≤ j < |V|};
3      while Q ≠ ∅ do
4          (v_i, v_j) ← Q.pop();
5          foreach v_k ∈ V | {v_i, v_k}, {v_j, v_k} ∈ E do
6              t ← C(v_i, v_k) ∩ (C(v_i, v_j) ◇ C(v_j, v_k));
7              if t ≠ C(v_i, v_k) then
8                  C(v_i, v_k) ← t; C(v_k, v_i) ← t^-1;
9                  if i ≤ k then
10                     Q ← Q ∪ {(v_i, v_k)};
11                 else
12                     Q ← Q ∪ {(v_k, v_i)};
13             t ← C(v_k, v_j) ∩ (C(v_k, v_i) ◇ C(v_i, v_j));
14             if t ≠ C(v_k, v_j) then
15                 C(v_k, v_j) ← t; C(v_j, v_k) ← t^-1;
16                 if k ≤ j then
17                     Q ← Q ∪ {(v_k, v_j)};
18                 else
19                     Q ← Q ∪ {(v_j, v_k)};
20     return N;
```

that efficiently implement the partial algebraic closure are based on the algorithms for enforcing $\diamond$-consistency on a given QCN, such as the WC algorithm we presented in Section 3.5.1.

5.2.1 The PWC Algorithm

An algorithm that implements the partial algebraic closure method, and that is based on the WC algorithm we presented in Section 3.5.1, is provided in Algorithm 11, called PWC. Similarly to how constraints are stored in algorithm WC, a queue is assumed as the primary data structure for storing constraints to be processed. Given a QCN $\mathcal{N} = (V, C)$ and a graph $G = (V, E)$, the queue is initialized with all the pairs of variables in V which correspond to the edges of graph G (line 2). Like in algorithm WC, standard consistency operations take place (lines 6 and 13), and whenever a constraint is modified by a reduction of its set of base relations (lines 7 and 14), it is appropriately added back to the queue (lines 10, 12, 17, and 19). In fact, the only differences between algorithms WC and PWC are present in lines 2 and 5; these differences restrict the application of algorithm WC to constraints corresponding to edges of graph G. As such, and due to Proposition 6 (at page 37), we can assert the following proposition:

Proposition 24 *Given a* QCN $\mathcal{N}$ *of a qualitative constraint language that is a relation algebra, and a graph* $G = (V, E)$*, algorithm* PWC *terminates and returns* $\diamond^{\diamond}_{G}(\mathcal{N})$*.*

Given a QCN $\mathcal{N} = (V, C)$ and a graph $G = (V, E)$, algorithm PWC enforces $\diamond^{\diamond}_{G}$-consistency on $\mathcal{N}$ in $O(\delta|E||\mathsf{B}|)$ time, where δ is the maximum vertex degree of G, as for each pair of variables $\{v_i, v_j\} \in E$ we can have at most δ variables $v_k \in V$ such that v_i, v_j, v_k forms a triangle in G

and the constraint corresponding to each such pair of variables can be revised at most $|\mathsf{B}|$ times. The same variations of the WC algorithm that exist with respect to the selection of the next constraint to be processed, but also the choice of the data structure for storing the constraints to be processed, exist for the PWC algorithm too. Thus, the discussion on those variations which took place in Section 3.5.1 applies here as well. However, we need to place some focus on how a given graph G is stored within our algorithm, as the representation of graph G should allow obtaining triangles of variables (line 5 of the PWC algorithm) as efficiently as possible. A graph G is represented as a list, i.e., a sequence of index-value pairs where an index corresponds to some variable (vertex of the graph G) and its associated value corresponds to the set of its neighbours in the graph G. When a pair of variables $(v_i, v_j) \in V$ is processed, the intersection between the sets of neighbours of vertices v_i and v_j respectively is computed so as to obtain all triangles of variables the pair of variables (v_i, v_j) is a part of. In the average case, the time complexity of this operation is $\min(\{O(|N(v_i)|), O(|N(v_j)|)\})$ for any robust implementation. It should be clear that the greater the maximum vertex degree of graph G, the greater the time complexity of the previous operation will be. As such, the PWC algorithm is mostly suitable for QCNs with large and sparse constraint graphs. However, for smaller QCNs one could precompute and store all triangles of variables prior to the execution of the PWC algorithm. This works particularly well when dealing with very hard to solve problems, such as the minimal labeling problem, which we will discuss in the sequel.

5.2.2 The iPWC Algorithm

In this section, we present an algorithm that enforces partial $\diamond$-consistency incrementally and is able to reduce the number of consistency operations and constraint revisions that occur in the standard PWC algorithm. In particular, given a QCN $\mathcal{N} = (V, C)$ and some graph G, the new algorithm begins with a QCN that is defined on a single variable of $\mathcal{N}$, for example $\mathcal{N} \downarrow_{\{v\}}$ for some $v \in V$, and augments that QCN with a new variable of V each time, while maintaining $\overset{\diamond}{}_G$-consistency in the process, until it constructs the $\overset{\diamond}{}_G$-consistent QCN of $\mathcal{N}$ on V. The idea is very simple, yet quite powerful as we will see in the sequel. The term *"incremental"* has been used in the literature to describe the problem of maintaining or enforcing partial $\diamond$-consistency for a fixed-sized network where new constraints among existing variables are added or existing constraints are tightened. This approach is well described in the work of Gerevini [Gerevini, 2005] for qualitative temporal reasoning (where complete constraint graphs are considered and, thus, partial $\diamond$-consistency is identical to $\diamond$-consistency [Bliek and Sam-Haroud, 1999, chapt. 6]) and the work of Planken et al. for the Simple Temporal Problem (STP) [Planken *et al.*, 2010]. These approaches differ from the algorithm that we will present, in that we consider extensions of a given network with new spatial or temporal entities accompanied by new sets of constraints, and not just subsequent edge tightenings within a fixed-sized network. In other words, our algorithm can be considered as a vertex-incremental approach, whereas the aforementioned related approaches can be considered as being edge-incremental.

Before we present the algorithm, we introduce some notations. Given a QCN $\mathcal{N} = (V, C)$ and a graph $G = (V, E)$, and a QCN $\mathcal{N}' = (V', C')$ and a graph $G' = (V', E')$ such that $V' \setminus V = \{v\}$ and $\forall\{u, u'\} \in E'$ we have that $u = v$ or $u' = v$, $\mathcal{N} \uplus \mathcal{N}'$ yields the QCN $\mathcal{N}'' = (V'', C'')$, where $V'' = V \cup V'$, $C''(v, v') = C''(v', v) = \mathsf{B}$ for all $(v, v') \in (V \setminus V') \times (V' \setminus V)$, $C''(v, v') = C(v, v')$ for all $(v, v') \in (V \times V)$, and $C''(v, v') = C'(v, v')$ and $C''(v', v) = C'(v', v)$ for all $(v, v') \in \{v\} \times V'$. The $\uplus$ operation on QCNs is presented in Figure 5.6.

Next, we introduce a function that receives a $\overset{\diamond}{}_G$-consistent QCN $\mathcal{N} = (V, C)$ with respect to a graph $G = (V, E)$, and a QCN $\mathcal{N}' = (V', C')$ along with a graph $G' = (V', E')$ such that

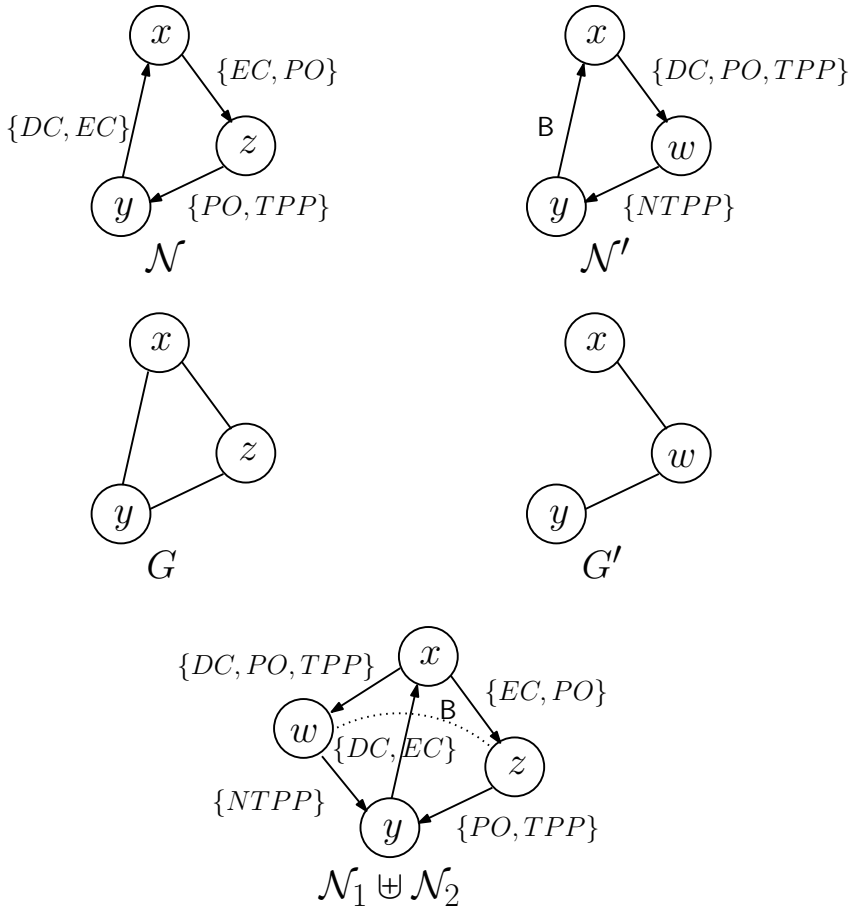


Figure 5.6: The $\uplus$ operation on two QCNs accompanied by their respective graphs

$V' \setminus V = \{v\}$ and $u = v$ or $u' = v$ $\forall \{u, u'\} \in E'$, and returns the $^{\diamond}_{G''}$-consistent QCN $\mathcal{N} \uplus \mathcal{N}'$, where $G'' = (V \cup V', E \cup E')$. The function is called AugmentQCN, presented in Algorithm 12. Regarding function AugmentQCN, we prove the following lemma:

Lemma 4 *Given a graph $G = (V, E)$, a graph $G' = (V', E')$ such that $V' \setminus V = \{v\}$ and $u = v$ or $u' = v$ $\forall \{u, u'\} \in E'$, and a $^{\diamond}_{G}$-consistent QCN $\mathcal{N} = (V, C)$ and a QCN $\mathcal{N}' = (V', C')$ of a qualitative constraint language that is a relation algebra, function* AugmentQCN *terminates and returns $^{\diamond}_{G''}(\mathcal{N} \uplus \mathcal{N}')$, where $G'' = (V \cup V', E \cup E')$.*

Proof. Due to Proposition 24 (at page 97), after AugmentQCN has terminated, $\mathcal{N}'$ will be $^{\diamond}_{G'}$-consistent, as function AugmentQCN is equivalent to a variant of algorithm PWC that considers a set of edges $E \cup E'$ (line 6), which is a superset of the set of edges E that would otherwise be considered by the standard algorithm PWC when given $\mathcal{N}'$ and G' as its input. Let (i, j, k) be a triangle of variables that is introduced in $\mathcal{N} \uplus \mathcal{N}'$ due to the augmentation of $\mathcal{N}$ with $\mathcal{N}'$. In particular, let $\{i, k\}, \{j, k\} \in E'$, and $\{i, j\} \in E$, and $\{k\} = V' \setminus V$. Since $\mathcal{N}$ is by assumption $^{\diamond}_{G}$-consistent, it suffices to show that function AugmentQCN can correctly enforce $\diamond$-consistency on those triangles of variables; any revised constraint corresponding to a pair of variables in the $\diamond$-consistent triangles of variables will be added to the queue for revision and all possibly affected paths of length 2 in G'' will be treated by function AugmentQCN exactly as they would be treated by algorithm PWC, hence, $^{\diamond}_{G''}$-consistency on $\mathcal{N} \uplus \mathcal{N}'$ will be correctly established.

Algorithm 12: AugmentQCN($\mathcal{N}$, $\mathcal{N}'$, G, G')

in : A graph $G = (V, E)$, a graph $G' = (V', E')$ such that $V' \setminus V = \{v\}$ and $u = v$ or $u' = v$ $\forall \{u, u'\} \in E'$, and a $\diamond_G$-consistent QCN $\mathcal{N} = (V, C)$ and a QCN $\mathcal{N}' = (V', C')$ of a qualitative constraint language that is a relation algebra.

output : $\diamond_{G''}(\mathcal{N} \uplus \mathcal{N}')$, where $G'' = (V \cup V', E \cup E')$.

```
1  begin
2      N'' = (V'', C'') ← N ⊎ N';
3      Q ← {(v_i, v_j) | {v_i, v_j} ∈ E' with 0 ≤ i ≤ j < |V|};
4      while Q ≠ ∅ do
5          (v_i, v_j) ← Q.pop();
6          foreach v_k ∈ V | {v_i, v_k}, {v_j, v_k} ∈ E ∪ E' do
7              t ← C''(v_i, v_k) ∩ (C''(v_i, v_j) ⋄ C''(v_j, v_k));
8              if t ≠ C''(v_i, v_k) then
9                  C''(v_i, v_k) ← t; C''(v_k, v_i) ← t^{-1};
10                 if i ≤ k then
11                     Q ← Q ∪ {(v_i, v_k)};
12                 else
13                     Q ← Q ∪ {(v_k, v_i)};
14             t ← C''(v_k, v_j) ∩ (C''(v_k, v_i) ⋄ C''(v_i, v_j));
15             if t ≠ C''(v_k, v_j) then
16                 C''(v_k, v_j) ← t; C''(v_j, v_k) ← t^{-1};
17                 if k ≤ j then
18                     Q ← Q ∪ {(v_k, v_j)};
19                 else
20                     Q ← Q ∪ {(v_j, v_k)};
21     return N'';
```

Regarding the triangle of variables (i, j, k), and assuming without loss of generality that $k < i, j$, function AugmentQCN will initially include edges (k, i) and (k, j) in its queue. With respect to edge (k, i), function AugmentQCN will perform the following consistency checks:

$$(\mathcal{N} \uplus \mathcal{N}')[k, j] \neq (\mathcal{N} \uplus \mathcal{N}')[k, j] \cap ((\mathcal{N} \uplus \mathcal{N}')[k, i] \diamond (\mathcal{N} \uplus \mathcal{N}')[i, j])$$

$$(\mathcal{N} \uplus \mathcal{N}')[j, i] \neq (\mathcal{N} \uplus \mathcal{N}')[j, i] \cap ((\mathcal{N} \uplus \mathcal{N}')[j, k] \diamond (\mathcal{N} \uplus \mathcal{N}')[k, i])$$

With respect to edge (k, j), function AugmentQCN will perform the following consistency operations:

$$(\mathcal{N} \uplus \mathcal{N}')[k, i] \neq (\mathcal{N} \uplus \mathcal{N}')[k, i] \cap ((\mathcal{N} \uplus \mathcal{N}')[k, j] \diamond (\mathcal{N} \uplus \mathcal{N}')[j, i])$$

$$(\mathcal{N} \uplus \mathcal{N}')[i, j] \neq (\mathcal{N} \uplus \mathcal{N}')[i, j] \cap ((\mathcal{N} \uplus \mathcal{N}')[i, k] \diamond (\mathcal{N} \uplus \mathcal{N}')[k, j])$$

We notice that out of all possible paths of length 2 in the the triangle of variables (i, j, k), we miss the ones corresponding to the following consistency operations:

$$(\mathcal{N} \uplus \mathcal{N}')[j, k] \neq (\mathcal{N} \uplus \mathcal{N}')[j, k] \cap ((\mathcal{N} \uplus \mathcal{N}')[j, i] \diamond (\mathcal{N} \uplus \mathcal{N}')[i, k])$$

$$(\mathcal{N} \uplus \mathcal{N}')[i, k] \neq (\mathcal{N} \uplus \mathcal{N}')[i, k] \cap ((\mathcal{N} \uplus \mathcal{N}')[i, j] \diamond (\mathcal{N} \uplus \mathcal{N}')[j, k])$$

However, as our qualitative constraint language is a relation algebra, due to $^{-1}$-involution and $^{-1}$-involutive distributivity we have that:

$$(\mathcal{N} \uplus \mathcal{N}')[j, k] \neq (\mathcal{N} \uplus \mathcal{N}')[j, k] \cap ((\mathcal{N} \uplus \mathcal{N}')[j, i] \diamond (\mathcal{N} \uplus \mathcal{N}')[i, k]) \text{ iff}$$

Algorithm 13: iPWC($\mathcal{N}$, G)

in : A QCN $\mathcal{N} = (V, C)$ of a qualitative constraint language that is a relation algebra, and a graph $G = (V, E)$.
output : $\overset{\diamond}{}_G(\mathcal{N})$.

```
1  begin
2      Vars ← ∅;
3      G_Vars ← (Vars, ∅);
4      N' ← (∅, Null);
5      foreach v ∈ V do
6          Vars ← Vars ∪ {v};
7          V' ← {v} ∪ {u ∈ Vars | {v, u} ∈ E};
8          G' ← (V', {{v, u} | {v, u} ∈ E});
9          N' ← AugmentQCN(N', N↓V', G_Vars, G');
10         G_Vars ← (Vars, E(G_Vars) ∪ E(G'));
11     N ← N';
12     return N;
```

$$((\mathcal{N} \uplus \mathcal{N}')[j,k])^{-1} \neq ((\mathcal{N} \uplus \mathcal{N}')[j,k])^{-1} \cap (((\mathcal{N} \uplus \mathcal{N}')[j,i] \diamond (\mathcal{N} \uplus \mathcal{N}')[i,k]))^{-1} \text{ iff}$$

$$(\mathcal{N} \uplus \mathcal{N}')[k,j] \neq (\mathcal{N} \uplus \mathcal{N}')[k,j] \cap (((\mathcal{N} \uplus \mathcal{N}')[i,k])^{-1} \diamond ((\mathcal{N} \uplus \mathcal{N}')[j,i])^{-1}) \text{ iff}$$

$$(\mathcal{N} \uplus \mathcal{N}')[k,j] \neq (\mathcal{N} \uplus \mathcal{N}')[k,j] \cap ((\mathcal{N} \uplus \mathcal{N}')[k,i] \diamond (\mathcal{N} \uplus \mathcal{N}')[j,i])$$

Similarly, we have that:

$$(\mathcal{N} \uplus \mathcal{N}')[i,k] \neq (\mathcal{N} \uplus \mathcal{N}')[i,k] \cap ((\mathcal{N} \uplus \mathcal{N}')[i,j] \diamond (\mathcal{N} \uplus \mathcal{N}')[j,k]) \text{ iff}$$

$$((\mathcal{N} \uplus \mathcal{N}')[i,k])^{-1} \neq ((\mathcal{N} \uplus \mathcal{N}')[i,k])^{-1} \cap (((\mathcal{N} \uplus \mathcal{N}')[i,j] \diamond (\mathcal{N} \uplus \mathcal{N}')[j,k]))^{-1} \text{ iff}$$

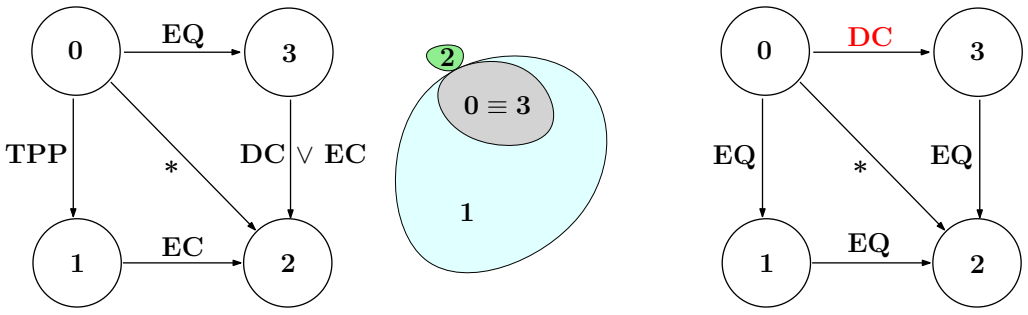
$$(\mathcal{N} \uplus \mathcal{N}')[k,i] \neq (\mathcal{N} \uplus \mathcal{N}')[k,i] \cap (((\mathcal{N} \uplus \mathcal{N}')[j,k])^{-1} \diamond ((\mathcal{N} \uplus \mathcal{N}')[i,j])^{-1}) \text{ iff}$$

$$(\mathcal{N} \uplus \mathcal{N}')[k,i] \neq (\mathcal{N} \uplus \mathcal{N}')[k,i] \cap ((\mathcal{N} \uplus \mathcal{N}')[k,j] \diamond (\mathcal{N} \uplus \mathcal{N}')[i,j])$$

Thus, the consistency operations performed by function AugmentQCN are already sufficient for enforcing $\diamond$-consistency in the triangle of variables (i, j, k). $\dashv$

Next we present our algorithm that enforces partial $\diamond$-consistency incrementally and is able to reduce the number of consistency operations and constraint revisions that occur in the standard PWC algorithm, as demonstrated in the proof of Lemma 4, but as we will also demonstrate with a running example and an evaluation in the sequel. Our algorithm is presented in Algorithm 13 and is called iPWC. Given a QCN $\mathcal{N} = (V, C)$ and some graph G, algorithm iPWC essentially begins with a QCN that is defined on a single variable of $\mathcal{N}$ (and that is by default and by definition $\diamond$-consistent), for example $\mathcal{N} \downarrow_{\{v\}}$ for some $v \in V$, and augments that QCN with a new variable of V each time, while maintaining $\overset{\diamond}{}_G$-consistency in the process, until it constructs the $\overset{\diamond}{}_G$-consistent QCN of $\mathcal{N}$ on V. Regarding algorithm iPWC, and due to Lemma 4, we can assert the following proposition:

Proposition 25 *Given a* QCN $\mathcal{N}$ *of a qualitative constraint language that is a relation algebra, and a graph* $G = (V, E)$*, algorithm* iPWC *terminates and returns* $\overset{\diamond}{}_G(\mathcal{N})$.



(a) consistent QCN along with a solution (b) inconsistent QCN

Figure 5.7: QCNs with respect to their constraint graphs

Starting with a single-entity QCN $\mathcal{N}_1$ and extending it with a new QCN $\mathcal{N}_i$ of one new entity at a time applying AugmentQCN a total of $n-1$ times for a given set of n entities (thus, $2 \leq i \leq n$), it follows that we will perform $O(\delta_2|E_2||\mathsf{B}| + \ldots + \delta_n|E_n||\mathsf{B}|)$ intersection and composition operations on average for constructing a partially $\diamond$-consistent QCN of n temporal or spatial entities, where δ_i is the maximum vertex degree of graph $((G_1 \cup G_2) \cup \ldots) \cup G_i$ and $O(|E_i|)$ is the number of constraints that the new QCN $\mathcal{N}_i$ contributes to QCN $((\mathcal{N}_1 \uplus \mathcal{N}_2) \uplus \ldots) \uplus \mathcal{N}_i$. As $\delta_2 \leq \ldots \leq \delta_n$ and $|E_2| \cup \ldots \cup |E_n| = |E|$, it follows that the complexity of iPWC is asymptotically upper bounded by $O(\delta_n|E||\mathsf{B}|)$, which is the complexity of PWC. Thus, we increase on average the performance of applying partial $\diamond$-consistency with respect to algorithm PWC, and retain the same worst-case complexity as algorithm PWC.

Running Example

Before moving on to our running example to better understand how algorithm iPWC and its core function AugmentQCN operate, it is important to explain the notions of *processed edges* and *constraint checks*. Given a QCN $\mathcal{N} = (V, C)$ and $v_i, v_k, v_j \in V$, and a graph $G = (V, E)$, an edge $e \in E$ is processed whenever it is popped out of the queue (line 5 in function AugmentQCN), and a constraint check is performed whenever we compute relation $r = C(v_i, v_j) \cap (C(v_i, v_k) \diamond C(v_k, v_j))$ and check if $r \subset C(v_i, v_j)$, so that we can propagate its constrainedness (lines 7–8 and 14–15 in function AugmentQCN). In our running example we demonstrate how algorithm iPWC is able to perform better than the one-shot partial $\diamond$-consistency algorithm PWC presented earlier. In what follows, we always give the graph $G \cup G'$ of QCN $\mathcal{N} \uplus \mathcal{N}'$ as input to AugmentQCN to facilitate description, as the initial network $\mathcal{N}$ along with graph G, and its augmentation $\mathcal{N}'$ along with graph G', are easily identifiable at each step by viewing the figures and following our line of reasoning.

Case of a satisfiable QCN. [13] Let us consider the satisfiable QCN of RCC-8 shown in Figure 5.7a. We will first build a partially $\diamond$-consistent version of this network incrementally, using function AugmentQCN repeatedly and exactly as it would be used by algorithm iPWC. We begin with node 0 and add nodes 1, 2, and 3 one at each step. We always pop edges from the left of the queue and push them to the right of it (i.e., in a right to left FIFO manner), and whenever needed we use the converse relation corresponding to the constraint of an edge. First, $(\{0\}, \emptyset)$ is given as input to AugmentQCN with no edges whatsoever, the queue is initialized with an empty set

[13] To be exact, we consider a satisfiable QCN whose satisfiability can be detected by partial $\diamond$-consistency.

of edges, no edges are processed, and, thus, no constraint checks occur. Then, $(\{0,1\},\{\{0,1\}\})$ is given as input to AugmentQCN, the queue is initialized with the set of edges $\{(0,1)\}$, a single edge is processed, and no constraint checks occur as there are no triangles in the network. Then, $(\{0,1,2\}, \{\{0,1\}, \{0,2\}, \{1,2\}\})$ is given as input to AugmentQCN, the queue is initialized with the set of edges $\{(1,2)\}$, viz., the constraint edges that accompany the newly inserted entity 2. Edge $(0,2)$ is not included in the queue as it corresponds to the universal relation B and we do not consider it at all during initialization (this detail is not provided in algorithm iPWC). (The weak composition between B and any other relation always yields the latter relation.) Edge $(1,2)$ is popped out of the queue. Two constraint checks take place among edges $\{0,1\}$, $\{0,2\}$, and $\{1,2\}$, leading to the pruning of the universal relation B corresponding to edge $(0,2)$ into the relation $DC \vee EC$, and edge $(0,2)$ is inserted in the queue which now holds the set of edges $\{(0,2)\}$. Edge $(0,2)$ is popped out of the queue. Two constraint checks take place among edges $\{0,1\}$, $\{0,2\}$, and $\{1,2\}$, leading to no pruning of the relations corresponding to edges $\{0,1\}$ and $\{1,2\}$. Finally, $(\{0,1,2,3\}, \{\{0,1\}, \{0,2\}, \{0,3\}, \{1,2\}, \{2,3\}\})$ is given as input to AugmentQCN, the queue is initialized with the set of edges $\{(0,3),(2,3)\}$. Both edges are popped out of the queue, each one leading to two constraint checks, with no further pruning of relations. In total we have processed 5 edges and performed 8 constraint checks.

We now proceed with algorithm PWC. First, $(\{0,1,2,3\}, \{\{0,1\}, \{0,2\}, \{0,3\}, \{1,2\}, \{2,3\}\})$ is given as input to PWC. The queue is initialized with the set of edges $\{(0,1), (0,3), (1,2), (2,3)\}$. Edge $(0,1)$ is popped out of the queue. Two constraint checks take place among edges $(0,1)$,$(0,2)$, and $(1,2)$, leading to the pruning of the universal relation B corresponding to edge $(0,2)$ into the relation $DC \vee EC$. Edge $(0,2)$ is inserted in the queue which now holds the set of edges $\{(0,3), (1,2), (2,3), (0,2)\}$. All edges are popped out of the queue with no further pruning of relations. Edges $(0,3)$, $(1,2)$, and $(2,3)$ lead to two constraint checks each, and edge $(0,2)$ to four, as it is part of two triangles. In total we have processed 5 edges and performed 12 constraint checks. PWC proccesses the same number of edges as iPWC, but performs 4 more constraint checks. The numbers may vary a bit depending on the order of the edges in the initialized queue (in our running example we have considered a sorted initial queue of edges), but the trend is that for satisfiable QCNs, iPWC will perform less constraint checks than PWC, and will process only slightly more edges than PWC, depending on whether an edge already exists in the queue or not. (As PWC works with a large initial queue, an edge might not have to be pushed/popped as often as it may already exist in the queue.)

Case of an unsatisfiable QCN. [14] Let us consider now the unsatisfiable QCN of RCC-8 shown in Figure 5.7b. Regions 0, 1, 2, and 3 are all equal to each other (they are essentially the one same region), thus, regions 0 and 3 cannot be disconnected. We will not go into detail as we did with the case of the satisfiable QCN, by now the reader should be able to verify (assuming a sorted initial queue of edges in every case) that iPWC processes in total 4 edges and performes 5 constraint checks, while PWC processes in total 2 edges and performes 3 constraint checks. In fact, Figure 5.7b describes the worst case scenario for iPWC+, i.e., an inconsistency that occurs when the last temporal or spatial entity is added to the network. By that point iPWC will already have fully reasoned with all previous entities and their accompanying constraints. On the other hand, PWC will always do a first iteration of the queue, and might be able to immediately capture the inconsistency, as with our running example. Again, the numbers may vary a bit depending on the order of the edges in the initialized queue, but the trend is that for unsatisfiable QCNs, iPWC will perform more constraint checks than PWC, and will process more

[14] To be exact, we consider an unsatisfiable QCN whose unsatisfiability can be detected by partial ⋄-consistency.

edges than PWC too.

Concluding our running example, we have observed that iPWC should perform better than PWC in the case of satisfiable QCNs and worse than PWC in the case of unsatisfiable QCNs. Further, iPWC works with a very small queue at each step. It is natural that a $\diamond$-consistency enforcing algorithm will run faster when there is an inconsistency in the input network, as the inconsistency will potentially not allow the algorithm to reason exhaustively with the network relations. Thus, we expect that the overall performance of iPWC should be better than that of PWC in the average case. We are about to experimentally verify this in the next section.

Experimental evaluation

We evaluate the performance of an implementation of the iPWC algorithm against an implementation of the PWC algorithm for enforcing partial $\diamond$-consistency on triangulations of constraint graphs of QCNs. Note that by definition partial $\diamond$-consistency does not involve chordal graphs, but as chordal graphs play an important role in deciding the satisfiability of QCNs (see Proposition 19) it is why we consider triangulations of constraint graphs of QCNs.[15]

Technical Specifications The experimentation was carried out on a computer with an Intel Core 2 Duo P7350 processor with a 2.00 GHz frequency per CPU core, 4 GB RAM, and the Lucid Lynx x86_64 OS (Ubuntu Linux). The implementations of iPWC and PWC were run with the CPython interpreter that implements Python 2.6 (`http://www.python.org/`). Only one of the CPU cores was used for the experiments. The implementations of iPWC and PWC can be found online in the following address: `http://www.cril.fr/~sioutis/work.php`.

Dataset and Measures We considered random datasets consisting of large Interval Algebra and RCC-8 networks generated by the BA(n, m) model [Barabasi and Albert, 1999], which we introduce here, and the standard A(n, d, l) model [Renz and Nebel, 2001] used extensively in literature.

The BA(n, m) model creates random scale-free graphs of order n and a *preferential attachment* value m. We provide the following simple definition of a scale-free graph and elaborate on the details:

Definition 38 *Graphs with a power law tail in their node degree distribution are called* scale-free graphs.

The degree of a node in a graph is the number of connections it has to other nodes (i.e., the number of its neighbours) and the degree distribution $P(k)$ is the probability distribution of these degrees over the whole graph, i.e, $P(k)$ is defined to be the fraction of nodes in the network with degree k. Thus, if there is a total number of n nodes in a graph and n_k of them have degree k, we have that $P(k) = n_k/n$. For scale-free graphs the degree distribution $P(k)$ follows a power law which can be expressed mathematically as $P(k) \simeq k^{-\gamma}$, where $2 < \gamma < 3$, although γ can lie marginally outside these bounds [Choromański *et al.*, 2013].

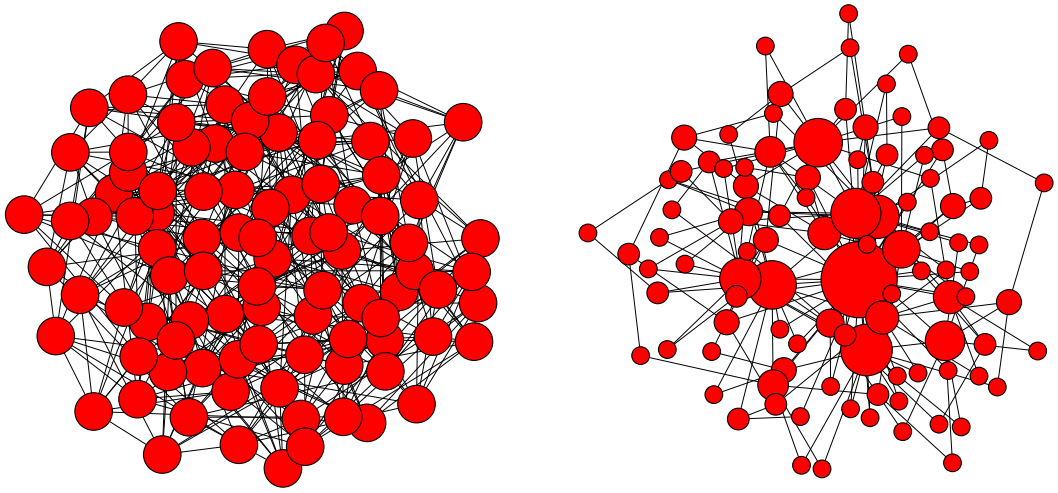
There are several models to create random scale-free graphs that rely on *growth* and *preferential attachment* [Bollobás, 2003]. Growth denotes the increase in the number of nodes in the graph over time. Preferential attachment refers to the fact that new nodes tend to connect to

[15] In the sequel, we will see that in the presense of the patchwork property alone and the absence of any other property, chordal graphs pose a minimal requirement for deciding the satisfiability of a QCN.

Algorithm 14: BA-Graph(n, m)

```
in      : The number of nodes n, and the number of edges to attach from a new node to existing
          nodes m.
output  : A scale-free graph G.
1  begin
2      if m < 1 or m ≥ n then
3          raise RuntimeError, "Invalid operation";
4      V ← {0, ..., m − 1}; E ← ∅;
5      targets ← V;
6      repeatedNodes ← list(∅);
7      source ← m;
8      while source < n do
9          V ← V ⋃{source}; E ← E ⋃{(source, i) | i ∈ targets };
10         foreach i ∈ targets do
11             repeatedNodes.append(i);
12             repeatedNodes.append(source);
13         targets ← ∅;
14         while |targets| < m do
15             x ← random.choice(repeatedNodes);
16             targets.add(x);
17         source ← source + 1;
18     return G = (V, E);
```

existing nodes of large degree, which means that the more connected a node is, the more likely it is to receive new links. Such higher degree nodes appear in descriptions of real-world networks and have stronger ability to grab links added to the network [Barabasi and Albert, 1999]. For example, if we were to describe the topological relations in Greece, Greece would be our major hub that would relate topologically to all of its regions and cities, followed by smaller hubs that would capture topological relations within the premises of a city or a neighborhood. It would not really make sense to specify that the porch of a house is located inside Greece, when it is already encoded that the house is located inside a city of Greece. Such natural and human-made systems are most often described by scale-free graphs [Barabasi and Bonabeau, 2003; Hein *et al.*, 2006; Steyvers and Tenenbaum, 2005; Barabasi and Albert, 1999]. Further, in this case Greece would be the higher degree node that would relate topologically to new regions (e.g., Imbros) in a deterministic and natural manner. In mathematical terms, preferential attachment means that the probability that an existing node i with degree k_i acquires a link with a new node is $p(k_i) \simeq \frac{k_i}{\sum_j k_j}$. Among the different models to create random scale-free graphs, the Barabási-Albert (BA) model is the most well-studied and widely known one [Barabasi and Albert, 1999; Albert and Barabási, 2002]. The BA model considers growth and preferential attachment as follows. Regarding growth, it starts with an initial number m_0 of connected nodes and at each following step it adds a new node with $m \leq m_0$ edges that link the new node to m different existing nodes in the graph. When choosing the m different existing nodes to which the new node will be linked, the BA model assumes that the probability p that the new node will be connected to node i depends on the degree k_i of node i with a value given by the expression $p = \frac{k_i}{\sum_j k_j}$, which is the preferential attachment that we mentioned earlier. The degree distribution resulting from the BA model is a power law of the form $P(k) \simeq k^{-3}$, thus, it is able to create a subset of all the scale-free graphs that are characterised by a value γ such that $2 < \gamma < 3$. The scaling exponent



(a) regular graph (b) scale-free graph

Figure 5.8: Structures of a random regular graph with an average degree $k = 9$ and a scale-free graph with a preferential attachment $m = 2$, both having 100 nodes

is independent of m, the only parameter in the model (other than the order of the graph one would like to obtain of course). An algorithm that constructs a scale-free graph according to the BA model is presented in Algorithm 14.

Scale-free graphs are particularly attractive when trying to retain the sparseness of graphs (as partial $\diamond$-consistency does) because they have the following characteristics:

- scale-free graphs are always sparse [Del Genio *et al.*, 2011; Steyvers and Tenenbaum, 2005];
- scale-free graphs have a clustering coefficient distribution that also follows a power law, which implies that many low-degree nodes are clustered together forming very dense subgraphs that are connected to each other through major hubs [Dorogovtsev *et al.*, 2002].

Due to the aforementioned characteristics, scale-free graphs present themselves as excellent candidates for triangulation, as sparseness keeps time complexity for triangulation low and chordal graphs also exhibit a clustering structure [Golumbic, 2004], thus, they are able to fit scale-free graphs quite effectively. As an illustration of scale-free graphs, Figure 5.8 depicts a random regular graph, such as the ones used for experimental evaluation in the field of qualitative constraint-based reasoning, and a random scale-free graph generated using the BA model. Note that the bigger the node is, the higher its degree is (these are the hubs). An algorithm that implements this model and generates QCNs of scale-free constraint graphs is shown in Algorithm 15. For model BA(n, m) the average number of base relations per edge defaults to $|\mathsf{B}|/2$, where B is the set of base relations of the considered qualitative constraint language.

We describe the dataset used with regard to model BA(n, m). For RCC-8 we considered 30 networks for each size between 1000 and 10000 nodes with a 1000 step and a preferential attachment value of $m = 2$. For this specific value of m and for the network sizes considered, we found the networks to lie within the *phase transition* region, where it is equally possible for networks to be satisfiable or unsatisfiable, thus, they are harder to solve. For IA we considered 30 networks for each size between 500 and 5000 nodes with a 500 step and a preferential attachment

Algorithm 15: BA(n, m, $\mathcal{A} = \emptyset$)

in : The number of nodes n, the number of edges to attach from a new node to existing nodes m, and an optional subclass $\mathcal{A}$.
output : A scale-free network $\mathcal{N} = (V, C)$.

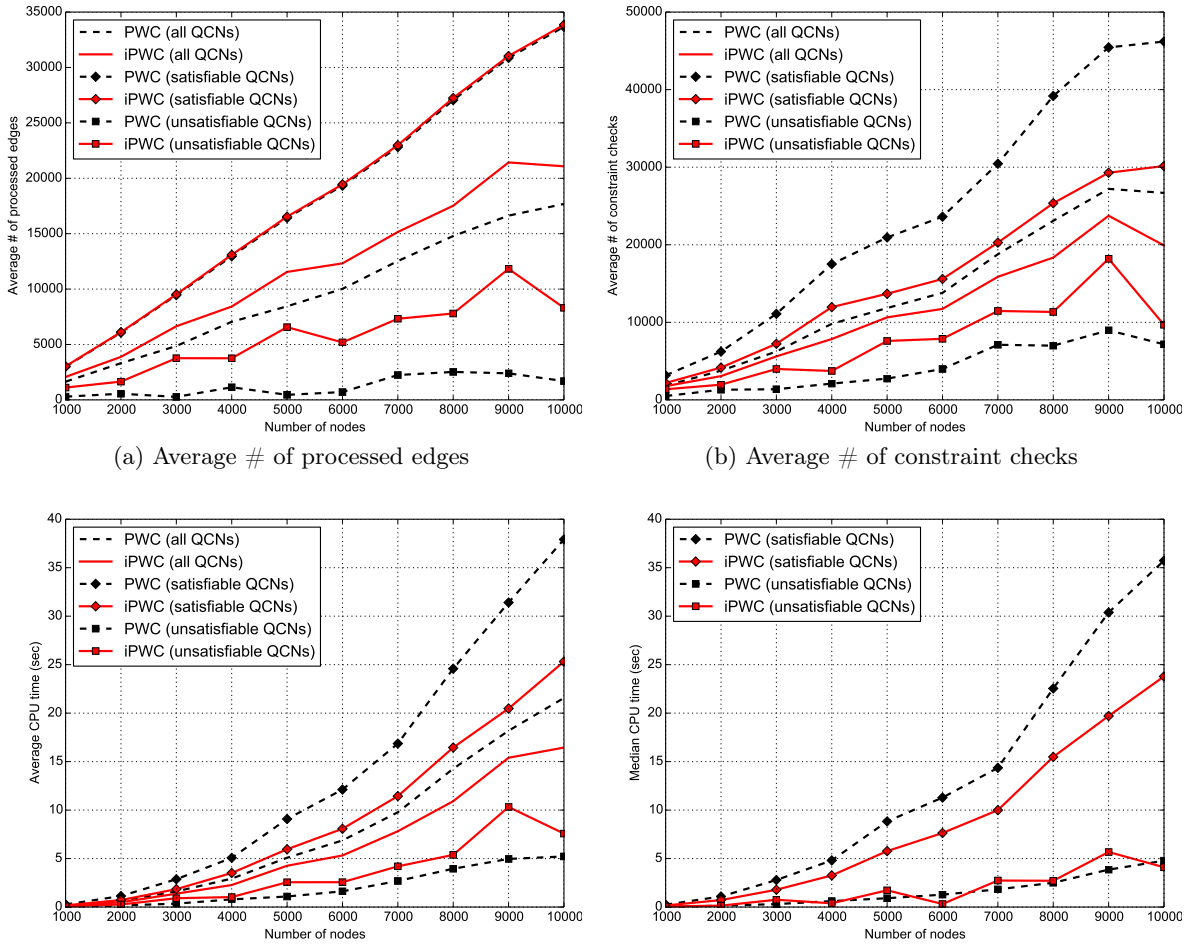
```
1 begin
2     G ← BA-Graph(n, m);
3     V ← V(G);
4     C ← map({((v, v') : (B if v ≠ v' else {EQ})) | v, v' ∈ V(G)});
5     foreach {v, v'} ∈ E(G) do
6         C(v, v') ← random.choice(2^B if A = ∅ else A); C(v', v) ← (C(v, v'))^-1;
7     return N = (V, C);
```

value of $m = 3$. We found that for this specific value of m and for the network sizes considered, the networks lie within the *phase transition* region, similarly to what was observed in the case of RCC-8. However, we note that the phase transition for Interval Algebra occurs for a greater value of m (viz., $m = 3$) than the value of m for RCC-8 (viz., $m = 2$). This is probably because Interval Algebra is a bigger calculus than RCC-8, containing 13 base relations instead of 8 respectively, which allows for satisfiable networks to be denser as there are more relations to be pruned and more relations that can *support* consistency in the network.

Regarding model A(n, d, l), we considered 50 QCNs of RCC-8 of 1000 variables and 50 QCNs of Interval Algebra of 500 variables in the phase transition region. As a reminder, and with respect to model A(n, d, l), n stands for the number of nodes in a given network, d for the average degree of its constraint graph, and l for the number of relations per edge in the constraint graph. Regarding RCC-8, the phase transition region is obtained for values $l = 4(= |\mathsf{B}|/2)$ and $8.0 \leq d \leq 11.0$ [Renz and Nebel, 2001], and regarding Interval Algebra, the phase transition region is obtained for values $l = 6.5(= |\mathsf{B}|/2)$ and $9.0 \leq d \leq 12.0$ [Nebel, 1997].

Regarding measures, our experimentation involves the average number of *constraint checks* performed and te average number of *edges processed* by a local consistency enforcing algorithm implementation, as well as the average and median (elapsed) *CPU time* of its operation. Note that the latter measures are strongly correlated with the first ones, as the runtime of local consistency enforcing algorithm implementations relies heavily on the number of constraint checks performed and edges processed by those implementations.

Results With regard to model BA(n, m), and RCC-8 in particular, on the average case, i.e., when all networks are considered, iPWC processes around 26.8% more edges than PWC, as shown in Figure 5.9a, iPWC performs around 15.3% less consistency checks than PWC, as shown in Figure 5.9b, and, finally, regarding average CPU time, iPWC runs around 18.9% faster than PWC, and 21.3% faster in the final step where networks of 10000 nodes are considered, as shown in Figure 5.9c; for the networks of 10000 nodes, iPWC runs using an average time of 15.3 sec per network and PWC using an average time of 19.4 sec per network. In Figure 5.9d we can also see the median CPU time for satisfiable and unsatisfiable networks. The interesting thing to note is that in the case of unsatisfiable networks the median allows us to discard some outlying measurements that influence the average CPU time in Figure 5.9c. As our dataset consists of a little more than 50% unsatisfiable networks for almost all network sizes, the diagram for the median CPU time for the combined dataset of satisfiable and unsatisfiable networks is very close to the dataset of the unsatisfiable networks only, as in almost all cases the median corresponds to the CPU processing time of an unsatisfiable network. Thus, we did not include this diagram



(a) Average # of processed edges

(b) Average # of constraint checks

(c) Average CPU time (sec)

(d) Median CPU time (sec)

Figure 5.9: Performance comparison of iPWC and PWC for RCC-8 networks

as it turned out to be pretty erratic and does not offer any additional information anyway.

With regard to model BA(n, m), and Interval Algebra in particular, on the average case, iPWC runs around 27.4% faster than PWC, and 34.4% faster in the final step where networks of 5000 nodes are considered; for the networks of 5000 nodes, iPWC runs using an average time of 44 sec per network and PWC using an average time of 67.1 sec per network.

With regard to model A(n, d, l), it should suffice to note that our experimentation yielded qualitatively similar results to those regarding model BA(n, m). In particular, for 50 RCC-8 networks of 1000 nodes and 50 IA networks of 500 nodes in the phase transition region there was a speed-up of around 20% and 30% respectively.

At this point we conclude our experimentation. We have demonstrated that iPWC performs better than PWC on average for random QCNs of IA and RCC-8. It should be noted that since iPWC works with a small queue in each incrementation step (as it initially considers only the constraints of a single network augmentation), in contrast to the queue utilized by PWC, iPWC is also much more memory efficient.

5.3 Directional Algebraic Closure and Directional $\diamond$-consistency

Given a QCN $\mathcal{N} = (V, C)$, the method of *directional algebraic closure* (also called *directional closure under weak composition*) removes certain base relations that are guaranteed to not participate in any solution of $\mathcal{N}$ by closing $\mathcal{N}$ under weak composition with respect to a particular ordering of its set of variables, i.e., with respect to a particular direction only. In particular, the directional algebraic closure applies the following iterative operation until a fixed state is reached:

$$\forall v_i, v_k, v_j \in V \text{ such that } i, j < k, \quad C(v_i, v_j) \leftarrow C(v_i, v_j) \cap (C(v_i, v_k) \diamond C(v_k, v_j))$$

Clearly, the method of directional algebraic closure considers less consistency operations than the method of algebraic closuse (which closes a given QCN with respect to all possible permutations of its set of variables and, hence, with respect to all possible directions). As such, the directional algebraic closure method is sound as it only removes base relations that do not participate in any solution of a given qualitative constraint network, but is not complete for deciding the satisfiability of any qualitative constraint network, i.e., we cannot conclude the satisfiability of an arbitrary qualitative constraint network if the directional algebraic closure method does not result in the assignment of the empty relation $\emptyset$ to a constraint of the network at hand. We will now present a local consistency, called directional $\diamond$-consistency, or simply $\overleftarrow{\diamond}$-consistency, that is directly related to the algebraic closure method and results from restricting $\diamond$-consistency, as presented in Section 3.5.1, to a particular ordering of the set of variables of an input network.

Definition 39 *A* QCN $\mathcal{N} = (V = \{v_0, v_1, \ldots, v_{n-1}\}, C)$ *is said to be* directionally $\diamond$-consistent*, or, simply,* $\overleftarrow{\diamond}$-consistent*, if and only if we have that* $C(v_i, v_j) \subseteq C(v_i, v_k) \diamond C(v_k, v_j)$, $\forall v_i, v_k, v_j \in V$ *with* $i, j < k$.

Given a QCN $\mathcal{N} = (V, C)$ with some ordering of its variables, the $\overleftarrow{\diamond}$*-consistent* QCN *of* $\mathcal{N}$ obtained after the application of the directional algebraic closure method on $\mathcal{N}$ with respect to that ordering of its variables is equivalent to $\mathcal{N}$ and unique with respect to that ordering of its variables. However, it should be noted that given two different orderings of the set of variables of $\mathcal{N}$, the respective directionally $\diamond$-consistent QCNs of $\mathcal{N}$ may differ in some of their constraints. The $\overleftarrow{\diamond}$-consistent QCN of $\mathcal{N}$ (with respect to a considered ordering of its set of variables) is called the closure of $\mathcal{N}$ under directional $\diamond$-consistency and it is denoted by $\overleftarrow{\diamond}(\mathcal{N})$. Network $\overleftarrow{\diamond}(\mathcal{N})$ corresponds to the largest (with respect to $\subseteq$) $\overleftarrow{\diamond}$-consistent sub-QCN of $\mathcal{N}$ with respect to some ordering of its set of variables). Given two QCNs $\mathcal{N} = (V, C)$ and $\mathcal{N}' = (V, C')$, and a particular ordering of the set of variables V, we can prove the following properties with respect to $\overleftarrow{\diamond}$-consistency:

- $\diamond(\mathcal{N}) \subseteq \overleftarrow{\diamond}(\mathcal{N}) \subseteq \mathcal{N}$ (Dominance);
- $\overleftarrow{\diamond}(\mathcal{N})$ is equivalent to $\mathcal{N}$ (Equivalence);
- $\overleftarrow{\diamond}(\overleftarrow{\diamond}(\mathcal{N})) = \overleftarrow{\diamond}(\mathcal{N})$ (Idempotence);
- if $\mathcal{N}' \subseteq \mathcal{N}$ then $\overleftarrow{\diamond}(\mathcal{N}') \subseteq \overleftarrow{\diamond}(\mathcal{N})$ (Monotonicity).

The aforementioned properties follow directly from the fact that $\overleftarrow{\diamond}$-consistency corresponds to $\diamond$-consistency restricted to a particular ordering of the set of variables of an input network. For simplicity, in what follows, when we state that a QCN is $\overleftarrow{\diamond}$-consistent, the ordering of the

variables of that QCN for the use case at hand will be implicitly considered. That ordering, if not provided separately, will be specified by the subscripts of the variables. For instance, a set of variables $\{v_0, v_1, \ldots, v_{n-1}\}$ specifies the ordered sequence of variables $(v_0, v_1, \ldots, v_{n-1})$. Next, we will show how $\overleftarrow{\diamond}$-consistency can be used to decide the satisfiability of QCNs under certain conditions. To this end, it will be necessary to introduce the following result by Long and Li:

Theorem 25 ([Long and Li, 2015]) *Let $\mathcal{N} = (V = \{v_0, v_1, \ldots, v_{n-1}\}, C)$ be a QCN defined over a distributive subclass of relations of a qualitative constraint language that is a relation algebra and for which every $\diamond$-consistent atomic QCN is satisfiable. If we have that $C(v_i, v_j) \subseteq C(v_i, v_{n-1}) \diamond C(v_{n-1}, v_j)$ for all $v_i, v_j \in V$ with $i, j < n-1$, then $\mathcal{N}{\downarrow}_{V \setminus \{v_{n-1}\}}$ is satisfiable only if $\mathcal{N}$ is satisfiable.*

Using Theorem 25, we prove the following proposition:

Proposition 26 *Let $\mathcal{N} = (V = \{v_0, v_1, \ldots, v_{n-1}\}, C)$ be a QCN defined over a distributive subclass of relations of a qualitative constraint language that is a relation algebra and for which every $\diamond$-consistent atomic QCN is satisfiable. If $\mathcal{N}$ is not trivially inconsistent and $\overleftarrow{\diamond}$-consistent, then $\mathcal{N}$ is satisfiable.*

Proof. Let V_l with $0 \leq l < n$ denote the set of variables $\{v_0, v_1, \ldots, v_l\}$, i.e., the set of variables of V from v_0 up to (and including) v_l. Then, due to $\mathcal{N}$ being $\overleftarrow{\diamond}$-consistent, for each $v_k \in V$ with $0 < k < n$ we have that $C(v_i, v_j) \subseteq C(v_i, v_k) \diamond C(v_k, v_j)$ for all $v_i, v_j \in V$ with $i, j < k$. As such, by Theorem 25, for each k with $0 < k < n$ we have that $\mathcal{N}{\downarrow}_{V_{k-1}}$ is unsatisfiable if $\mathcal{N}{\downarrow}_{V_k}$ is unsatisfiable. By the "if...then" transitive property, we have that $\mathcal{N}{\downarrow}_{V_1}$ is unsatisfiable if $\mathcal{N}{\downarrow}_{V} = \mathcal{N}$ is unsatisfiable. Let us assume that $\mathcal{N}$ is indeed unsatisfiable. Then, as $\mathcal{N}{\downarrow}_{V_1}[v_0, v_0] = \mathcal{N}{\downarrow}_{V_1}[v_1, v_1] = \{\mathsf{Id}\}$ by definition of a QCN, this can only mean that $\mathcal{N}{\downarrow}_{V_1}[v_0, v_1] = \emptyset$, or, equivalently, that $\mathcal{N}{\downarrow}_{V_1}[v_1, v_0] = \emptyset$, as $\mathcal{N}{\downarrow}_{V_1}[v_1, v_0] = (\mathcal{N}{\downarrow}_{V_1}[v_0, v_1])^{-1}$ by definition of a QCN. This is a contradiction, as $\mathcal{N}$ is not trivially inconsistent. We can conclude that if $\mathcal{N}$ is $\overleftarrow{\diamond}$-consistent and not trivially inconsistent, then $\mathcal{N}$ is satisfiable. ⊣

Due to Propositions 1 (at page 21), 3 (at page 34), and 26, we can obtain the following corollary:

Corollary 21 *Let $\mathcal{N} = (V = \{v_0, v_1, \ldots, v_{n-1}\}, C)$ be a QCN defined over a distributive subclass of relations of Point Algebra, Cardinal Direction Calculus, Interval Algebra, Block Algebra, or RCC-8. If $\mathcal{N}$ is not trivially inconsistent and $\overleftarrow{\diamond}$-consistent, then $\mathcal{N}$ is satisfiable.*

We now discuss some complexity results with respect to directional $\diamond$-consistency. Clearly, given a QCN $\mathcal{N} = (V = \{v_0, v_1, \ldots, v_{n-1}\}, C)$, $\overleftarrow{\diamond}$-consistency for $\mathcal{N}$ can be decided or determined in $O(n^3)$ time, where $n = |V|$, as there exist at most $n(n-1)(n-2)/3$ possible triples of variables v_i, v_j, and v_k in V such that $i, j < k$ and a single check $C(v_i, v_j) \subseteq C(v_i, v_k) \diamond C(v_k, v_j)$ is performed for every such triple of variables. Actually, in practice $\overleftarrow{\diamond}$-consistency for $\mathcal{N}$ can be decided in $O(\delta^2|V|)$ time, where δ is the maximum vertex degree of the constraint graph $G(\mathcal{N})$ of $\mathcal{N}$, if we choose to disregard the universal relations in $\mathcal{N}$. This will become more apparent when we will present our implementation of the directional algebraic closure method through an algorithm for applying or enforcing directional $\diamond$-consistency on $\mathcal{N}$.

Algorithm 16: DWC$(\mathcal{N}, \alpha)$

in : A QCN $\mathcal{N} = (V, C)$ of a qualitative constraint language that is a relation algebra with $|V| = n$, and a bijection α of V onto $\{0, 1, \ldots, n-1\}$.
output : $\overleftarrow{\diamond}(\mathcal{N})$.

```
1  begin
2      G ← (V, E = E(G(N)));
3      for x from n − 1 to 1 do
4          v ← α^{-1}(x);
5          adj ← {v' | {v', v} ∈ E ∧ α(v') < α(v)};
6          foreach v', v'' ∈ adj do
7              if α(v') < α(v'') then
8                  if {v', v''} ∉ E then
9                      E ← E ∪ {{v', v''}};
10                 t = C(v', v'') ∩ (C(v', v) ⋄ C(v, v''));
11                 if t ≠ C(v', v'') then
12                     C(v', v'') ← t; C(v'', v') ← t^{-1};
13     return N;
```

5.3.1 The DWC Algorithm

An algorithm that implements the directional algebraic closure method is provided in Algorithm 16, called DWC. As opposed to any of the local consistency enforcing algorithms that we have discussed up to this point, viz., WC, PWC, and iPWC, algorithm DWC does not need a queue to store constraints or any other data structure for that matter. Given a QCN $\mathcal{N} = (V, C)$ with $|V| = n$ and a bijection α of V onto $\{0, 1, \ldots, n-1\}$, we rewrite each variable $u \in V$ as $v_{\alpha(u)}$. As such, the set of variables V will be the set $\{v_0, v_1, \ldots, v_{n-1}\}$. Upon termination of algorithm DWC, we will have a network $\mathcal{N}' \subseteq \mathcal{N}$ and for each $v_k \in V$ with $0 < k < n$ we will have that $N'[v_i, v_j] \subseteq N'[v_i, v_k] \diamond N'[v_k, v_j]$ for all $v_i, v_j \in V$ with $i, j < k$. Thus, by definition of $\overleftarrow{\diamond}$-consistency, we have that $\mathcal{N}'$ will be $\overleftarrow{\diamond}$-consistent, or, more precisely, that there exists a QCN $\overleftarrow{\diamond}(\mathcal{N})$ such that $\mathcal{N}' = \overleftarrow{\diamond}(\mathcal{N})$, viz., the $\overleftarrow{\diamond}$-consistent QCN of $\mathcal{N}$ with respect to the considered ordering of the variables of V. Note that in each iteration of the outer loop (line 3) we only choose constraints that correspond to non-universal relations (as a requirement to discard of weak compositions that yield the universal relations) or contraints that have been involved in a previous consistency operation; we keep track of these constraints through maintaing a set of edges E that at any point contains the corresponding pairs of variables of all processed constraints up to that point. Further, note that we exploit the properties of $^{-1}$-involution and $^{-1}$-involutive distributivity met in a relation algebra to calculate the result of a weak composition between two relations and its converse in a single step (line 7), i.e., given three variables v_i,v_j, and v_k such that $i < j < k$, we only calculate $C(v_i, v_j) \cap (C(v_i, v_k) \diamond C(v_k, v_j))$, as $C(v_j, v_i) \cap (C(v_j, v_k) \diamond C(v_k, v_i))$ can be obtained by conversing $C(v_i, v_j) \cap (C(v_i, v_k) \diamond C(v_k, v_j))$ (line 12). (The soundness of this technique has been proven formally in the proof of Lemma 4 at page 99.) As such, we can assert the following proposition:

Proposition 27 *Given a* QCN $\mathcal{N}$ *over* n *variables of a qualitative constraint language that is a relation algebra, and a bijection* α *of* V *onto* $\{0, 1, \ldots, n-1\}$*, algorithm* DWC *terminates and returns* $\overleftarrow{\diamond}(\mathcal{N})$.

Given a QCN $\mathcal{N} = (V, C)$, algorithm DWC has a runtime of $O(\delta^2|V|)$, where δ is the maximum vertex degree of the graph G utilized in its operation, as it iterates $O(|V|)$ variables (lines 3–4) and performs $O(\delta^2)$ constant operations for the variable v at hand in each iteration (lines 5–12). It should be noted that algorithm DWC is built on the idea of the *variable elimination* algorithm, a simple and general exact inference algorithm in probabilistic graphical models, such as Bayesian networks and Markov random fields [Zhang and Poole, 1994]. Notably, a variant of the variable elimination algorithm has also been introduced for finite-domain constraint satisfaction problems (CSPs) defined over *connected row convex* constraints [Deville *et al.*, 1999] by Zhang et al. in [Zhang and Marisetti, 2009; Zhang, 2007]. Of course, in our case variables are not really eliminated in the process, they are just ignored, at some point, from further consideration.

Given a QCN $\mathcal{N}$, we note that algorithm DWC may consider pairs of variables that do not exist in $E(G(\mathcal{N}))$ (lines 2–9), which are nevertheless involved in consistency operations with respect to the constraints that are associated with them (lines 10–12). It would be interesting to explore if there exists a condition such that no pair of variables that does not exist in $E(G(\mathcal{N}))$ needs to be considered, as this would allow us to perform less consistency operations in general. To this end, the bijection α that is given as input to algorithm VarElimination plays an important role, as it defines the ordering in which the variables of V are eliminated. Let us show when the ordering defined by α guarantees that no pair of variables that does not exist in $E(G(\mathcal{N}))$ will be considered by algorithm VarElimination. To do so, we will first show that algorithm DWC constructs a chordal graph as a byproduct. In particular we have the following result:

Proposition 28 *Given a* QCN $\mathcal{N} = (V, C)$*, with* $|V| = n$*, and a bijection* $\alpha : V \to \{0, 1, \ldots, n-1\}$*, we have that upon termination of algorithm* DWC*, the internally used graph* G *will be a chordal graph such that* $G(\mathcal{N}) \subseteq G$*.*

Proof. Graph G is initialized to $(V, E = E(G(\mathcal{N})))$ (line 2), where V is also the set of vertices of $G(\mathcal{N})$. Let us show that $G(\mathcal{N}) \subseteq G$ and G is also chordal after the termination of algorithm DWC. As defined earlier in this chapter, G_i is the subgraph of G induced by $\{\alpha^{-1}(0),\alpha^{-1}(1),\ldots, \alpha^{-1}(i)\}$, with $0 \leq i < n$. For each x from $n-1$ to 1, line 5 provides us with the set of neighbors of vertex $\alpha^{-1}(x)$ in G_x, denoted by $N_x(\alpha^{-1}(x))$. Then, in lines 6–9 we add edges to E (if not existing in E and, thus, neither in $E(G(\mathcal{N}))$) such that every two vertices in $N_x(\alpha^{-1}(x))$ become connected by an edge in G_x. As such, vertex $\alpha^{-1}(x)$ becomes a simplicial vertex of G_x, since the subgraph of G_x induced by $N_x(\alpha^{-1}(x))$ becomes complete. (Note that the check $\alpha(v') < \alpha(v'')$ in line 7 for vertices $v', v'' \in V$ does not cause a problem, as we deal with edges that are not ordered pairs of vertices, but rather doubletons of vertices; this is also apparent from our notation.) After processing the set of vertices $\{\alpha^{-1}(1),\alpha^{-1}(2),\ldots,\alpha^{-1}(n-1)\}$ of V, we will have admitted a perfect elimination ordering $(\alpha^{-1}(n-1),\alpha^{-1}(n-2),\ldots,\alpha^{-1}(0))$ of G through the addition of new edges to the set of edges E when necessary. As such, we have that the edge-augmented graph $G = (V, E)$ of $G(\mathcal{N})$ is a chordal graph by Theorem 23 (at page 90). We can conclude that upon termination of algorithm DWC, the internally used graph G is a chordal graph such that $G(\mathcal{N}) \subseteq G$. ⊣

As shown in the proof of Proposition 28, given a QCN $\mathcal{N} = (V, C)$, with $|V| = n$, and a bijection $\alpha : V \to \{0, 1, \ldots, n-1\}$, algorithm VarElimination treats the ordering $(\alpha^{-1}(n-1),\alpha^{-1}(n-2),\ldots,\alpha^{-1}(0))$ as a perfect elimination ordering of $G(\mathcal{N})$ and consequently constructs a chordal supergraph G of $G(\mathcal{N})$. We can assert the following result:

Proposition 29 *Given a* QCN $\mathcal{N} = (V, C)$*, with* $|V| = n$*, and a bijection* $\alpha : V \to \{0, 1, \ldots, n-1\}$*, we have that upon termination of algorithm* DWC*, the internally used graph* G *will be a chordal*

graph such that $G(\mathcal{N}) = G$ *if* $(\alpha^{-1}(n-1),\alpha^{-1}(n-2),\ldots,\alpha^{-1}(0))$ *is a perfect elimination ordering of* $G(\mathcal{N})$.

Proposition 29 ensures that if its specified condition holds, then no pair of variables that does not exist in $E(G(\mathcal{N}))$ will be considered by algorithm DWC. We already saw earlier in this chapter in our discussion about chordal graphs, tringulations, and perfect elimination orderings, that a perfect elimination ordering of a given chordal graph is easily obtainable. As a reminder, given a chordal graph $G = (V, E)$, with $|V| = n$, we can obtain a perfect elimination ordering of G in $O(|V| + |E|)$ time using the *maximum cardinality search* (MCS) algorithm [Tarjan and Yannakakis, 1984], which visits the vertices of a graph in an order such that, at any point, a vertex is visited that has the largest number of visited neighbors. Consequently, MCS produces a bijection $\alpha : V \to \{0, 1, \ldots, n-1\}$ such that $(\alpha^{-1}(n-1),\alpha^{-1}(n-2),\ldots,\alpha^{-1}(0))$ is a perfect elimination ordering of G. Given a QCN $\mathcal{N} = (V, C)$, if $G(\mathcal{N})$ is not chordal, MCS will define an elimination ordering of the variables of $\mathcal{N}$, which, although not perfect, in general will allow less pairs of variables that do not exist in $E(G(\mathcal{N}))$ to be considered by algorithm VarElimination, than a randomly chosen elimination ordering. An alternative would be to use some special greedy heuristic instead of the MCS algorithm to obtain an elimination ordering, the simplest and fastest of which being the *approximate minimum degree* heuristic [Heggernes *et al.*, 2001]. This heuristic has a runtime of $O(|V||E|)$ for a given graph $G = (V, E)$ [Heggernes *et al.*, 2001]. Thus, its use can still be an overkill for large QCNs, which are of our particular interest in this chapter. Another choice is the *minimum fill-in* heuristic with a runtime of $O(|V|^3)$ [Rose, 1972; Jégou and Terrioux, 2014b], which again makes its use prohibitive for large QCNs. However, such heuristics can still prove to be beneficial when dealing with small QCNs.

Up to this point, we have showed how we can efficiently decide the satisfiability of a given QCN defined over a distributive subclass of relations. Next, we will show how a scenario of that QCN can actually be extracted. The contribution involves performing a generic backtrack-free procedure for refining a $\overleftarrow{\diamond}$-consistent and not trivially inconsistent QCN $\mathcal{N}$ defined over a distributive subclass of relations to an atomic $\overleftarrow{\diamond}$-consistent sub-QCN of $\mathcal{N}$. To this end, we have the following result:

Proposition 30 *Let* $\mathcal{N} = (V = \{v_0, v_1, \ldots, v_{n-1}\}, C)$ *be a* $\overleftarrow{\diamond}$*-consistent and not trivially inconsistent* QCN *defined over a distributive subclass of relations of a qualitative constraint language that is a relation algebra and for which every* $\diamond$*-consistent atomic* QCN *is satisfiable. Then,* $\mathcal{N}$ *can be refined to a scenario* $\mathcal{S}$ *of* $\mathcal{N}$ *as follows. For each* k *from* 1 *to* $n-1$*, and for each* $i \in \{0, \ldots, k-1\}$*, do:*

- $C(v_k, v_i) \leftarrow \bigcap_{j=0}^{k-1} C(v_k, v_j) \diamond C(v_j, v_i)$;
- $C(v_k, v_i) \leftarrow \{b\}$ *for some* $b \in C(v_k, v_i)$;
- $C(v_i, v_k) \leftarrow (C(v_k, v_i))^{-1}$.

Proof. Let V_l with $0 \leq l < n$ denote the set of variables $\{v_0,v_1,\ldots,v_l\}$, i.e., the set of variables of V from v_0 up to (and including) v_l. By Proposition 26 (at page 110) we have that $\mathcal{N}$ is satisfiable. As such, $\mathcal{N}{\downarrow_{V_{l'}}}$ is satisfiable for some $0 \leq l' < n-1$ (as with any other restriction of $\mathcal{N}$ to a proper subset of its variables). Since $\mathcal{N}{\downarrow_{V_{l'}}}$ is satisfiable, we can refine it to a scenario $\mathcal{S}{\downarrow_{V_{l'}}}$ of $\mathcal{N}{\downarrow_{V_{l'}}}$, and have that $\mathcal{N}[v_j, v_{j'}] = \mathcal{S}{\downarrow_{V_{l'}}}[v_j, v_{j'}]$ for every $j, j' \in \{0, \ldots, l'\}$. We will show that we can extend that scenario to a scenario of $\mathcal{N}{\downarrow_{V_{l'+1}}}$ with our proposed construction that is specified by the three successive operations listed in our proposition. It is clear that $\mathcal{N}{\downarrow_{V_{l'+1}}}$ is

$\overleftarrow{\diamond}$-consistent with respect to the variable ordering $\{v_0,v_1,\ldots,v_{l'+1}\}$ and not trivially inconsistent. Let $\mathcal{N}_i$ with $i \in \{0,\ldots,l'\}$ denote $\bigcap_{j=0}^{l'} \mathcal{N}[v_{l'+1},v_j] \diamond \mathcal{S}{\downarrow_{V_{l'}}}[v_j,v_i]$. First, we need to show that $\mathcal{N}_i$ is not empty. By Theorem 1 (at page 23) it suffices to show that $\mathcal{N}[v_{l'+1},v_j] \diamond \mathcal{S}{\downarrow_{V_{l'}}}[v_j,v_i] \cap \mathcal{N}[v_{l'+1},v_{j'}] \diamond \mathcal{S}{\downarrow_{V_{l'}}}[v_{j'},v_i] \neq \emptyset$ for all $j,j' \in \{0,\ldots,l'\}$. As $\mathcal{N}$ is defined on a qualitative constraint language that is a relation algebra, due to the Peircean law that holds for relation algebras (cf. [Dylla *et al.*, 2013]), we have that $\mathcal{N}[v_{l'+1},v_j] \diamond \mathcal{S}{\downarrow_{V_{l'}}}[v_j,v_i] \cap \mathcal{N}[v_{l'+1},v_{j'}] \diamond \mathcal{S}{\downarrow_{V_{l'}}}[v_{j'},v_i] \neq \emptyset$ iff $\mathcal{N}[v_{l'+1},v_{j'}] \diamond \mathcal{S}{\downarrow_{V_{l'}}}[v_{j'},v_i] \diamond \mathcal{S}{\downarrow_{V_{l'}}}[v_i,v_j] \cap \mathcal{N}[v_{l'+1},v_j] \neq \emptyset$ iff $\mathcal{N}[v_{j'},v_{l'+1}] \diamond \mathcal{N}[v_{l'+1},v_j] \cap \mathcal{S}{\downarrow_{V_{l'}}}[v_{j'},v_i] \diamond \mathcal{S}{\downarrow_{V_{l'}}}[v_i,v_j] \neq \emptyset$. As $\mathcal{S}{\downarrow_{V_{l'}}}[v_{j'},v_j] \subseteq \mathcal{S}{\downarrow_{V_{l'}}}[v_{j'},v_i] \diamond \mathcal{S}{\downarrow_{V_{l'}}}[v_i,v_j]$ and $\mathcal{S}{\downarrow_{V_{l'}}}[v_{j'},\ v_j] \subseteq \mathcal{N}[v_{j'},v_{l'+1}] \diamond \mathcal{N}[v_{l'+1},v_j]$, we have that $\mathcal{N}_i$ is not empty. As such, there exists a base relation $b \in \mathsf{B}$ such that $b \in \mathcal{N}_i$. Let us assign the base relation b to $\mathcal{N}[v_{l'+1},v_i]$, and have that $\mathcal{N}[v_{l'+1},v_i] = \{b\}$ and $\mathcal{N}[v_i,v_{l'+1}] = \{b^{-1}\}$. We will now show that the refined $\mathcal{N}{\downarrow_{V_{l'+1}}}$ remains $\overleftarrow{\diamond}$-consistent with respect to the variable ordering $\{v_0,v_1,\ldots,v_{l'+1}\}$ and not trivially inconsistent. As $b \neq \emptyset$, $\mathcal{N}{\downarrow_{V_{l'+1}}}$ remains not trivially inconsistent. To show that $\mathcal{N}{\downarrow_{V_{l'+1}}}$ also remains $\overleftarrow{\diamond}$-consistent with respect to the considered variable ordering, we need to show that $\mathcal{N}[v_j,v_i] \subseteq \mathcal{N}[v_j,v_{l'+1}] \diamond \mathcal{N}[v_{l'+1},v_i]$, i.e., $\mathcal{N}[v_j,v_i] \subseteq \mathcal{N}[v_j,v_{l'+1}] \diamond \{b\}$, for every $j \in \{0,\ldots,l'\}$. As $b \in \mathcal{N}_i$, we have that $\{b\} \subseteq \mathcal{N}[v_{l'+1},v_j] \diamond \mathcal{N}[v_j,v_i]$. (Note that $\mathcal{N}[v_j,v_i] = \mathcal{S}{\downarrow_{V_{l'}}}[v_j,v_i]$.) Thus, $\{b\} \cap \mathcal{N}[v_{l'+1},v_j] \diamond \mathcal{N}[v_j,v_i] \neq \emptyset$. Then, due to the Peircean law, we have that $\mathcal{N}[v_j,v_{l'+1}] \diamond \{b\} \cap \mathcal{N}[v_j,v_i] \neq \emptyset$. As $\mathcal{N}[v_j,v_i]$ is a singleton relation, we can only have that $\mathcal{N}[v_j,v_i] \subseteq \mathcal{N}[v_j,v_{l'+1}] \diamond \{b\}$. Up to this point, we have shown that given any scenario $\mathcal{S}{\downarrow_{V_{l'}}}$ of $\mathcal{N}{\downarrow_{V_{l'}}}$ for some $0 \leq l' < n-1$, and as long as $\mathcal{N}{\downarrow_{V_{l'+1}}}$ remains $\overleftarrow{\diamond}$-consistent and not trivially inconsistent, then, for any $i \in \{0,\ldots,l'\}$, every base relation $b \in \mathcal{N}_i$ is a feasible base relation of $\mathcal{N}{\downarrow_{V_{l'+1}}}$ with regard to constraint $\mathcal{N}[v_{l'+1},v_i]$. Further, we showed that after assigning to $\mathcal{N}[v_{l'+1},v_i]$ a base relation of $\mathcal{N}_i$, the refined $\mathcal{N}{\downarrow_{V_{l'+1}}}$ will remain $\overleftarrow{\diamond}$-consistent and not trivially inconsistent. This property allows us to extend the scenario $\mathcal{S}{\downarrow_{V_{l'}}}$ to a scenario of $\mathcal{N}{\downarrow_{V_{l'+1}}}$ by (*i*) choosing some not already considered constraint $\mathcal{N}[v_{l'+1},v_i]$ of $\mathcal{N}{\downarrow_{V_{l'+1}}}$, for some $i \in \{0,\ldots,l'\}$, and calculating $\mathcal{N}_i$ with respect to the refined $\mathcal{N}{\downarrow_{V_{l'+1}}}$ that has resulted from the assignment of a base relation to each one of its already considered constraints (if any), (*ii*) soundly assigning any base relation of $\mathcal{N}_i$ to $\mathcal{N}[v_{l'+1},v_i]$ and, thus, further refining $\mathcal{N}{\downarrow_{V_{l'+1}}}$, and (*iii*) repeating the procedure for all not already considered constraints of $\mathcal{N}{\downarrow_{V_{l'+1}}}$ by going back to (*i*). It is important to stress that in each loop of the aforementioned procedure, a relation $\mathcal{N}_i$, for some $i \in \{0,\ldots,l'\}$, is by definition calculated with respect to the refined $\mathcal{N}{\downarrow_{V_{l'+1}}}$ that has resulted from the assignment of a base relation to each one of its already considered constraints; in technical terms, in each loop, $\mathcal{N}{\downarrow_{V_{l'+1}}}$ is mutated (i.e., *refined* in our context) in place, and a calculation of some $\mathcal{N}_i$ takes into account the latest refinement of $\mathcal{N}{\downarrow_{V_{l'+1}}}$. The procedure terminates when all unassigned constraints of $\mathcal{N}{\downarrow_{V_{l'+1}}}$ have been considered, and $\mathcal{N}{\downarrow_{V_{l'+1}}}$ has become an atomic and satisfiable (by Proposition 26 at page 110) QCN. Using this procedure, and with respect to k and i as they appear in our proposition, we can refine $\mathcal{N}$ to a scenario $\mathcal{S}$ of $\mathcal{N}$, by strictly following the ordering $(1,\ldots,n-1)$ for k, and any permutation of the ordering $(0,\ldots,k-1)$ for i. In particular, we start by assigning to $\mathcal{N}[v_0,v_1]$ any base relation b such that $b \in \mathcal{N}[v_0,v_1]$, as $\mathcal{N}{\downarrow_{V_1}}$ being a QCN of two variables is minimal and weakly globally consistent, and acquire the rest of the scenario up to $\mathcal{N}$ incrementally, at which point $\mathcal{N}$ will have been refined to an atomic satisfiable QCN. ⊣

Finally, we present an algorithm for extracting a scenario of a given satisfiable QCN $\mathcal{N} = (V,C)$ defined over a distributive subclass of relations. The algorithm is given in Algorithm 17 and is called ExtractScenario. First, algorithm ExtractSolution uses algorithm DWC to make $\mathcal{N}$ $\overleftarrow{\diamond}$-consistent and, then, it applies the procedure specified in Proposition 30 to refine $\mathcal{N}$ to a scenario of $\mathcal{N}$.

Algorithm 17: ExtractScenario($\mathcal{N}$)

in : A satisfiable QCN $\mathcal{N} = (V, C)$ of a qualitative constraint language that is a relation algebra and for which every $\diamond$-consistent atomic QCN is satisfiable, with $|V| = n$.

output: A scenario $\mathcal{S}$ of $\mathcal{N}$.

```
1  begin
2      α ← (f : V → {0, 1, ..., n − 1}) s.t. α is bijective;
3      N ← DWC(N, α);
4      for x from 1 to n − 1 do
5          v ← α^{-1}(x);
6          foreach v' ∈ V | α(v') < α(v) do
7              adj ← {v'' ∈ V | α(v'') < α(v)};
8              C(v, v') ← ⋂_{v''∈adj} C(v, v'') ◇ C(v'', v');
9              C(v, v') ← {b} for some b ∈ C(v, v');
10             C(v', v) ← (C(v, v'))^{-1};
11     return N;
```

With respect to algorithm ExtractScenario, we can assert the following theorem in relation to Propositions 1 (at page 21) and 3 (at page 34):

Theorem 26 *Given a satisfiable* QCN $\mathcal{N} = (V, C)$*, with* $|V| = n$*, defined over a distributive subclass of relations of a qualitative constraint language that is a relation algebra and for which every* $\diamond$*-consistent atomic* QCN *is satisfiable, algorithm* ExtractScenario *terminates and returns a scenario* $\mathcal{S}$ *of* $\mathcal{N}$*.*

Algorithm ExtractScenario has a runtime of $O(|V|^3)$, which includes the runtime of algorithm DWC (line 3), and the time needed to iterate $O(|V|)$ variables (lines 4–5) and realize $O(|V|^2)$ constant operations for the variable v at hand in each iteration (lines 6–10).

5.4 Efficient Algorithms for the Satisfiability Problem of QCNs

In the case where a QCN is defined over a distributive subclass of relations, and since algorithm DWC is able to enforce $\overleftarrow{\diamond}$-consistency on a given QCN of a relation algebra due to Proposition 27 (at page 111), we have the following result:

Proposition 31 *Let* $\mathcal{A} \in 2^{\mathsf{B}}$ *be a subclass of relations of a qualitative constraint language that is a relation algebra, over which not trivially inconsistent and* $\overleftarrow{\diamond}$*-consistent* QCN*s are satisfiable. Then, given a* QCN $\mathcal{N} = (V, C)$ *defined over* $\mathcal{A}$*, with* $|V| = n$*, and a bijection* α *of* V *onto* $\{0, 1, \ldots, n-1\}$*, algorithm* DWC *terminates and returns a trivially inconsistent* QCN $\overleftarrow{\diamond}(\mathcal{N})$ *if and only if* $\mathcal{N}$ *is unsatisfiable.*

From Proposition 31 we can assert the following theorem:

Theorem 27 *Let* $\mathcal{A} \in 2^{\mathsf{B}}$ *be a subclass of relations of a qualitative constraint language that is a relation algebra, over which not trivially inconsistent and* $\overleftarrow{\diamond}$*-consistent* QCN*s are satisfiable. Then, given a* QCN $\mathcal{N} = (V, C)$ *defined over* $\mathcal{A}$*, with* $|V| = n$*, and a bijection* α *of* V *onto* $\{0, 1, \ldots, n-1\}$*, algorithm* DWC *is sound and complete for deciding the satisfiability of* $\mathcal{N}$*.*

From Theorem 27 and Proposition 26 (at page 110) we can assert the following theorem:

Theorem 28 *Let $\mathcal{A} \in 2^{\mathsf{B}}$ be a distributive subclass of relations of a qualitative constraint language that is a relation algebra and for which every $\diamond$-consistent atomic* QCN *is satisfiable. Then, given a* QCN *$\mathcal{N} = (V, C)$ defined over $\mathcal{A}$, with $|V| = n$, and a bijection α of V onto $\{0, 1, \ldots, n-1\}$, algorithm* DWC *is sound and complete for deciding the satisfiability of $\mathcal{N}$.*

Finally, due to Theorem 28, and Propositions 1 (at page 21) and 3 (at page 34), we have the following result:

Corollary 22 *Let $\mathcal{N} = (V, C)$ be a* QCN *defined over a distributive subclass of relations of Point Algebra, Cardinal Direction Calculus, Interval Algebra, Block Algebra, or* RCC-*8. We have that algorithm* DWC *is sound and complete for deciding the satisfiability of $\mathcal{N}$.*

In the case where a QCN is defined over a subclass of relations of a qualitative constraint language that has patchwork for not trivially inconsistent and $\diamond$-consistent QCNs defined over that subclass of relations, and since algorithm PWC is able to enforce partial $\diamond$-consistency on a given QCN of a relation algebra due to Proposition 24 (at page 97), by Proposition 19 (at page 90) we can obtain the following result:

Proposition 32 *Let $\mathcal{A} \in 2^{\mathsf{B}}$ be a subclass of relations of a qualitative constraint language that is a relation algebra and has patchwork for not trivially inconsistent and $\diamond$-consistent* QCN*s defined over that subclass of relations. Then, given a* QCN *$\mathcal{N} = (V, C)$ defined over $\mathcal{A}$ and a chordal graph $G = (V, E)$ such that $G(\mathcal{N}) \subseteq G$, algorithm* PWC *terminates and returns a trivially inconsistent* QCN *$^{\diamond}_{G}(\mathcal{N})$ if and only if $\mathcal{N}$ is unsatisfiable.*

From Proposition 32 we can assert the following theorem:

Theorem 29 *Let $\mathcal{A} \in 2^{\mathsf{B}}$ be a subclass of relations of a qualitative constraint language that is a relation algebra and has patchwork for not trivially inconsistent and $\diamond$-consistent* QCN*s defined over that subclass of relations. Then, given a* QCN *$\mathcal{N} = (V, C)$ defined over $\mathcal{A}$ and a chordal graph $G = (V, E)$ such that $G(\mathcal{N}) \subseteq G$, algorithm* PWC *is sound and complete for deciding the satisfiability of $\mathcal{N}$*

Then, due to Theorem 29 and Propositions 1 (at page 21) and 16 (at page 54), we have the following result:

Corollary 23 *Let $\mathcal{N} = (V, C)$ be a* QCN *of Point Algebra, Cardinal Direction Calculus, Interval Algebra, Block Algebra, or* RCC-*8, defined over one of the classes $\mathcal{P}_{\mathsf{PA}}$, $\mathcal{P}_{\mathsf{CDC}}$, $\mathcal{H}_{\mathsf{IA}}$, $\mathcal{H}^{n}_{\mathsf{IA}}$, or $\hat{\mathcal{H}}_8$, $\mathcal{C}_8$, and $\mathcal{Q}_8$ respectively. We have that algorithm* PWC *is sound and complete for deciding the satisfiability of $\mathcal{N}$.*

Following the same line of reasoning and using Proposition 25 (at page 101), we can obtain the following result as well:

Theorem 30 *Let $\mathcal{A} \in 2^{\mathsf{B}}$ be a subclass of relations of a qualitative constraint language that is a relation algebra and has patchwork for not trivially inconsistent and $\diamond$-consistent* QCN*s defined over that subclass of relations. Then, given a* QCN *$\mathcal{N} = (V, C)$ defined over $\mathcal{A}$ and a chordal graph $G = (V, E)$ such that $G(\mathcal{N}) \subseteq G$, algorithm* iPWC *is sound and complete for deciding the satisfiability of $\mathcal{N}$*

Then, due to Theorem 30 and Propositions 1 (at page 21) and 16 (at page 54), we have the following result:

Corollary 24 *Let $\mathcal{N} = (V, C)$ be a* QCN *of Point Algebra, Cardinal Direction Calculus, Interval Algebra, Block Algebra, or* RCC-*8, defined over one of the classes $\mathcal{P}_{\mathsf{PA}}$, $\mathcal{P}_{\mathsf{CDC}}$, $\mathcal{H}_{\mathsf{IA}}$, $\mathcal{H}^n_{\mathsf{IA}}$, or $\hat{\mathcal{H}}_8$, $\mathcal{C}_8$, and $\mathcal{Q}_8$ respectively. We have that algorithm* iPWC *is sound and complete for deciding the satisfiability of $\mathcal{N}$.*

For the general case of QCNs, i.e., for QCNs that are not strictly defined over subclasses of relations for which partial or directional $\diamond$-consistency alone are able to decide satisfiability, some kind of backtracking algorithm must be used that operates on the basis of splitting the relations of a given QCN into disjunctions of relations belonging to a tractable subclass for which partial or directional $\diamond$-consistency alone are indeed able to decide satisfiability. The approach is completely similar to the one we presented in Section 3.5.2 with regard to the backtracking algorithm involing $\diamond$-consistency, in the sense that we will simply replace $\diamond$-consistency with partial $\diamond$-consistency as the *forward checking* step of choice in the backtracking algorithm. In what follows, and with respect to backtracking algorithms in particular, we will focus on partial $\diamond$-consistency only as it is a more general approach that allows for efficient backtracking algorithm implementations. In particular, we have showed earlier that partial $\diamond$-consistency is able to decide the satisfiability of QCNs over maximal tractable subclasses of their relations, whereas directional $\diamond$-consistency applies to QCNs over distributive subclasses of relations, which are by default properly contained in maximal tractable sublasses of relations; the bigger a subclass of relations, the smaller the search space in a backtracking algorithm that utilizes that subclass of relations (see Section 3.5.2), which is critical in its performance. However, we make a discussion about how directional $\diamond$-consistency can be utilized in a backtracking algorithm and we give some directions for future work regarding that matter in the end of this section.

5.4.1 The PartialConsistency Algorithm

An algorithm that implements a backtracking search for solving general QCNs is provided in Algorithm 18, called PartialConsistency. Algorithm PartialConsistency receives as input a QCN $\mathcal{N} = (V, C)$, a graph $G = (V, E)$, and a subclass of relations $\mathcal{A}$. First, $\overset{\diamond}{}_G$-consistency is enforced through the PWC algorithm (line 2), which we described in detail in Section 5.2.1. Then, granted that enforcing $\diamond$-consistency did not result in a trivially inconsistent network, algorithm PartialConsistency chooses a constraint $C(v_i, v_j)$ with $\{v_i, v_j\} \in E$ such that $C(v_i, v_j) \notin \mathcal{A}$ (line 7). In the case where such a constraint does not exist, a not trivially inconsistent and $\overset{\diamond}{}_G$-consistent refinement of network $\mathcal{N}$ defined over subclass $\mathcal{A}$ is returned as a solution. In any other case, constraint $C(v_i, v_j)$ is split into subrelations $r_1, ..., r_k \in \mathcal{A}$, for which we have that $r_1 \cup ... \cup r_k = C(v_i, v_j)$ (line 8). Each of these relations is instantiated accordingly to the constraint network $\mathcal{N}$ (line 10) and a recursive call is initiated (line 11). When the algorithm termitates, it is guaranteed to return either Null, or a not trivially inconsistent and $\overset{\diamond}{}_G$-consistency refinement of network $\mathcal{N}$ defined over subclass $\mathcal{A}$ otherwise (line 13). Note that except for the first step, in all subsequent recursive calls of the PartialConsistency algorithm, the PWC algorithm only has to be run for the paths that are possibly affected by each prior instantiation, which takes $O(|E|)$ intersections and weak compositions. (Note that this detail is not included in the PWC algorithm.) To obtain this constraint incremental variation of the PWC algorithm, it suffices to replace the command $Q \leftarrow \{(v_i, v_j) \mid \{v_i, v_j\} \in E \text{ with } 0 \leq i \leq j < |V|\}$ in line 2 of the PWC algorithm, with the command $Q \leftarrow \{(v_i, v_j)\}$, where v_i, v_j with $i \leq j$ are the variables appearing

Algorithm 18: PartialConsistency($\mathcal{N}$, G, $\mathcal{A}$)

```
   in      : A QCN N = (V, C), a graph G = (V, E), and a subclass A.
   output  : Null, or a refinement of network N over A.
1  begin
2      N ← PWC(N, G);
3      if ∃{v_i, v_j} ∈ E such that C(v_i, v_j) = ∅ then
4          return Null;
5      if ∀{v_i, v_j} ∈ E we have that C(v_i, v_j) ∈ A then
6          return N;
7      choose a constraint C(v_i, v_j) with {v_i, v_j} ∈ E such that C(v_i, v_j) ∉ A;
8      split C(v_i, v_j) into r_1, ..., r_k ∈ A: r_1 ∪ ... ∪ r_k = C(v_i, v_j);
9      foreach r ∈ {r_l | 1 ≤ l ≤ k} do
10         N[v_i, v_j] ← r; N[v_j, v_i] ← r^-1;
11         result ← PartialConsistency(N, G, A);
12         if result ≠ Null then
13             return result;
14     return Null;
```

in the constraint $C(v_i, v_j)$ in line 10 of the PartialConsistency algorithm. As such, the PWC algorithm can be seen as approximating consistency for general QCNs, but also as realising *forward checking* in the backtracking algorithm by closing all triples of variables of a given QCN under weak composition and eliminating base relations that are unfeasible. With respect to the first step of algorithm PartialConsistency, we note that it is possible to replace the call to algorithm PWC in line 2 with a call to algorithm iPWC instead and benefit from a notable performance gain for achieving $\overset{\diamond}{}_G$-consistency as we explained in Section 5.2.2. That being said, a call to algorithm iPWC would not provide any benefits over a call to algorithm PWC with regard to all subsequent steps of algorithm PartialConsistency; once $\overset{\diamond}{}_G$-consistency is established vertex-incrementally on a given QCN using algorithm iPWC, a constraint incremental variation of the iPWC algorithm will operate exactly as a constraint incremental variation of the PWC algorithm (all vertices of G are already considered and, thus, algorithm iPWC falls back to algorithm PWC since its objective of establishing $\overset{\diamond}{}_G$-consistency has already been achieved).

Heuristics. With respect to heuristics, the same heuristics regarding constraint and subrelation selection as well as other techniques described in the context of algorithm Consistency (Algorithm 2 at page 42)—all of which are thoroughly presented in Section 3.5.2—apply for algorithm PartialConsistency too. In the case of algorithm PartialConsistency, we can additionally take into account the structural properties of the graph G that is given as input to algorithm PartialConsistency. As an example, assuming that graph G contains more than one connected components, it would be wise to initially focus on the QCNs corresponding to the smaller components, as any inconsistency found there would immediately render the QCN corresponding to the entire (disconnected) graph unsatisfiable and we would not have to explore bigger components of the graph. Further, a structured graph G may contain vertices of a high average degree, at least in comparison with other vertices. Therefore, it would be natural to start with constraints corresponing to edges of high average degree vertices, as choosing a relation for those constraints may result in a faster propagation overall and a quicker unveiling of a possible inconsistency. This particular heuristic can be used in conjuction with the cardinality heuristic, which typically

prefers a constraint with the smallest number of subrelations without taking into account its "neighbouring" constraints. Such heuristics have been explored for CSPs in the work of Walsh in [Walsh, 2001], and can be easily carried to the field of constraint-based qualitative spatial and temporal reasoning.

Given a QCN $\mathcal{N} = (V, C)$, a graph $G = (V, E)$, and a subclass $\mathcal{A}$, the search space for algorithm PartialConsistency is $O(\alpha^{|E|})$, where α is the branching factor (see line 8 of the PartialConsistency algorithm) provided by subclass $\mathcal{A}$ (e.g., $\alpha = 1.4375$ for subclass $\hat{\mathcal{H}}_8$ for RCC-8 [Renz and Nebel, 2001]). With respect to the PartialConsistency algorithm, due to Proposition 32 (at page 116) we have the following result:

Proposition 33 *Let $\mathcal{A} \in 2^{\mathsf{B}}$ be a subclass of relations of a qualitative constraint language that is a relation algebra and has patchwork for not trivially inconsistent and $\diamond$-consistent QCNs defined over that subclass of relations. Then, given $\mathcal{A}$, a QCN $\mathcal{N}$ defined over 2^{B}, and a chordal graph $G = (V, E)$ such that $G(\mathcal{N}) \subseteq G$, algorithm PartialConsistency terminates and returns Null if and only if $\mathcal{N}$ is unsatisfiable.*

From Proposition 33 we can assert the following theorem:

Theorem 31 *Let $\mathcal{A} \in 2^{\mathsf{B}}$ be a subclass of relations of a qualitative constraint language that is a relation algebra and has patchwork for not trivially inconsistent and $\diamond$-consistent QCNs defined over that subclass of relations. Then, given $\mathcal{A}$, a QCN $\mathcal{N}$ defined over 2^{B}, and a chordal graph $G = (V, E)$ such that $G(\mathcal{N}) \subseteq G$, algorithm PartialConsistency is sound and complete for deciding the satisfiability of $\mathcal{N}$.*

Then, due to Theorem 31, and Propositions 1 (at page 21) and 16 (at page 54), we have the following result:

Corollary 25 *Given one of the classes $\mathcal{P}_{\mathsf{CDC}}$, $\mathcal{H}_{\mathsf{IA}}$, $\mathcal{H}^n_{\mathsf{IA}}$, or $\hat{\mathcal{H}}_8$, $\mathcal{C}_8$, and $\mathcal{Q}_8$, a QCN $\mathcal{N}$ of Cardinal Direction Calculus, Interval Algebra, Block Algebra, or RCC-8 respectively, and a chordal graph $G = (V, E)$ such that $G(\mathcal{N}) \subseteq G$, we have that algorithm PartialConsistency is sound and complete for deciding the satisfiability of $\mathcal{N}$.*

5.4.2 The IterativePartialConsistency Algorithm

Like with the iterative counterpart of algorithm Consistency presented in Section 3.5.2, namely, algorithm IterativeConsistency, we can obtain the iterative counterpart of algorithm PartialConsistency in a very similar manner. Again, the recursive and iterative algorithms are functionally equivalent; however, the structure of the iterative algorithm makes it easier to fine-tune it, performance-wise. We call the iterative algorithm IterativePartialConsistency, presented in Algorithm 19.

5.4.3 Reasoners

In this section we describe our own implementations of qualitative reasoners for solving general QCNs that build on the new techniques that we presented up to this point, and consequently our practical approach of choice for tackling large scale-free QCNs of RCC-8. In order to obtain a more complete comparison with the state of the art techniques that we presented in Section 3.5.2, we also present our own implementation of a qualitative reasoner that builds strictly on those techniques (much like GQR [Gantner *et al.*, 2008]).

Algorithm 19: IterativePartialConsistency($\mathcal{N}$, $\mathcal{A}$)

```
in     : A QCN 𝒩 = (V, C), a graph G = (V, E), and a subclass 𝒜.
output : Null, or a refinement of network 𝒩 over 𝒜.
1  begin
2      Stack ← list(∅) // Initialize stack;
3      𝒩 ← PWC(𝒩, G);
4      if ∃{v_i, v_j} ∈ E such that C(v_i, v_j) = ∅ then
5          return Null;
6      while True do
7          if ∀{v_i, v_j} ∈ E we have that C(v_i, v_j) ∈ 𝒜 then
8              return 𝒩;
9          choose a constraint C(v_i, v_j) with {v_i, v_j} ∈ E such that C(v_i, v_j) ∉ 𝒜;
10         split C(v_i, v_j) into r_1, ..., r_k ∈ 𝒜: r_1 ∪ ... ∪ r_k = C(v_i, v_j);
11         r_values ← {r_l | 1 ≤ l ≤ k};
12         while True do
13             if r_values = ∅ then
14                 while Stack ≠ ∅ do
15                     𝒩, r_values ← Stack.pop();
16                     if r_values then
17                         break;
18                 if r_values = ∅ and Stack = ∅ then
19                     return Null;
20             r ← r_values.pop();
21             𝒩' ← 𝒩;
22             𝒩[v_i, v_j] ← r; 𝒩[v_j, v_i] ← r^-1;
23             𝒩 ← PWC(𝒩, G);
24             if ∄{v_i, v_j} ∈ E such that C(v_i, v_j) = ∅ then
25                 break;
26             𝒩 ← 𝒩';
27         Stack.append(𝒩', r_values);
```

Phalanx. We have implemented Phalanx in Python that is the generalized and code refactored version of PyRCC8 [Sioutis and Koubarakis, 2012]. Phalanx supports small arbitrary binary constraint calculi developed for spatial and temporal reasoning that are relation algebras, such as RCC-8 [Randell *et al.*, 1992] and Allen's interval algebra (IA) [Allen, 1983], in a way similar to GQR [Gantner *et al.*, 2008]. Further, Phalanx presents significant improvements over PyRCC8 regarding scalability and speed. In particular, the new reasoner handles the constraint matrix that represents a qualitative network more efficiently during backtracking search, i.e., it does not create a copy of the matrix at each forward step of the backtracking algorithm (as is the case with Renz's solver [Renz and Nebel, 2001] or PyRCC8), but it only keeps track of the values that are altered at each forward step to be able to reconstruct the matrix in the case of backtracking. A similar mechanism is also used to keep track of unassigned constraints (i.e., constraints corresponding to relations that do not belong to maximal tractable subclasses of relations and are decomposed to subrelations at each forward step of the backtracking algorithm) that may dynamically change in number due to the appliance of $\diamond$-consistency at each forward step of the backtracking algorithm. For example, given a non-tractable RCC-8 network and a

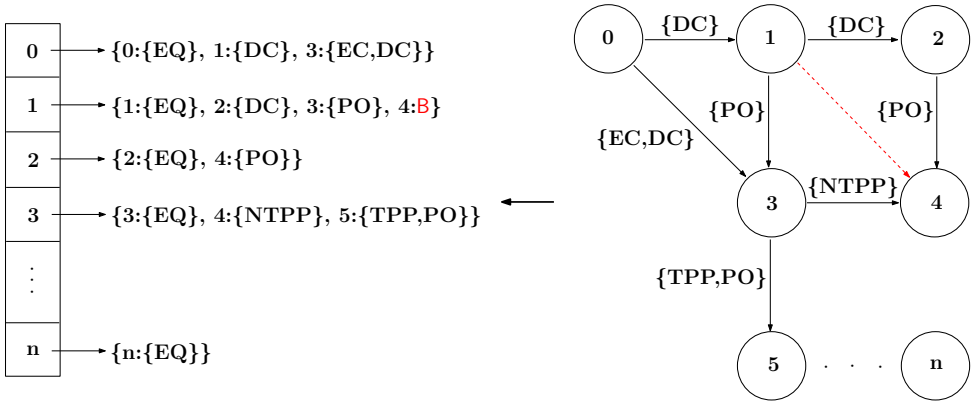


Figure 5.10: Hash table based adjacency list for representing a chordal RCC-8 network

maximal tractable subclass $\mathcal{A}$, the $\diamond$-consistency algorithm can prune a relation that belongs to subclass $\mathcal{A}$ into a relation that does not belong to subclass $\mathcal{A}$, and vice versa.[16] This allows us to apply the heuristics that deal with the selection of the next unassigned constraint faster, as we keep our set of unassigned constraints minimal and do not have to go over the whole set of variables each time to find the ones that are unassigned. The $\diamond$-consistency algorithm implementation has been also modified to better handle the cases where weak composition of relations leads to the universal relation. In these cases we can continue the iterative operation of the $\diamond$-consistency algorithm since the intersection of the universal relation with any other relation leaves the latter relation intact. Finally, there is a weight over learned weights dynamic heuristic for unassigned constraint selection, as in the latest version of GQR, under release 1500[17].

Phalanx$\triangledown$. We have implemented Phalanx$\triangledown$ in Python that is the generalized and code refactored version of PyRCC8$\triangledown$ [Sioutis and Koubarakis, 2012]. Phalanx$\triangledown$ is essentially Phalanx where the $\diamond$-consistency algorithm has been replaced with a partial $\diamond$-consistency algorithm, as described in Section 5.2.1. Phalanx$\triangledown$ supports small arbitrary binary constraint calculi developed for spatial and temporal reasoning that are relation algebras, and for which Proposition 19 (at page 90) holds, such as RCC-8 [Randell *et al.*, 1992] and Allen's Interval Algebra (IA) [Allen, 1983]. Regarding triangulation, Phalanx$\triangledown$ is coupled with the implementation of the maximum cardinality search algorithm [Tarjan and Yannakakis, 1984] and a fast fill in procedure (as presented in Algorithm 10 at page 89), as opposed to the heuristic based, but rather naive, triangulation procedure implemented in [Sioutis and Koubarakis, 2012]. Though the maximum cardinality search algorithm does not yield minimal triangulations if the constraint graph of the input network is not chordal, it does guarantee that no fill edges are inserted if the graph is indeed chordal, as explained earlier in the discussion around Algorithm 10 (at page 89). In addition, even for the non-chordal cases, we obtain good results with this approach and have a fine trade-off between time efficiency and good triangulations. As with the case of Phalanx and PyRCC8, Phalanx$\triangledown$ presents significant improvements over PyRCC8$\triangledown$ regarding scalability and speed, by incorporating all the state of the art techniques that apply to Phalanx and were mentioned earlier.

[16] Note that in the case where the RCC-8 network comprises only relations from the maximal tractable subclass $\mathcal{A}$ (i.e., it is tractable), this would not be possible as $\mathcal{A}$ is closed under the $\diamond$-consistency operations.

[17] `http://sfbtr8.informatik.uni-freiburg.de/R4LogoSpace/downloads/gqr-1500.tar.bz2`

Table 5.1: Triangulation time based on different methods

triangulation method	BA-Graph(10 000, 2)
MCS	2.26s
MD	112.004s
GFI	–

Sarissa. We improve the state of the art techniques for tackling large scale-free QCNs of RCC-8 by opting for a hash table based adjacency list to represent and reason with the chordal completion of the constraint graph of an input network. The variables of the input network (or the nodes) are represented by index numbers of a list, and each variable (or node) is associated with a hash table that stores key-value pairs of variables and relations. Figure 5.10 shows how an example chordal RCC-8 network is represented by our hash table based adjacency list approach. Loops (of the identity relation EQ) have been omitted from the network. The dashed edge $(1, 4)$ corresponds to a fill edge that results after triangulating the initial non-chordal graph consisting of solid edges. This fill edge is stored in the hash table based adjacency list as the universal relation B. For a given QCN $N = (V, C)$ and for $G = (V, E)$ its chordal constraint graph, our approach requires $O(|V| + |E|b)$ memory, where b is the size needed to represent a relation from the set of relations 2^{B} of the qualitative constraint language at hand. Having a constraint matrix to represent an input QCN (that is typically used by the reasoners Phalanx and Phalanx$_{\bigtriangledown}$ presented earlier), results in a $O(|V|^2 b)$ memory requirement, even if chordal graphs are used leaving a big part of the matrix empty, as is the case with Phalanx$_{\bigtriangledown}$, or Sparrow for IA [Chmeiss and Condotta, 2011]. Further, we still retain an $O(1)$ average access and update time complexity which becomes $O(\delta)$ in the amortized worst case, where δ is the maximum vertex degree of the chordal constraint graph that corresponds to the input network. Given that we target large scale-free, and, thus, sparse networks, this only incures a small penalty for the related experiments performed. The partial $\diamond$-implementation also benefits from this approach as the queue data structure on which it is based has to use only $O(|E|)$ of memory to store the relations compared to the $O(|V|^2)$ memory requirement of Phalanx, GQR, and Renz's solver. The triangulation algorithm is based on Algorithm 10 (at page 89), as with Phalanx$_{\bigtriangledown}$. As mentioned earlier in this chapter, an alternative would be to use some special greedy heuristic rather than the maximum cardinality search (MCS) algorithm to obtain an elimination ordering, the simplest and fastest of which being the *minimum degree* (MD) heuristic [Heggernes *et al.*, 2001]. As noted in [Heggernes *et al.*, 2001], this heuristic has a time complexity of $O(|V||E|)$ for a given graph $G = (V, E)$, which is an overkill for the large networks that are of our interest here. Another choice would be the (minimum) greedy fill-in (GFI) heuristic (used in [Amaneddine *et al.*, 2013; Chmeiss and Condotta, 2011]) with a time complexity of $O(|V|^3)$ [Rose, 1972; Jégou and Terrioux, 2014b], which again marks it as prohibitive in our case. Nevertheless, we present in Table 5.1 the time needed to calculate a triangulation based on the aforementioned methods on average per graph of a small dataset of 10 graphs created by the BA-Graph(10000, 2) generation model (Algorithm 14 at page 105). The experiment was carried out on a computer with an Intel Core 2 Quad Q9400 processor with a CPU frequency of 2.66 GHz, 8 GB RAM, and the Precise Pangolin x86_64 OS (Ubuntu Linux), and shows that even for the simplest MD heuristic, the triangulation operation alone for a given graph, takes more time than triangulating that graph with MCS and solving the network corresponding to that graph with a reasoner, as observed in Figure 5.15a (at page 130) for random network instances with constraint graphs of the same

Table 5.2: Tabular overview of our reasoners

reasoner	data structure	core algorithm	decision capability
Phalanx	adjacency matrix	$\diamond$-consistency	arbitrary QCNs
Phalanx$_\triangledown$	adjacency matrix	partial $\diamond$-consistency	arbitrary QCNs
Sarissa	hash table based adjacency list	partial $\diamond$-consistency	arbitrary QCNs
Pyrrhus	hash table based adjacency list	$\overleftarrow{\diamond}$-consistency	distributive QCNs

type and order as the ones used in our experiment.[18] Note that GFI was not able to compute a triangulation within the 5-minute timeout it was given. Closing our parenthesis on the different triangulations we explored, the techniques that we mentioned prior to our small comparison are implemented under the hood of our new reasoner which is called Sarissa. Sarissa, as with our other tools presented here, is a generic and open source qualitative reasoner written in Python, a general-purpose, interpreted high-level programming language which enables rapid application development. In the experimens that follow, Sarissa will be shown to greatly outperform state of the art reasoners, such as GQR, for large RCC-8 networks of scale-free structure. However, in defense of GQR which was found to perform poorly in [Sioutis and Koubarakis, 2012] under release 1418, we state that the latest version of GQR has undergone massive scalability improvements and is currently the most complete and fastest reasoner for handling reasonably scaled random *regular* qualitative networks, i.e., qualitative constraint networks with unstructured constraint graphs. At this point, we can also claim that Renz's solver [Renz and Nebel, 2001] has been fairly outdated, as it will become apparent in the experiments that we employ it for. Finally, our hash table based adjacency list allows extending a network with new nodes and edges in constant time. This is particularly handy in the case of the vertex-incremental partial $\diamond$-consistency algorithm that we presented in Section 5.2.2, as it allows obtaining online partial $\diamond$-consistency algorithm implementations.

Pyrrhus. We implement the $\overleftarrow{\diamond}$-consistency enforcing algorithm DWC (as presented in Algorithm 16 at page 111) under the hood of a novel reasoner called Pyrrhus. Pyrrhus borrows the same hash table based adjacency list structure to represent and reason with the chordal completion of the constraint graph of an input network from Sarissa. In the time of writting this thesis, Pyrrhus cannot be used to solve arbitrary QCNs as it only implements Algorithm 16 (at page 111), which can be used to soundly decide the satisfiability of a QCN that it strictly defined over a distributive subclass of relations of a qualitative constraint language (see Theorem 28 at page 116). For future work, we plan to devise an edge-incremental variant of the DWC algorithm and fit it in a backtracking algorithm as a forward checking step; the role of the backtracking algorithm will be to spawn a search tree given a QCN and retrieve a $\overleftarrow{\diamond}$-consistent refinement of that QCN that will be defined over a distributive subclass of relations. However, we will demonstrate that Pyrrhus, even in its current state, can be extremely useful in deciding the satisfiability of real-world QCNs, as these QCNs are in general defined over base relations, the closure of which under converse, intersection, and weak composition defines a distributive subclass of relations. Finally, and as noted earlier, variable elimination orderings to be used alongside the DWC algorithm are calculated with the use of the maximum cardinality search (MCS) algorithm.

A tabular overview of the reasoners presented in this section is given in Table 5.2. All tools used in this thesis are open-source, under a free license, and comprise a new set of qualitative reasoners that build on state of the art Artificial Intelligence techniques. They can be acquired

[18] We used the LibTW library for MD and GFI (`http://treewidth.com/`).

along with the datasets used in this thesis upon request from the author and also found online in the following address: `http://www.cril.fr/~sioutis/work.php`.

5.4.4 Experimental evaluation

In this section, we evaluate the performance of our algorithms both for the particular case where the QCNs are defined over a distributive subclass of relations (to showcase the importance of $\overleftarrow{\diamond}$-consistency alone and its potential in future applications) and for the case where arbitrary QCNs are concerned.

Evaluation with Distributive QCNs

We evaluate the performance of our implementation of the $\overleftarrow{\diamond}$-consistency enforcing DWC algorithm, against state of the art implementations of $\diamond_G$-consistency and $\diamond$-consistency enforcing algorithms, for checking the satisfiability of a given QCN defined over a distributive subclass of relations. As noted earlier, algorithm DWC is implemented under the hood of the novel reasoner Pyrrhus. A state of the art $\diamond_G$-consistency enforcing algorithm implementation is provided by reasoner Sarissa and a state of the art $\diamond$-consistency enforcing algorithm implementation is provided by reasoner Phalanx.

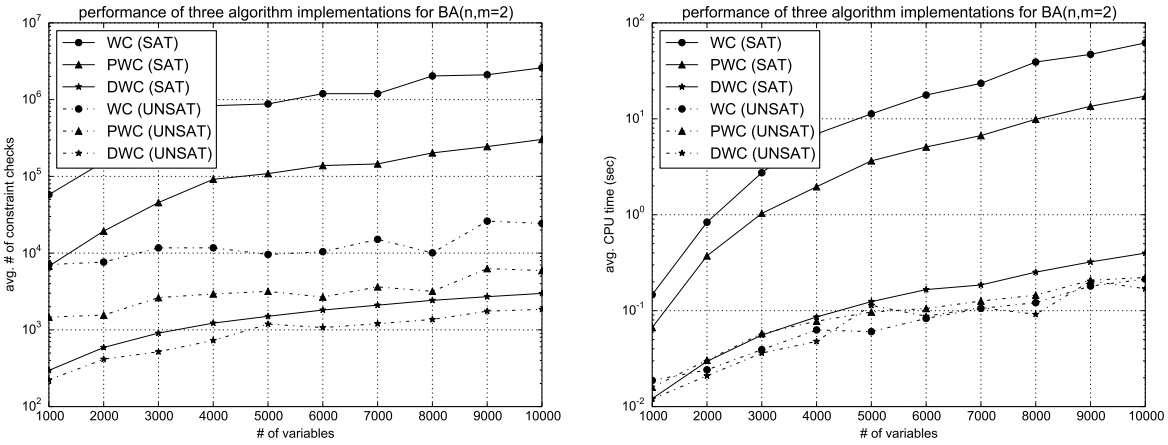
Technical Specifications The experimentation was carried out on a computer with an Intel Core i7-2820QM processor with a 2.30 GHz frequency per CPU core, 8 GB of RAM, and the Trusty Tahr x86_64 OS (Ubuntu Linux). Pyrrhus, Sarissa, and Phalanx were run with PyPy 2.2.1[19], which implements Python 2.7. Only one of the CPU cores was used.

Dataset and Measures We considered random RCC-8 networks generated by the BA(n, m) model [Barabasi and Albert, 1999], the use of which in qualitative constraint-based reasoning has been well motivated in the experimental evaluation that took place in Section 5.2.2, and real-world RCC-8 datasets that have been recently used in [Sioutis *et al.*, 2015f].

In particular, we used the BA(n, m) model (Algorithm 15 at page 107) to create random scale-free graphs of order n with a preferential attachment value m; each such graph was then treated as the constraint graph of a given QCN of RCC-8, by labeling the constraints of the QCN corresponding to edges of the graph with relations from the maximal distributive subclass $\mathcal{D}_8^{64}$ of RCC-8 [Li *et al.*, 2015b], and the rest of the constraints of the QCN strictly with the universal relation, viz., B. We considered 10 satisfiable and 10 unsatisfiable RCC-8 network instances of BA(n, m) for each order $1000 \leq n \leq 10000$ of their constraint graphs with a 1000-vertex step and a preferential attachment value of $m = 2$. Both satisfiable and unsatisfiable network instances were randomly filtered out of a large number of 1 000 network instances to ensure validity of the results. Regarding real-world RCC-8 datasets, we employed the ones recently used in [Sioutis *et al.*, 2015f], described as follows (in the description by constraints we mean non-universal relations).

- `nuts`: an RCC-8 network of a nomenclature of territorial units with 2 235/3 176 variables/constraints.[20]
- `adm1`: an RCC-8 network of the administrative geography of Great Britain with 11 762/44 832 variables/constraints [Goodwin *et al.*, 2008].

[19] `http://pypy.org/`
[20] Retrieved from: `http://www.linkedopendata.gr/`



(a) Comparison on # of constraint checks (b) Comparison on CPU time

Figure 5.11: Performance comparison for random scale-free RCC-8 networks

- `gadm1`: an RCC-8 network of the German administrative units with 42 749/159 600 variables/constraints.[20]
- `gadm2`: an RCC-8 network of the world's administrative areas with 276 729/589 573 variables/constraints.[21]
- `adm2`: an RCC-8 network of the administrative geography of Greece with 1 732 999/5 236 270 variables/constraints.[20]

The aforementioned datasets are satisfiable. Further, the relations of each dataset are contained in one of the maximal distributive subclasses $\mathcal{D}_8^{41}$ and $\mathcal{D}_8^{64}$ of RCC-8 [Li *et al.*, 2015b].

Our experimentation involves two measures, which we describe as follows. The first measure considers the number of *constraint checks* performed by a local consistency enforcing algorithm implementation. Given a QCN $\mathcal{N} = (V, C)$ and $v_i, v_k, v_j \in V$, a constraint check is performed when we compute relation $r = C(v_i, v_j) \cap (C(v_i, v_k) \diamond C(v_k, v_j))$ and check if $r \subset C(v_i, v_j)$, so that we can propagate its constrainedness. (Weak compositions that yield relation B are disregarded.) The second measure concerns the CPU time and is strongly correlated with the first one, as the runtime of local consistency enforcing algorithm implementations relies heavily on the number of constraint checks performed.

Results In what follows, WC (for $\diamond$-consistency or closure under weak composition) will denote the $\diamond$-consistency enforcing algorithm implementation of Phalanx, PWC (for $\overset{\diamond}{}_G$-consistency or partial closure under weak composition) will denote the $\overset{\diamond}{}_G$-consistency enforcing algorithm implementation of Sarissa, and DWC (for $\overleftarrow{\diamond}$-consistency or directional closure under weak composition) will denote the $\overleftarrow{\diamond}$-consistency enforcing algorithm implementation of Pyrrhus, viz., the implementation of algorithm DWC. As a final note, the maximum cardinality search algorithm was used to obtain a variable elimination ordering for DWC, and a triangulation of the constraint graph of a given QCN for PWC (to be able to make sound use of Proposition 19 at page 90).

Regarding random scale-free RCC-8 networks, the experimental results are shown in Figure 5.11. DWC performs significantly less constraint checks than PWC and WC for both satisfiable

[21] `http://gadm.geovocab.org/`

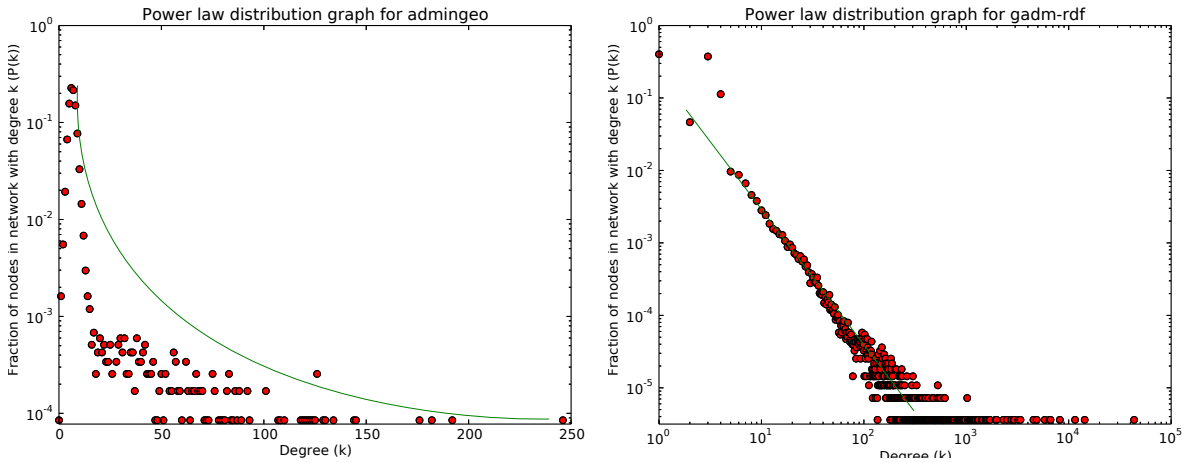
Table 5.3: Evaluation with real-world RCC-8 datasets

network	WC	PWC	DWC
nuts	$\frac{0.12s}{9\,632}$	$\frac{0.09s}{5\,808}$	$\frac{0.08s}{1\,180}$
adm1	$\frac{13\,783.39s}{1\,287\,288\,879}$	$\frac{83.36s}{18\,498\,096}$	$\frac{0.31s}{92\,266}$
gadm1	$\frac{25\,587.76s}{1\,927\,158\,080}$	$\frac{105.07s}{34\,140\,998}$	$\frac{0.82s}{339\,611}$
gadm2	$\frac{6.72s}{1\,891\,032}$	$\frac{1.61s}{1\,885\,100}$	$\frac{0.56s}{483\,377}$
adm2	∞	$\frac{399.60s}{118\,799\,994}$	$\frac{2.25s}{5\,471\,745}$

and unsatisfiable network instances, as shown in Figure 5.11a. In particular, across all network instances of different size, DWC performs on average 98.2% and 99.8% less constraint checks than PWC and WC respectively for satisfiable network instances, and 70.1% and 92.2% less constraint checks than PWC and WC respectively for unsatisfiable network instances. This also reflects on the CPU time, as shown in Figure 5.11b. In particular, across all network instances of different size, DWC is on average 94.7% and 98.0% faster than PWC and WC respectively for satisfiable network instances, and 20.8% and 1.8% faster than PWC and WC respectively for unsatisfiable network instances. We note that for unsatisfiable network instances all approaches are in a virtual tie, as they unveil the inconsistencies in centiseconds and any difference in performance is thus marginal.

Regarding real-world RCC-8 datasets, the experimental results are summarized in Table 5.3, where a fraction $\frac{x}{y}$ denotes that an approach required x seconds of CPU time and performed y constraint checks to decide the satisfiability of a given network instance. Symbol ∞ denotes that an implementation hit the memory limit. Again, we can see that DWC significantly outperforms PWC and WC with regard to both the CPU time required and the number of constraint checks performed for deciding the satisfiability of a network instance. It should suffice to mention that for the largest of the network instances, viz., adm2, DWC decides its satisfiability in 2.25 sec, when PWC requires 399.60 sec for the same task, and WC does not even complete that task as it hits the memory limit after several hours of reasoning. The same trend holds for the number of constraint checks performed by the different algorithm implementations.

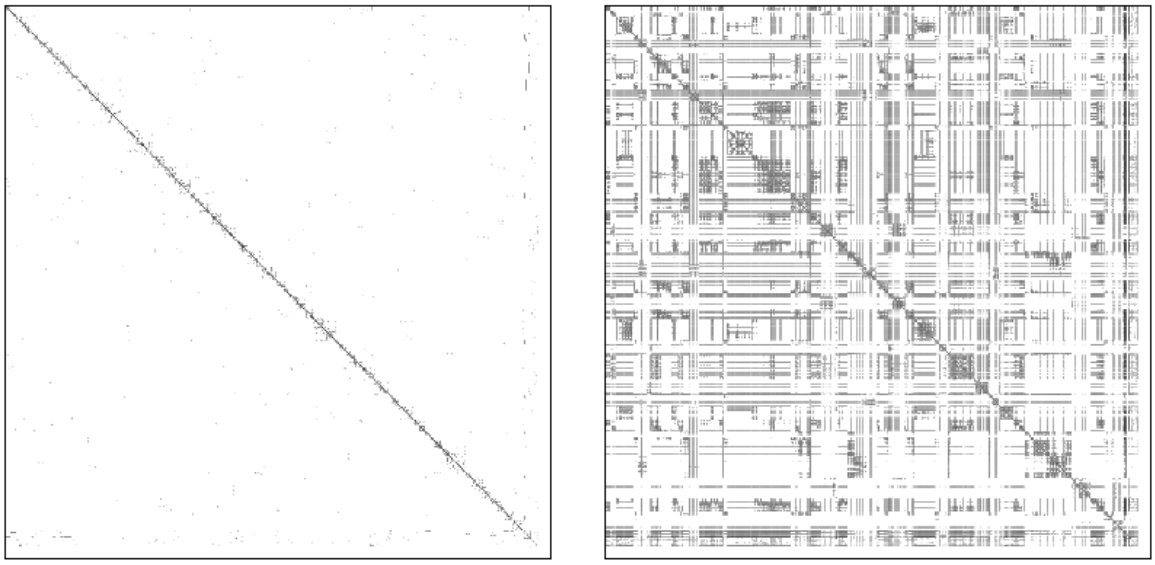
In the context of Proposition 28 (at page 112), it is worth investigating how triangulations can maintain the sparseness of scale-free structured real-world RCC-8 datasets. To backup our argument about the scale-free-like structure of real RCC-8 networks, we present Figure 5.12 that displays the power law degree distribution of two of our real networks. As gadm2 is a very large network, we display its degree distribution in log-log scale where the power law function is seen as a straight line [Barabasi and Albert, 1999]. We remind the reader that adm1 and gadm2 comprise 11 762/44 832 nodes/edges and 276 729/589 573 nodes/edges respectively. Triangulating the constraint graph of adm1 results in a total of 1 961 820 edges, which is 97% less than completing the graph that results in $(11\,762^2 - 11\,762)/2 \simeq 70$ million edges. Triangulating the constraint graph of gadm2 results in a total of 596 574 edges, which is $\sim 99.99\%$ less than completing the graph that results in $(276\,729^2 - 276\,729)/2 \simeq 40$ billion edges. In fact, adm1 consists of some big cycles that result in the addition of plenty fill edges in its initial graph, as opposed to gadm2. To obtain a better understanding on the effect of triangulation, Figure 5.13 depicts the adjacency matrix of the initial constraint graph of adm1 (Figure 5.13a) and the matrix of the



(a) degree distribution for `adm1`

(b) degree distribution for `gadm2`

Figure 5.12: Evidence of the power law node degree distribution of the real datasets considered

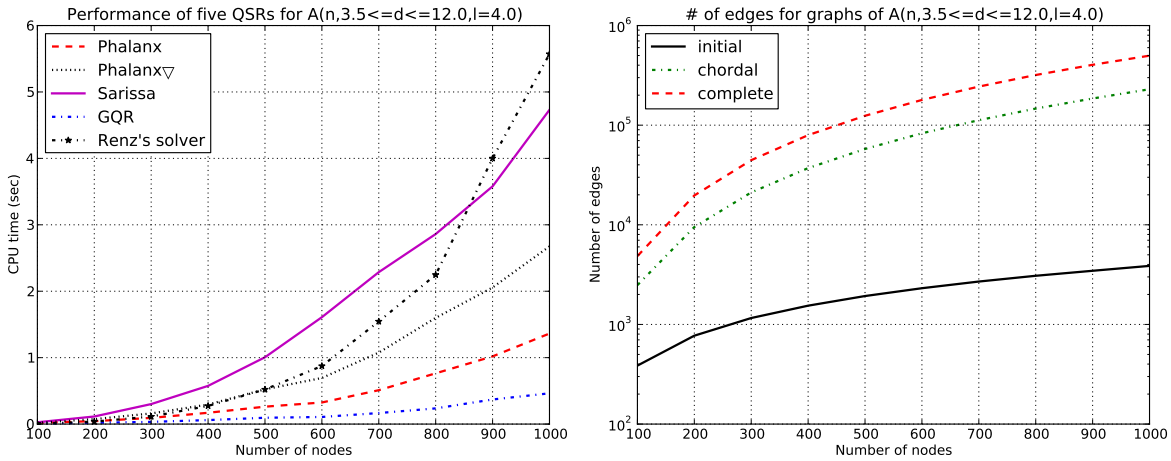


(a) initial constraint graph of `adm1`

(b) chordal constraint graph of `adm1`

Figure 5.13: Matrix representations of different graph configurations of `adm1`

chordal constraint graph of `adm1` (Figure 5.13b). We used gnuplot to create matrices of 800×800 pixels. For a given network of 8000 nodes for example, each pixel corresponds to a submatrix of $(8000/800) \times (8000/800)$ size, which results by partitioning the initial matrix of $(8000) \times (8000)$ size with a $(8000/800)$ step for each row and each collumn. For a particular pixel, color white indicates the complete absense of an edge in the $(8000/800) \times (8000/800)$ submatrix, and color black a full of edges submatrix. In-between shades, such as light and dark grey, indicate, more or less, how many edges the corresponding submatrix has. For a complete network, its adjacency matrix would be completely black. In this particular example of `adm1`, we can see exactly how a triangulation retains the sparseness of the initial graph (as opposed to a completion of that



(a) performance for regular networks

(b) edges considered for different graph configurations of regular networks

Figure 5.14: Experiment with random regular networks

same graph).

It is worth noting that `adm1` exhibits a strong locality of reference property [Liakos *et al.*, 2014a], i.e., adjacent nodes are close to each other with respect to their labelling and, thus, form a dense part around the main diagonal of the graph's adjacency matrix.

Evaluation with Arbitrary QCNs

We evaluate the performance of Sarissa, Renz's solver [Renz and Nebel, 2001], GQR (release 1500) [Gantner *et al.*, 2008], Phalanx, and Phalanx▽, with their best performing heuristics enabled.

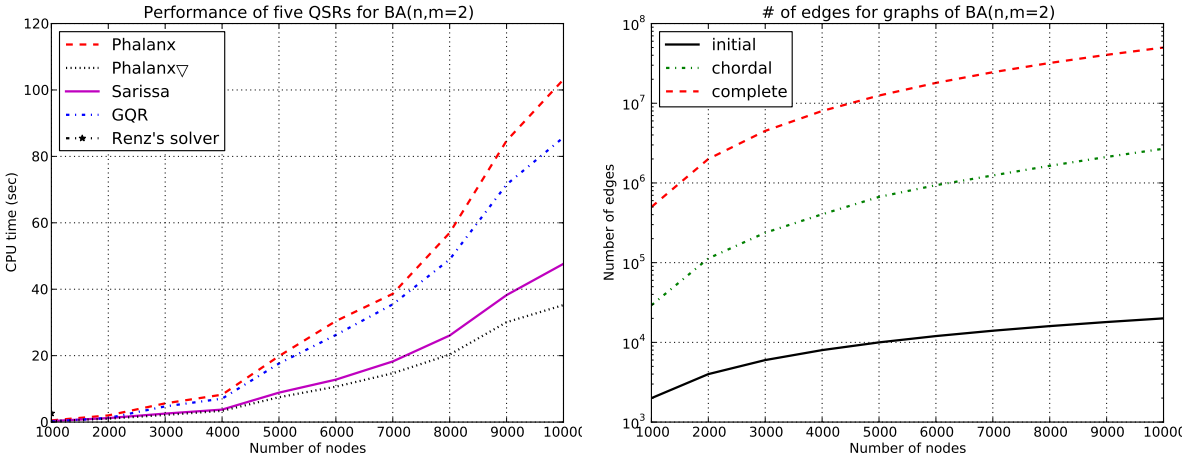
Technical Specifications The experimentation was carried out on a computer with an Intel Core 2 Quad Q9400 processor with a CPU frequency of 2.66 GHz, 8 GB RAM, and the Precise Pangolin x86_64 OS (Ubuntu Linux). Renz's solver and GQR were compiled with gcc/g++ 4.6.3. Sarissa, Phalanx, and Phalanx▽ were run with PyPy 1.9[22], which implements Python 2. Only one of the CPU cores was used for the experiments.

Dataset and Measures We considered random datasets generated by two different models. In particular, random datasets consist of RCC-8 networks generated by the usual $\mathsf{A}(\mathsf{n},\mathsf{d},\mathsf{l})$ model [Renz and Nebel, 2001] and *large* RCC-8 networks generated by the $\mathsf{BA}(\mathsf{n},\mathsf{m})$ model. In short, model $\mathsf{A}(\mathsf{n},\mathsf{d},\mathsf{l})$ creates random regular networks (like the one depicted in Figure 5.8a at page 106) of size n, degree d, and an average number l of RCC-8 relations per edge, whereas model $\mathsf{BA}(\mathsf{n},\mathsf{m})$ creates random scale-free-like networks (like the one depicted in Figure 5.8b at page 106) of size n and a preferential attachment value m. For model $\mathsf{BA}(\mathsf{n},\mathsf{m})$ the average number of base RCC-8 relations per edge defaults to $|\mathsf{B}|/2$, where B is the set of base relations of RCC-8. The average CPU time per network was used as a comparison measure, as well as the number of edges considered by each approach for different graph configurations of networks.

[22] `http://pypy.org/`

Results For model A(n, d, l) we considered network sizes between 100 and 1000 nodes with a 100 step and $l = 4$ ($= |\mathsf{B}|/2$) relations per edge. For each size series we created 270 networks that span over a degree d between 3.5 and 12.0 with a 0.5 step, i.e., 15 network instances were generated for each degree. Regarding model A(n, d, l), network instances with an average degree d between values 8.0 and 11.0 lie within the *phase transition* region, where it is equally possible for networks to be consistent or inconsistent and are, thus, harder to solve, as pointed out in [Renz and Nebel, 2001]. The results are shown in Figure 5.14a. GQR clearly outperforms all other reasoners with Phalanx coming close 2^{nd} and Renz's solver last. In the final step, when networks of 1000 nodes are considered, Renz's solver decides the networks using an average time of 5.571 sec per network, GQR using 0.463 sec per network, Phalanx using 1.363 sec per network, Phalanx$_\bigtriangledown$ using 2.675 sec per network, and Sarissa using 4.731 sec per network. In the particular case of Sarissa and Phalanx$_\bigtriangledown$ that use chordal graphs we note that they pay an extra cost for calculating the triangles of constraints for each appliance of partial $\diamond$-consistency as these are not precomputated and stored in advance for memory efficiency. As noted in Section 5.2.1, a graph G in the PWC algorithm is represented as a list, i.e., a sequence of index-value pairs where an index corresponds to some variable (vertex of the graph G) and its associated value corresponds to the set of its neighbours in the graph G. When a pair of variables $(v_i, v_j) \in V$ is processed, the intersection between the sets of neighbours of vertices v_i and v_j respectively is computed so as to obtain all triangles of variables the pair of variables (v_i, v_j) is a part of. In the average case, the time complexity of this operation is $\min(\{O(|N(v_i)|), O(|N(v_j)|)\})$ for any robust implementation. It should be clear that the greater the maximum vertex degree of graph G, the greater the time complexity of the previous operation will be. Sarissa also pays an additional cost for not being able to access or update relational values in constant worst case time as it does not use a matrix. It is a fact that random regular networks are not triangulated very efficiently with our approach, which results in dense chordal graphs in most of the cases. This fact is depicted in Figure 5.14b where one can clearly see that the total number of edges in a triangulated constraint graph of an initial random regular network, is very close to the total number of edges in the completion of the constraint graph of that network. To be even more precise, for the random regular network instances considered, their respective chordal constraint graphs contained on average only 53% less edges than the completions of those graphs.

For model BA(n, m) we considered 30 networks for each size between 1000 and 10000 nodes with a 1000 step and a preferential attachment value of $m = 2$. Regarding model BA(n, m), and for the network sizes considered, we found that the *phase transition* region is defined for the specific value of $m = 2$. A value of $m = 1$ yields networks that do not have any triangles of constraints, and, therefore, are always consistent, and a value of $m = 3$ yields almost exclusively inconsistent networks as the network size increases, that are very easy to decide. In particular, we experimented with 50 network instances of 10000 nodes and a preferential attachment value of $m = 3$, and none of them was consistent. The results for the network instances with a preferential attachment value of $m = 2$ are shown in Figure 5.15a. Sarissa and Phalanx$_\bigtriangledown$ outperform all other reasoners by a large scale, and Renz's solver was able to solve only the networks of 1000 nodes (slower than all others) as it quickly hit the memory limit in our computer due to many recursive calls (leading to storing many copies of its constraint matrix). In the final step, when networks of 10000 nodes are considered, GQR decides the networks using an average time of 85.727 sec per network, Phalanx using 103.159 sec per network, Phalanx$_\bigtriangledown$ using 35.241 sec per network, and Sarissa using 47.653 sec per network. Figure 5.15b depicts the number of edges that we consider when using a chordal completion of the constraint graph corresponding to a random scale-free-like network, rather than the (full) completion of that graph. In particular, for the random scale-free-like networks instances considered, their respective chordal constraint graphs



(a) performance for scale-free-like networks

(b) edges considered for different graph configurations of scale-free-like networks

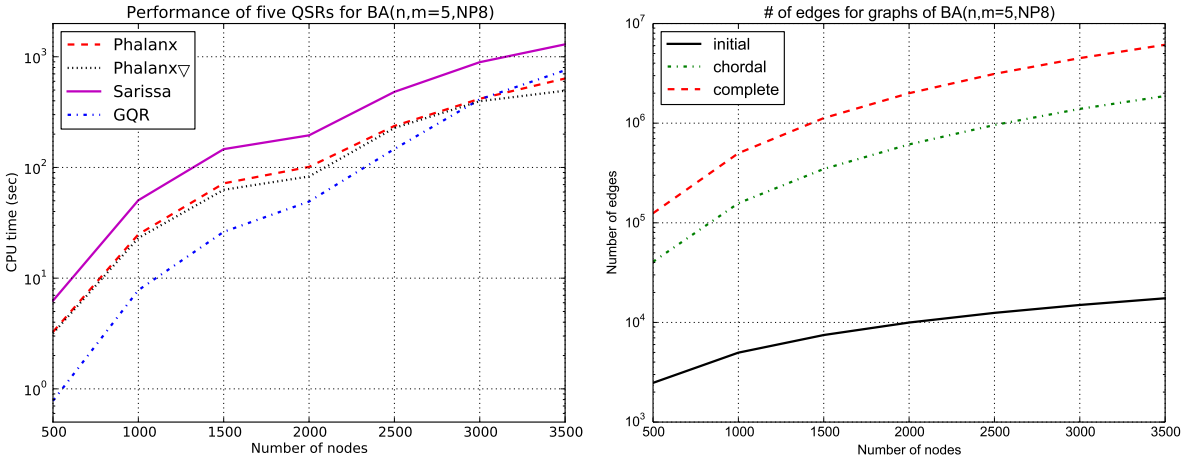
Figure 5.15: Experiment with random scale-free-like networks

contained on average around 95% less edges than the completions of those graphs. This is a huge improvement when compared with the 53% in the case of random regular networks. We note that Sarissa is still burdened with the additional cost of not being able to access or update relational values in constant worst case time.

We tried to push our implementations even further, and extended model BA(n, m) to generate random scale-free-like instances using the $\mathcal{NP}_8$ set of RCC-8 relations [Renz and Nebel, 2001]. This set contains relations that do not belong to any maximal tractable subclass of RCC-8, which significantly increases the search space of the problem. In particular, when using all RCC-8 relations and the maximal tractable subclass $\hat{\mathcal{H}}_8$ as a split set, the average branching factor is 1.4375. For a network instance of n nodes this translates to a $O(1.4375^{(n^2-n)/2})$ search space, as the n nodes will correspond to $O(n^2)$ unassigned constraints.[23] On the other hand, when using only $\mathcal{NP}_8$ relations, and again $\hat{\mathcal{H}}_8$ as a split set, the average branching factor is 2.053, which given a network instance of n nodes translates to a $O(2.05^{(n^2-n)/2})$ search space. As such, the search space in the latter case is $O((2.053/1.434)^{(n^2-n)/2})$ bigger than in the first case, which seriously impacts evaluation. We call our new extended model BA(n, m,$\mathcal{NP}_8$).

For model BA(n, m,$\mathcal{NP}_8$) we considered 10 networks for each size between 500 and 3500 with a 500 step and a preferential attachment value of $m = 5$. Regarding model BA(n, m,$\mathcal{NP}_8$), and for the network sizes considered, we found that the *phase transition* region is defined for the specific value of $m = 5$. In fact, for that value of m, checking the consistency of the network instances is very time consuming, and, thus, we did not push the network sizes to an even larger number. In any case, our dataset was sufficient to obtain a trend for the reasoners involved, and networks of 3500 nodes can be qualified as large, especially when $\mathcal{NP}_8$ relations are considered. The results for the hard network instances with a preferential attachment value of $m = 5$ are shown in Figure 5.16a. Initially we considered five reasoners, the four that are mentioned in the legend and Renz's solver. However, Renz's solver lacks the weight over learned weights dynamic heuristic for variable selection and could not keep up with the rest of the reasoners even for the smallest networks of 500 nodes. Further, it hit the memory limit. Sarissa has the

[23] However, this is an overestimation, as using the $\diamond$-consistency algorithm as a preprocessing and forward checking step in a backtracking algorithm significantly reduces search space [Ladkin and Reinefeld, 1997].



(a) performance for hard scale-free-like networks

(b) edges considered for different graph configurations of hard scale-free-like networks

Figure 5.16: Experiment with hard random scale-free-like networks

worst performance of all reasoners, deciding the networks of 3500 nodes using an average time of 1294.27 sec per network. This was expected, since the increased search space for the hard instances leads to massively more table lookups, wearing down the hash table based adjacency list. However, the performance of Sarissa converges to the performance of GQR, which was the biggest surprise in this experiment. Due to its more advanced heuristic mechanics (restarts and nogood recording), we expected GQR to set the bar for this experiment. Judging by the experiment with normal random scale-free-like networks, network sizes of 3500 nodes were not really that of a challenge for GQR (Figure 5.15a). In particular, GQR starts off very well, but its performance deteriotates very fast and is outrunned by both Phalanx▽ and Phalanx by the time when networks of 3000 nodes are considered. GQR should at least perform better than Phalanx as it did for normal random scale-free-like networks (Figure 5.15a), since both implementations use a matrix and GQR has more advanced heuristics.

In the final step, when networks of 3500 nodes are considered, GQR decides the networks using an average time of 754.98 sec per network, Phalanx using 636.32 sec per network, Phalanx▽ using 493.37 sec per network, and as already mentioned Sarissa using 1294.27 sec per network. Figure 5.16b depicts the number of edges that we consider when using a chordal completion of the constraint graph corresponding to a hard random scale-free-like network, rather than the completion of that graph. In particular, for the hard random scale-free-like networks instances considered, their respective chordal constraint graphs contained on average around 71% less edges than the completions of those graphs.

To the best of our knowledge, the datasets used in this thesis with respect to checking the satisfiability of QCNs are the biggest ones to date of all others that exist in the literature.

A compact SAT encoding for RCC-8

In [Huang *et al.*, 2013] an implementation based on a compact SAT encoding is proposed for RCC-8. The authors introduce the notion of a *dtree* (decomposition tree), which, in fact, can be seen as a chordal graph, since a chordal graph yields a natural decomposition tree of its cliques [Golumbic, 2004]. Further, the authors use a dedicated graph partitioning software with

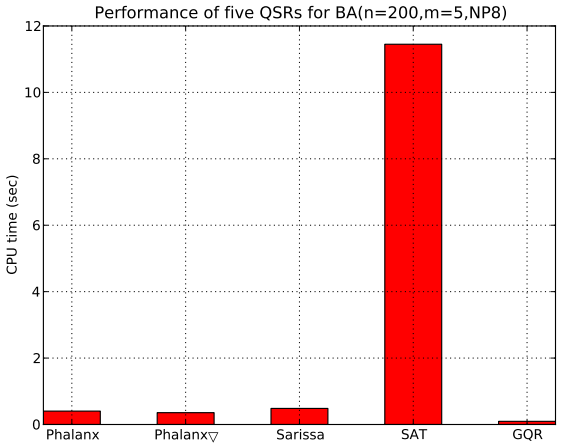


Figure 5.17: Experiment with hard random scale-free-like networks and a SAT implementation

heuristics to partition the network, viz., hMETIS[24], and do not rely in a linear time triangulation of its underlying graph. The implementation is significantly outrunned by GQR under the older release 1418 by the time networks of 150 nodes are considered (see Figure 13c in [Huang *et al.*, 2013, chapt. 6.4]). Here, we considered hard network instances of 3500 nodes, and were able to perform better than GQR under its newest release, 1500. The compact SAT encoding presented in [Huang *et al.*, 2013] becomes too large when network sizes increase beyond a few hundred nodes [Huang *et al.*, 2013, chapt. 6.4].

Nevertheless, we decided to run a small experiment with the SAT-based implementation of [Huang *et al.*, 2013], the GQR reasoner under its newest release 1500, and our own reasoners Phalanx, Phalanx$_{\bigtriangledown}$, and Sarissa. The benchmark that was used in [Huang *et al.*, 2013] consisted of RCC-8 networks generated by the $\mathsf{H}(\mathsf{n},\mathsf{d},\mathsf{l})$ model [Renz and Nebel, 2001]. Here, we considered RCC-8 networks generated by the $\mathsf{BA}(\mathsf{n},\mathsf{m},\mathcal{NP}_8)$ model.[25] In particular, we created 10 networks with a size of 200 nodes and a preferential attachment value of $m = 5$. The size of 200 nodes might seem too small, but some of the instances consumed over 2 GB when processed by the SAT-based implementation. In any case, the size of 200 nodes is sufficient to draw a conclusion. The results are shown in Figure 5.17. For a given hard RCC-8 network, Phalanx, Phalanx$_{\bigtriangledown}$, and Sarissa all decide its consistency in approximately 0.4 sec, and GQR in around 0.1 sec. In fact, most of the time consumed by our own reasoners was for initializing data structures and allowing the JIT compiler of PyPy to kick in. On the other hand, the SAT-based implementation needed nearly 12 sec to decide the consistency of a given RCC-8 network, on average. It follows that state of the art SAT encodings are not able to tackle large scale-free-like RCC-8 networks.

At this point we conclude our experimental evaluation. We have presented a complete set of experiments with several network generation models, and several graphs displaying the amount of edges considered in each approach and the effect of the triangulations. Most importantly, we have introduced a new model for generating RCC-8 networks, viz., $\mathsf{BA}(\mathsf{n},\mathsf{m}[,\mathcal{NP}_8])$, and have identified the phase transition region for both normal and hard network instances of that model. The techniques implemented in Sarissa can be easily adopted by any reasoner, such as GQR.

[24] `http://glaros.dtc.umn.edu/gkhome/metis/hmetis/overview`

[25] We used this particular model because a SAT-based implementation deals quite well with hard instances when compared with native reasoners [Westphal and Wölfl, 2009]. We verified this by conducting some experiments with networks generated by the $\mathsf{BA}(\mathsf{n},\mathsf{m})$ model for which the SAT-based implementation considered here had indeed even poorer performance when compared with the rest of the reasoners.

The outcome of our evaluation should be interpreted more as a comparison between different techniques, and not just as a comparison between differently coded reasoners per se.

5.5 Efficient Algorithms for the Minimal Labeling Problem of QCNs

In the case where a QCN is defined over a subclass of relations for which $\diamond$-consistency alone is able to guarantee minimality and partial $\diamond$-consistency is able to maintain the same pruning capacity as $\diamond$-consistency with respect to the common edges between a given graph and the completion of the constraint graph of that QCN, and since algorithm PWC is able to enforce partial $\diamond$-consistency on a given QCN of a relation algebra due to Proposition 24 (at page 97), we have the following result:

Proposition 34 *Let $\mathcal{A} \in 2^{\mathsf{B}}$ be a subclass of relations of a qualitative constraint language that is a relation algebra, over which not trivially inconsistent and $\diamond$-consistent* QCN*s are minimal. Then, given a satisfiable* QCN *$\mathcal{N} = (V, C)$ defined over $\mathcal{A}$ and a graph $G = (V, E)$ such that $\forall\{v, v'\} \in E$ we have that $\diamond(\mathcal{N})[v, v'] = \overset{\diamond}{}_{G}(\mathcal{N})[v, v']$, algorithm* PWC *terminates and returns a* QCN *$\mathcal{N}' = (V, C')$ such that $\forall\{v, v'\} \in E$ we have that $C'(v, v')$ is minimal.*

Then, due to Propositions 22 (at page 94) and 34, and Theorem 2 (at page 35), we have the following result:

Proposition 35 *Let $\mathcal{N} = (V, C)$ be a satisfiable* QCN *defined over a distributive subclass of relations of a qualitative constraint language that is a relation algebra and for which every $\diamond$-consistent atomic* QCN *is satisfiable, and $G = (V, E)$ a chordal graph such that $G(\mathcal{N}) \subseteq G$. Then, algorithm* PWC *terminates and returns a* QCN *$\mathcal{N}' = (V, C')$ such that $\forall\{v, v'\} \in E$ we have that $C'(v, v')$ is minimal.*

Finally, due to Proposition 35, and Propositions 1 (at page 21) and 3 (at page 34), we have the following result:

Corollary 26 *Let $\mathcal{N} = (V, C)$ be a satisfiable* QCN *defined over a distributive subclass of relations of Point Algebra, Cardinal Direction Calculus, Interval Algebra, Block Algebra, or* RCC*-8. We have that algorithm* PWC *terminates and returns a* QCN *$\mathcal{N}' = (V, C')$ such that $\forall\{v, v'\} \in E$ we have that $C'(v, v')$ is minimal.*

In the case where a QCN is defined over a subclass of relations of a qualitative constraint language that has patchwork for not trivially inconsistent and $\diamond$-consistent QCNs defined over that subclass of relations, and since algorithm PWC is able to enforce partial $\diamond$-consistency on a given QCN of a relation algebra due to Proposition 24 (at page 97), we can devise an algorithm based on PWC that will extract the minimal relations of that QCN corresponding to edges of the graph that is given as input to PWC under the conditions specified in Proposition 19 (at page 90). Such an algorithm is presented in Algorithm 20, called PartialMinimize. We have the following result:

Proposition 36 *Let $\mathcal{A} \in 2^{\mathsf{B}}$ be a subclass of relations of a qualitative constraint language that is a relation algebra and has patchwork for not trivially inconsistent and $\diamond$-consistent* QCN*s defined over that subclass of relations. Then, given a* QCN *$\mathcal{N} = (V, C)$ defined over $\mathcal{A}$ and a chordal graph $G = (V, E)$ such that $G(\mathcal{N}) \subseteq G$, algorithm* PartialMinimize *terminates and returns a* QCN *$\mathcal{N}' = (V, C')$ such that $\forall\{v, v'\} \in E$ we have that $C'(v, v')$ is minimal.*

Algorithm 20: PartialMinimize($\mathcal{N}, G$)

in : A QCN $\mathcal{N} = (V, C)$ defined over a subclass of relations of a qualitative constraint language that is a relation algebra and has patchwork for not trivially inconsistent and $\diamond$-consistent QCNs defined over that subclass of relations, and a graph $G = (V, E)$.

output : A QCN $\mathcal{N}' = (V, C')$ such that $\forall \{v, v'\} \in E$ we have that $C'(v, v')$ is minimal.

```
1 begin
2     N' ← N;
3     foreach {v_i, v_j} ∈ E with 0 ≤ i ≤ j < |V| do
4         foreach b ∈ N'[v_i, v_j] do
5             if b ∉ PWC(N'_{[v_i,v_j]/{b}}, G)[v_i, v_j] then
6                 N[v_i, v_j] ← N[v_i, v_j] \ {b};
7                 N[v_j, v_i] ← N[v_j, v_i] \ {b^-1};
8     return N;
```

Then, due to Proposition 36 and Propositions 1 (at page 21) and 16 (at page 54), we have the following result:

Corollary 27 *Let $\mathcal{N} = (V, C)$ be a satisfiable* QCN *of Point Algebra, Cardinal Direction Calculus, Interval Algebra, Block Algebra, or* RCC-*8, defined over one of the classes $\mathcal{P}_{\mathsf{PA}}$, $\mathcal{P}_{\mathsf{CDC}}$, $\mathcal{H}_{\mathsf{IA}}$, $\mathcal{H}^n_{\mathsf{IA}}$, or $\hat{\mathcal{H}}_8$, $\mathcal{C}_8$, and $\mathcal{Q}_8$ respectively, and $G = (V, E)$ a chordal graph such that $G(\mathcal{N}) \subseteq G$. We have that algorithm* PartialMinimize *terminates and returns a* QCN $\mathcal{N}' = (V, C')$ *such that $\forall \{v, v'\} \in E$ we have that $C'(v, v')$ is minimal.*

Given a QCN $\mathcal{N} = (V, C)$ and a graph $G = (V, E)$, algorithm PartialMinimize runs in $O(\delta |E|^2 |\mathsf{B}|^2)$ time. PartialMinimize iterates a number of $O(|E|)$ constraints and calls algorithm PWC for each one of the $O(|\mathsf{B}|)$ base relations of a constraint; each such call requiring $O(\delta |E| |\mathsf{B}|)$ time.

Before we consider the case of arbitrary QCNs, we will introduce and study a new form of partial consistency for QCNs, called ${}^{\blacklozenge}_{G}$-consistency, where G is a graph on the set of variables V of the considered QCN. Intuitively, a QCN $\mathcal{N}$ on V is ${}^{\blacklozenge}_{G}$-consistent iff for every pair of variables (v, v') and every base relation $b \in \mathcal{N}[v, v']$, after instantiating $\mathcal{N}[v, v']$ with $\{b\}$ and computing the closure under ${}^{\diamond}_{G}$-consistency, $\mathcal{N}[v, v']$ is defined by $\{b\}$. Formally, ${}^{\blacklozenge}_{G}$-consistency of a QCN is defined as follows.

Definition 40 *Let $\mathcal{N} = (V, C)$ be a* QCN *and $G = (V, E)$ a graph. $\mathcal{N}$ is said to be ${}^{\blacklozenge}_{G}$-consistent iff $\forall \{v, v'\} \in E$ and $\forall b \in \mathcal{N}[v, v']$, $\{b\} = {}^{\diamond}_{G}(\mathcal{N}_{[v,v']/\{b\}})[v, v']$.*

If G is a complete graph, i.e., $G = K_V$, we can easily verify that ${}^{\blacklozenge}_{G}$-consistency corresponds to ${}^{\diamond}_{\mathsf{B}}$-consistency of the familly of ${}^{\diamond}_{f}$-consistencies studied in [Condotta and Lecoutre, 2010]. Interestingly, ${}^{\blacklozenge}_{G}$-consistency can also be seen as a partial singleton arc consistency (SAC) [Debruyne and Bessière, 1997] for QCNs. Given a QCN $\mathcal{N} = (V, C)$ and a graph $G = (V, E)$, for every $b \in \mathsf{B}$ and every $v, v' \in V$, we will say that b is ${}^{\blacklozenge}_{G}$-consistent for $\mathcal{N}[v, v']$ iff $\{b\} = {}^{\diamond}_{G}(\mathcal{N}_{[v,v']/\{b\}})[v, v']$. We have the following proposition:

Proposition 37 *Let $\mathcal{N} = (V, C)$ and $\mathcal{N}' = (V, C')$ be two* QCN*s such that $\mathcal{N} \subseteq \mathcal{N}'$, and $G = (V, E)$ a graph. For every $b \in \mathsf{B}$ and every $v, v' \in V$, if b is ${}^{\blacklozenge}_{G}$-consistent for $\mathcal{N}[v, v']$ then b is ${}^{\blacklozenge}_{G}$-consistent for $\mathcal{N}'[v, v']$.*

Algorithm 21: Make$^{\blacklozenge}_{G}(\mathcal{N},G)$

in : A QCN $\mathcal{N} = (V, C)$ of a qualitative constraint language that is a relation algebra, and a graph $G = (V, E)$.
output : $^{\blacklozenge}_{G}(\mathcal{N})$.

```
1  begin
2      repeat
3          N' ← N;
4          foreach {v_i, v_j} ∈ E with 0 ≤ i ≤ j < |V| do
5              foreach b ∈ N'[v_i, v_j] do
6                  if b ∉ PWC(N'_{[v_i,v_j]/{b}}, G)[v_i, v_j] then
7                      N[v_i, v_j] ← N[v_i, v_j] \ {b};
8                      N[v_j, v_i] ← N[v_j, v_i] \ {b^-1};
9      until N = N';
10     return N;
```

Proof. Suppose that $^{\diamond}_{G}(\mathcal{N}_{[v,v']/\{b\}})[v, v'] = \{b\}$. Since we know that $\mathcal{N} \subseteq \mathcal{N}'$, we have that $\mathcal{N}_{[v,v']/\{b\}} \subseteq \mathcal{N}'_{[v,v']/\{b\}}$. Therefore, by monotonicity of $^{\diamond}_{G}$ we have that $^{\diamond}_{G}(\mathcal{N}_{[v,v']/\{b\}}) \subseteq {}^{\diamond}_{G}(\mathcal{N}'_{[v,v']/\{b\}})$. Since $^{\diamond}_{G}(\mathcal{N}_{[v,v']/\{b\}})[v, v'] = \{b\}$ and $^{\diamond}_{G}(\mathcal{N}'_{[v,v']/\{b\}})[v, v'] \subseteq \{b\}$, we can conclude that $^{\diamond}_{G}(\mathcal{N}'_{[v,v']/\{b\}})[v, v'] = \{b\}$. ⊣

Next, we prove the following properties to show that there exists a closure under $^{\blacklozenge}_{G}$-consistency as with $^{\diamond}_{G}$-consistency:

Proposition 38 *Let V be a set of variables and $G = (V, E)$ a graph. We have: (1) for any* QCN*s $\mathcal{N}_1$ and $\mathcal{N}_2$ defined on V, if $\mathcal{N}_1$ and $\mathcal{N}_2$ are $^{\blacklozenge}_{G}$-consistent, then $\mathcal{N}_1 + \mathcal{N}_2$ is $^{\blacklozenge}_{G}$-consistent, and (2) every scenario $\mathcal{S}$ defined on V is a $^{\blacklozenge}_{G}$-consistent* QCN.

Proof. (1) Let $v, v' \in V$ and $b \in (\mathcal{N}_1 + \mathcal{N}_2)[v, v']$. Suppose that $\mathcal{N}_1$ and $\mathcal{N}_2$ are $^{\blacklozenge}_{G}$-consistent. It is clear that $b \in \mathcal{N}_1[v, v']$ and/or $b \in \mathcal{N}_2[v, v']$. Suppose that $b \in \mathcal{N}_1[v, v']$ (the other case is similar). Since $\mathcal{N}_1$ is $^{\blacklozenge}_{G}$-consistent, we have that b is $^{\blacklozenge}_{G}$-consistent for $\mathcal{N}_1[v, v']$. Therefore, since $\mathcal{N}_1 \subseteq \mathcal{N}_1 + \mathcal{N}_2$, from Proposition 37 we have that b is $^{\blacklozenge}_{G}$-consistent for $(\mathcal{N}_1 + \mathcal{N}_2)[v, v']$. (2) Every scenario $\mathcal{S}$ defined on V is $\diamond$-consistent and, hence, $^{\diamond}_{G}$-consistent. Moreover, $\forall v, v' \in V$ and $\forall b \in \mathcal{S}[v, v']$, $\mathcal{S}[v, v'] = \{b\}$. Thus, $\forall v, v' \in V$ and $b \in \mathcal{S}[v, v']$ we have that $^{\diamond}_{G}(\mathcal{S}_{[v,v']/\{b\}})[v, v'] = {}^{\diamond}_{G}(\mathcal{S})[v, v'] = \mathcal{S}[v, v'] = \{b\}$. ⊣

From Property (1) of the aforementioned proposition we can assert that for a QCN $\mathcal{N} = (V, C)$ and a graph $G = (V, E)$ there exists a unique largest $^{\blacklozenge}_{G}$-consistent sub-QCN of $\mathcal{N}$, i.e., a closure of $\mathcal{N}$ under $^{\blacklozenge}_{G}$-consistency. By denoting this closure by $^{\blacklozenge}_{G}(\mathcal{N})$, we have that $^{\blacklozenge}_{G}(\mathcal{N}) = \{\mathcal{N}' \mid \mathcal{N}' \subseteq \mathcal{N}$ and $\mathcal{N}'$ is $^{\blacklozenge}_{G}$-consistent$\}$. Further, we have that $^{\blacklozenge}_{G}(\mathcal{N})$ is equivalent to $\mathcal{N}$ since from Property (2) of the previous proposition we know that every scenario $\mathcal{S}$ of $\mathcal{N}$ is a $^{\blacklozenge}_{G}(\mathcal{N})$ sub-QCN of $\mathcal{N}$.

A naive algorithm for enforing $^{\blacklozenge}_{G}$-consistency is presented in Algorithm 21 and it is called Make$^{\blacklozenge}_{G}$. Given a QCN $\mathcal{N} = (V, C)$ and a graph $G = (V, E)$, algorithm Make$^{\blacklozenge}_{G}$ allows computing $^{\blacklozenge}_{G}(\mathcal{N})$ with a worst-case time complexity of $O(\delta|E|^3|\mathsf{B}|^3)$, where δ is the maximum vertex degree of graph G.

We can prove the following result:

Proposition 39 *Let $\mathcal{N} = (V, C)$ be a* QCN *and $G = (V, E)$ a graph. If $\mathcal{N}$ is $^{\blacklozenge}_{G}$-consistent then $\mathcal{N}$ is $^{\diamond}_{G}$-consistent.*

Proof. Suppose that $\mathcal{N}$ is $\overset{\blacklozenge}{}_G$-consistent and let (v, v''), (v, v'), $(v', v'') \in E$. We will show that $\mathcal{N}[v, v''] \subseteq \mathcal{N}[v, v'] \diamond \mathcal{N}[v', v'']$. Since $\mathcal{N}$ is $\overset{\blacklozenge}{}_G$-consistent, we have that $\forall b \in \mathcal{N}[v, v'']$, $\{b\} = \overset{\diamond}{}_G(\mathcal{N}_{[v,v'']/\{b\}})[v, v'']$. Therefore, $\forall b \in \mathcal{N}[v, v'']$, $b \in \mathcal{N}[v, v'] \diamond \mathcal{N}[v', v'']$. Hence, $\mathcal{N}[v, v''] \subseteq \mathcal{N}[v, v'] \diamond \mathcal{N}[v', v'']$. ⊣

Note that, in the general case, a $\overset{\diamond}{}_G$-consistent QCN is not necessary $\overset{\blacklozenge}{}_G$-consistent.

Now, we show that for QCNs defined over a subclass $\mathcal{A}$ of relations of a qualitative constraint language that has patchwork for not trivially inconsistent and $\diamond$-consistent QCNs defined over that subclass of relations, enforcing $\overset{\blacklozenge}{}_G$-consistency with respect to a chordal graph $G(\mathcal{N}) \subseteq G$ ensures the feasibility of the base relations belonging to the constraints of $\mathcal{N}$ that correspond to edges of G.

Proposition 40 *Let $\mathcal{A} \subseteq 2^{\mathsf{B}}$ be a subclass of relations of a qualitative constraint language that has patchwork for not trivially inconsistent and $\diamond$-consistent* QCN*s defined over that subclass of relations, $\mathcal{N} = (V, C)$ a not trivially inconsistent* QCN *defined over $\mathcal{A}$, and $G = (V, E)$ a chordal graph such that $G(\mathcal{N}) \subseteq G$. We have:*

(1) for every $(v, v') \in E$ and every $b \in \mathcal{N}[v, v']$, b is $\overset{\blacklozenge}{}_G$-consistent for $\mathcal{N}[v, v']$ iff $b \in \mathcal{N}_{\mathsf{min}}[v, v']$;

(2) for every $(v, v') \in E$, $\mathcal{N}_{\mathsf{min}}[v, v'] = \overset{\blacklozenge}{}_G(\mathcal{N})[v, v']$.

Proof. (1) Consider a base relation b which is $\overset{\blacklozenge}{}_G$-consistent for $\mathcal{N}[v, v']$ with $(v, v') \in E$. Let $\mathcal{N}'$ be the QCN defined by $\mathcal{N}' = \overset{\diamond}{}_G(\mathcal{N}_{[v,v']/\{b\}})$. Let us suppose that $\mathcal{N}'$ is trivially inconsistent. Since $\mathcal{N}$ is not trivially inconsistent, G is chordal and $\mathcal{N}'$ is $\overset{\diamond}{}_G$-consistent, we can show that $\mathcal{N}'[v, v'] = \emptyset$. There is a contradiction. Therefore, $\mathcal{N}'$ is not trivially inconsistent. Moreover, as $\mathcal{N}_{[v,v']/\{b\}}$ is defined over $\mathcal{A}$, we know that $\mathcal{N}'$ is also defined over $\mathcal{A}$. Consequently, from Proposition 19 (at page 90) we can assert that $\mathcal{N}'$ admits a scenario $\mathcal{S}$. $\mathcal{S}$ is also a scenario of $\mathcal{N}$ since $\mathcal{N}' \subseteq \mathcal{N}$. Since $\mathcal{N}'[v, v'] = \{b\}$, we can infer that $\mathcal{S}[v, v'] = \{b\}$. We can conclude that $b \in \mathcal{N}_{\mathsf{min}}[v, v']$. Now, consider $b \in \mathcal{N}_{\mathsf{min}}[v, v']$ with $(v, v') \in E$. There exists a scenario $\mathcal{S}$ of $\mathcal{N}$ such that $\mathcal{S}[v, v'] = \{b\}$. From Proposition 38 we have that $\mathcal{S}$ is $\overset{\blacklozenge}{}_G$-consistent. Therefore, b is $\overset{\blacklozenge}{}_G$-consistent for $\mathcal{S}[v, v']$. From Proposition 37 we can conclude that b is $\overset{\blacklozenge}{}_G$-consistent for $\mathcal{N}[v, v']$ as $\mathcal{S} \subseteq \mathcal{N}$. Next, (2) can be established directly from (1) and the fact that $\overset{\blacklozenge}{}_G(\mathcal{N})$ is an equivalent QCN of $\mathcal{N}$ for which every base relation $b \in \overset{\blacklozenge}{}_G(\mathcal{N})[v, v']$ with $(v, v') \in E$ is $\overset{\blacklozenge}{}_G$-consistent for $\mathcal{N}[v, v']$. ⊣

From the aforementioned proposition, we can notice that for a not trivially inconsistent QCN $\mathcal{N}$ defined over a subclass $\mathcal{A}$ of relations of a qualitative constraint language that has patchwork for not trivially inconsistent and $\diamond$-consistent QCNs defined over that subclass of relations and for a chordal graph G such that $G(\mathcal{N}) \subseteq G$, we can compute $\overset{\blacklozenge}{}_G(\mathcal{N})$ using algorithm $\mathsf{Make}^{\blacklozenge}_G$ whitout its outer loop. In fact, in this particular case, algorithm $\mathsf{Make}^{\blacklozenge}_G$ will fall back to algorithm PartialMinimize that we presented earlier, which has a worst-case runtime of $O(\delta|E|^2|\mathsf{B}|^2)$.

Before considering the next result, note that given two QCNs $\mathcal{N}$ and $\mathcal{N}'$ defined on V and a graph $G = (V, E)$, $\mathcal{N}_{G/\mathcal{N}'}$ denotes the QCN $\mathcal{N}'' = (V, C'')$ defined by $\mathcal{N}''[v, v'] = \mathcal{N}'[v, v']$ if $(v, v') \in E$, and $\mathcal{N}''[v, v'] = \mathcal{N}[v, v']$ otherwise. We have the following result which will be useful in the sequel:

Proposition 41 *Let $\mathcal{N}$, $\mathcal{N}'$, $\mathcal{N}''$ be three* QCN*s defined on V, $\mathcal{A}$ a subclass of relations of a qualitative constraint language that has patchwork for not trivially inconsistent and $\diamond$-consistent* QCN*s defined over that subclass of relations, and G a chordal graph such that: $G(\mathcal{N}) \subseteq G$, $\mathcal{N}'$ is an equivalent sub-*QCN *of $\mathcal{N}$, $\mathcal{N}''$ is a not trivially inconsistent and $\overset{\diamond}{}_G$-consistent sub-*QCN *of $\mathcal{N}'$ with $\mathcal{A}(\mathcal{N}'') \subseteq_G \mathcal{N}'$. By denoting $\mathcal{N}'_{G/\mathcal{A}(\mathcal{N}'')}$ by $\mathcal{N}^*$, we have:*

Algorithm 22: Minimize($\mathcal{N}$,$\mathcal{A}$)

in : $\mathcal{N} = (V, C)$ a QCN on 2^{B}, $\mathcal{A}$ a subclass of 2^{B}.
output : A sub-QCN of $\mathcal{N}$.

```
1  begin
       // Step 1: Initialization
2      N_I ← N; N* ← ⊥^V;
3      G = (V, E) ← Triangulation(G(N)); N ← WC(N);
4      if N = ⊥^V then
5          return ⊥^V;
       // Step 2: Minimization w.r.t. G
6      while not (N* =_G N) do
7          select {v, v'} ∈ E such that N*[v, v'] ⊂ N[v, v'];
8          r ← N[v, v'] \ N*[v, v']; r' ← N[v, v'];
9          N' ← ◇_G-SubQCN(N_{[v,v']/r}, G, A);
10         if N' = Null then
11             N[v, v'] ← r' \ r; N[v', v] ← (r' \ r)^{-1};
12         else
13             N'' ← N_{G/A(N')};
14             N* ← extractFeasible(N'',N*,G);
       // Step 3: End of the minimization
15     while N* ⊂ N do
16         select v, v' ∈ V such that N*[v, v'] ⊂ N[v, v'];
17         r ← N[v, v'] \ N*[v, v']; r' ← N[v, v'];
18         N[v, v'] ← r; N[v', v] ← r^{-1};
19         N' ← ◇_G-SubQCN(N, K_V, A);
20         if N' = Null then
21             N[v, v'] ← r' \ r; N[v', v] ← (r' \ r)^{-1};
22         else
23             N[v, v'] ← r'; N[v', v] ← (r')^{-1};
24             N* ← extractFeasible(A(N'),N*,K_V);
25     return N*;// Step 4: Return of the result
```

(1) $\mathcal{N}^*$ is a satisfiable QCN*;*

(2) each $\overset{\blacklozenge}{G}$-consistent base relation of $\mathcal{N}^*[v, v']$ with $(v, v') \in E$ is a feasible relation of $\mathcal{N}$;

(3) each $\overset{\blacklozenge}{K_V}$-consistent base relation of $\mathcal{N}^*$ is a feasible relation of $\mathcal{N}$.

Proof. (1) $\mathcal{N}_{G/\mathcal{N}^*}$ is clearly a not trivially inconsistent and $\overset{\diamond}{G}$-consistent QCN defined over $\mathcal{A}$ such that $G(\mathcal{N}_{G/\mathcal{N}^*}) \subseteq G$. From Proposition 19 (at page 90) we know that $\mathcal{N}_{G/\mathcal{N}^*}$ is satisfiable and admits a scenario $\mathcal{S}$. As $\mathcal{N}$ is equivalent to $\mathcal{N}'$, we have that $\mathcal{S}$ is a scenario of ${\mathcal{N}'}_{G/\mathcal{N}^*}$. By remarking that ${\mathcal{N}'}_{G/\mathcal{N}^*}$ and $\mathcal{N}^*$ are equal, we can affirm that $\mathcal{S}$ is a scenario of $\mathcal{N}^*$. (2) Consider a $\overset{\blacklozenge}{G}$-consistent base relation b of $\mathcal{N}^*[v, v']$ with $(v, v') \in E$. As $\mathcal{N}^* \subseteq \mathcal{N}_{G/\mathcal{N}^*}$, from Proposition 37 we have that b is a $\overset{\blacklozenge}{G}$-consistent base relation of $\mathcal{N}_{G/\mathcal{N}^*}[v, v']$. From Proposition 40, we can assert that b is a feasible base relation of $\mathcal{N}_{G/\mathcal{N}^*}$. We can conclude that b is also a feasible base relation of $\mathcal{N}$ since $\mathcal{N}_{G/\mathcal{N}^*} \subseteq \mathcal{N}$. (3) Each $\overset{\blacklozenge}{K_V}$-consistent base relation b of $\mathcal{N}^*$ is also a $\overset{\blacklozenge}{K_V}$-consistent

Algorithm 23: $\overset{\diamond}{\mathsf{G}}$-SubQCN($\mathcal{N}$,$G$,$\mathcal{A}$)

in : A QCN $\mathcal{N}$=(V, C) of a qualitative constraint language that is a relation algebra, a graph G=(V, E), a subclass $\mathcal{A}$.
output : Null, or a $\overset{\diamond}{G}$-consistent sub-QCN $\mathcal{N}'$ of $\mathcal{N}$ with $\mathcal{A}(\mathcal{N}') \subseteq_G \mathcal{N}$.

```
1  begin
2      N' ← PWC(N, G);
3      if ∃{v_i, v_j} ∈ E such that N'[v_i, v_j] = ∅ then
4          return Null;
5      if ∀v_i, v_j ∈ V we have that A(N'[v_i, v_j]) ⊆ N[v_i, v_j] then
6          return N';
7      choose a constraint N'[v_i, v_j] with v_i, v_j ∈ V such that A(N'[v_i, v_j]) ⊈ N[v_i, v_j];
8      split N'[v_i, v_j] into r_1, ..., r_k ∈ A: r_1 ∪ ... ∪ r_k = N'[v_i, v_j];
9      foreach r ∈ {r_l | 1 ≤ l ≤ k} do
10         N'[v_i, v_j] ← r; N'[v_j, v_i] ← r^-1;
11         result ← ◇_G-SubQCN(N',G,A);
12         if result ≠ Null then
13             return result;
14     return Null;
```

base relation of $\mathcal{N}$ since $\mathcal{N}^* \subseteq \mathcal{N}$. From Proposition 40 (by using K_V as a chordal graph) we can conclude that b is a feasible base relation of $\mathcal{N}$. ⊣

Finally, in order to solve the MLP for arbitrary QCNs, we present in this section the algorithm Minimize, presented in Algorithm 22. Minimize has two parameters, the first one being a QCN $\mathcal{N} = (V, C)$ for which we aim to derive the feasible base relations, and the second one being a subclass $\mathcal{A}$ of relations of a qualitative constraint language that has patchwork for not trivially inconsistent and $\diamond$-consistent QCNs defined over that subclass of relations. Minimize proceeds in an iterative manner that we explain as follows. In each iteration, a relation r is defined by a set of untreated base relations of a constraint $\mathcal{N}[v, v']$, followed by the derivation of a consistent sub-QCN $\mathcal{N}''$ of $\mathcal{N}$ defined over $\mathcal{A}$, for which $\mathcal{N}''[v, v']$ contains some base relations of r and, in particular, feasible relations of $\mathcal{N}$. In the case where such a sub-QCN does not exist, the base relations of r are not feasible. This process continues until all base relations of $\mathcal{N}$ are treated. The expected efficiency of function Minimize is due, on one hand, to the fact that several feasible (or unfeasible) base relations are derived in each iteration and, on the other hand, to the fact that searching for the subQCN $\mathcal{N}''$ and deriving feasible base relations can be made efficiently by applying partial consistencies $\overset{\diamond}{G}$-consistency and $\overset{\blacklozenge}{G}$-consistency for a given subclass $\mathcal{A}$ of relations of a qualitative constraint language that has patchwork for not trivially inconsistent and $\diamond$-consistent QCNs defined over that subclass of relations, where G is a chordal graph such that $G(\mathcal{N}) \subseteq G$.

Next, we consider the auxiliary function $\overset{\diamond}{\mathsf{G}}$-SubQCN. This function has the following three parameters: a QCN $\mathcal{N} = (V, C)$, a graph $G = (V, E)$, and a subclass $\mathcal{A} \subseteq 2^{\mathsf{B}}$. Note that this function is very similar to the one presented in Section 5.4 for solving the satisfiability problem of a QCN. Function $\overset{\diamond}{\mathsf{G}}$-SubQCN aims to derive and return a not trivially inconsistent and $\overset{\diamond}{G}$-consistent QCN $\mathcal{N}'$, such that $\mathcal{N}' \subseteq \mathcal{N}$, $\mathcal{A}(\mathcal{N}') \subseteq_G \mathcal{N}$, and $\mathcal{N}'[v, v'] = \mathcal{N}[v, v']$ $\forall\{v, v'\} \notin E$. In the case where such a QCN $\mathcal{N}'$ does not exist, the function returns Null. For this purpose, a backtrack search is realized by performing the closure under $\overset{\diamond}{G}$-consistency through a call to algorithm PWC for propagating constraints and ensuring that the result is $\overset{\diamond}{G}$-consistent. In each

Algorithm 24: extractFeasible($\mathcal{N}$,$\mathcal{N}'$,G)

in : Two QCNs $\mathcal{N}$ and $\mathcal{N}'$ on V of a qualitative constraint language that is a relation algebra, and a graph $G = (V, E)$.

output : $\mathcal{N}'$ in which are added $\overset{\blacklozenge}{}_{G}$-consistent base relations of $\mathcal{N}[v, v']$ with $\{v, v'\} \in E$ and $\overset{\diamond}{}_{K_V}$-consistent base relations of $\mathcal{N}[v, v']$ with $\{v, v'\} \notin E$.

```
1  begin
2      foreach v, v' ∈ V do
3          foreach b ∈ N[v, v'] \ N'[v, v'] do
4              if {v, v'} ∈ E and {b} = PWC(N_{[v,v']/{b}}, G)[v, v'] then
5                  N'[v, v'] ← N'[v, v'] ∪ {b};
6                  N'[v', v] ← N'[v', v] ∪ {b^-1};
7              else if {v, v'} ∉ E and {b} = WC(N_{[v,v']/{b}})[v, v'] then
8                  N'[v, v'] ← N'[v, v'] ∪ {b};
9                  N'[v', v] ← N'[v', v] ∪ {b^-1};
10     return N';
```

step, a constraint corresponding to an edge of G is selected and split into non-empty relations of $\mathcal{A}$. Then, this constraint is iteratively instantiated with each of these relations. The search continues through recursive calls of $\overset{\diamond}{\mathsf{G}}$-SubQCN.

We can assert the following result:

Proposition 42 *Let $\mathcal{N} = (V, C)$ be a* QCN*, $G = (V, E)$ a graph, and $\mathcal{A} \subseteq 2^{\mathsf{B}}$ a subclass of a qualitative constraint language that is a relation algebra. If $\mathcal{N}$ is satisfiable, then function $\overset{\diamond}{\mathsf{G}}$-SubQCN with $\mathcal{N}$, G, $\mathcal{A}$ as parameters returns a not trivially inconsistent and $\overset{\diamond}{}_{G}$-consistent* QCN *$\mathcal{N}'$, such that $\mathcal{A}(\mathcal{N}') \subseteq_G \mathcal{N}$ and $\mathcal{N}'[v, v'] = \mathcal{N}[v, v']$ $\forall \{v, v'\} \notin E$.*

Algorithm Minimize uses also another auxiliary function, called extractFeasible, that takes as parameters two QCNs $\mathcal{N}$ and $\mathcal{N}'$ defined on V and a graph $G = (V, E)$. This function returns $\mathcal{N}'$ augmented with $\overset{\blacklozenge}{}_{G}$-consistent base relations b belonging to $\mathcal{N}[v, v']$ with $\{v, v'\} \in E$, and $\overset{\blacklozenge}{}_{K_V}$-consistent base relations b belonging to $\mathcal{N}[v, v']$ with $\{v, v'\} \notin E$.

Now, we describe in detail algorithm Minimize. Minimize takes as parameters a QCN $\mathcal{N} = (V, C)$, for which we want to calculate the feasible base relations, and a subclass $\mathcal{A} \subseteq 2^{\mathsf{B}}$ of relations of a qualitative constraint language that has patchwork for not trivially inconsistent and $\diamond$-consistent QCNs defined over that subclass of relations. To begin with, Minimize comprises the following four successive steps: the initialization of different variables, the calculation of the feasible base relations corresponding to the edges in the set of edges E of a chordal graph, the minimization step by considering the constraints not corresponding to E, and finally, the return of the result.

The different variables initialized in the first step are: $\mathcal{N}_{\mathcal{I}}$, $\mathcal{N}^*$, and G. $\mathcal{N}_{\mathcal{I}}$ allows saving the initial state of the QCN $\mathcal{N}$ given as parameter. The QCN $\mathcal{N}^*$ accumulates the base relations of $\mathcal{N}$ detected as feasible during the treatment, and is therefore initialized to $\bot^V$. At the end of the treatment, $\mathcal{N}^*$ will be the minimal QCN of $\mathcal{N}_{\mathcal{I}}$. G is initialized to a chordal (triangulated) graph of $G(\mathcal{N})$. After these initializations, an optional preliminary treatment is performed (line 3) by calculating the closure under $\diamond$-consistency of $\mathcal{N}$. Its aim is to quickly eliminate some unfeasible base relations. In the case where $\mathcal{N}$ is detected as trivially inconsistent, $\mathcal{N}_{\mathcal{I}}$ is unsatisfiable and $\bot^V$ is returned (line 5). In the contrary case, we continue the treatment. We note here that

during this process, and until the end of the treatment, $\mathcal{N}$ is an equivalent sub-QCN of $\mathcal{N}_{\mathcal{I}}$. Only unfeasible base relations will be removed from $\mathcal{N}$.

In the second step (line 6), the constraints of $\mathcal{N}_{\mathcal{I}}$ corresponding to the edges in the set of edges E are treated until all base relations of these constraints are detected as feasible and accumulated into $\mathcal{N}^*$, or as unfeasible and removed from $\mathcal{N}$. For this purpose, a pair $\{v, v'\} \in E$ such that $\mathcal{N}[v, v']$ contains some non-marked feasible base relations is selected in line 7. These base relations correspond to the relation r (line 8). In line 9, a QCN $\mathcal{N}'$ is computed through a call to $\overset{\diamond}{\mathsf{G}}$-SubQCN with $\mathcal{N}[v, v']/r$, G, and $\mathcal{A}$ as parameters. Two cases must be considered: $\mathcal{N}' = \mathsf{Null}$, and $\mathcal{N}' \neq \mathsf{Null}$. In the case where $\mathcal{N}' = \mathsf{Null}$, we can affirm from Proposition 42 that $\mathcal{N}[v, v']/r$ is unsatisfiable and, therefore, the base relations of r are unfeasible for $\mathcal{N}$ and also for $\mathcal{N}_{\mathcal{I}}$, since these two QCNs are equivalent. The base relations of r can be removed from $\mathcal{N}$ (line 11). Now, suppose that $\mathcal{N}' \neq \mathsf{Null}$. In line 13, the QCN $\mathcal{N}_{G/\mathcal{A}(\mathcal{N}')}$ is saved into $\mathcal{N}''$. From Proposition 41 we know that each $\overset{\blacklozenge}{G}$-consistent base relation of $\mathcal{N}''[v'', v''']$ with $\{v'', v'''\} \in E$ and each $\overset{\blacklozenge}{K_V}$-consistent base relation of $\mathcal{N}''[v'', v''']$ with $\{v'', v'''\} \notin E$ are feasible relations of $\mathcal{N}_{\mathcal{I}}$ and can be added to $\mathcal{N}^*$ (line 14). Notice that since $\mathcal{N}''$ is satisfiable, *at least one* newly detected feasible base relation $b \in r$ is added to $\mathcal{N}^*[v, v']$. Following the second step, all base relations of the constraints $\mathcal{N}_{\mathcal{I}}[v, v']$ with $\{v, v'\} \in E$ have been treated. The third step of Minimize finishes the minimization of $\mathcal{N}_{\mathcal{I}}$ by considering the constraints $\mathcal{N}_{\mathcal{I}}[v, v']$ with $\{v, v'\} \notin E$. Notice that some base relations of these constraints have already been detected as feasible in the second step. As the third step is very similar to the second step we are not going to describe it in detail.

Following the third step we can then affirm due to Propositions 42 and 41 that the QCN $\mathcal{N}^*$ returned in line 25 corresponds to the minimal QCN of $\mathcal{N}_{\mathcal{I}}$. Thus, we can establish the following main result:

Theorem 32 *Given a satisfiable* QCN $\mathcal{N} = (V, C)$ *and a subclass* $\mathcal{A} \subseteq 2^{\mathsf{B}}$ *of relations of a qualitative constraint language that is a relation algebra and has patchwork for not trivially inconsistent and* $\diamond$*-consistent* QCN*s defined over that subclass of relations, we have that algorithm* Minimize*, with* $\mathcal{N}$ *and* $\mathcal{A}$ *as parameters, terminates and returns* $\mathcal{N}_{\mathsf{min}}$*.*

Then, due to Theorem 32 and Propositions 1 (at page 21) and 16 (at page 54), we have the following result:

Corollary 28 *Given a satisfiable* QCN $\mathcal{N} = (V, C)$ *of Point Algebra, Cardinal Direction Calculus, Interval Algebra, Block Algebra, or* RCC-*8 and one of the classes* $\mathcal{P}_{\mathsf{PA}}$*,* $\mathcal{P}_{\mathsf{CDC}}$*,* $\mathcal{H}_{\mathsf{IA}}$*,* $\mathcal{H}^n_{\mathsf{IA}}$*, or* $\hat{\mathcal{H}}_8$*,* $\mathcal{C}_8$*, and* $\mathcal{Q}_8$ *respectively, we have that algorithm* Minimize*, with* $\mathcal{N}$ *and* $\mathcal{A}$ *the given class as parameters, terminates and returns* $\mathcal{N}_{\mathsf{min}}$*.*

5.5.1 Experimental Evaluation

We evaluate an implementation of the approach that we presented in this section, namely, the Minimize algorithm, against two approaches that we describe as follows. The first approach, which we call NaiveMin, refers to a naive minimization algorithm which (after pruning some unfeasible base relations with the weak composition) consists of testing in an iterative way the feasibility of every base relation of each of the constraints of the QCN through the use of a method that solves the satisfiability problem (here the function $\overset{\diamond}{\mathsf{G}}$-SubQCN using the subclasses $\mathcal{H}_{\mathsf{IA}}$ or $\hat{\mathcal{H}}_8$ and the complete graph as parameters). As a first step, the constraint containing the base relation b to be tested is replaced with the singleton constraint $\{b\}$. Then, the satisfiability check with respect to the obtained QCN is executed in order to obtain the feasibility or the unfeasibility of b. The second approach is essentially the Minimize algorithm itself where a complete graph is used in all cases.

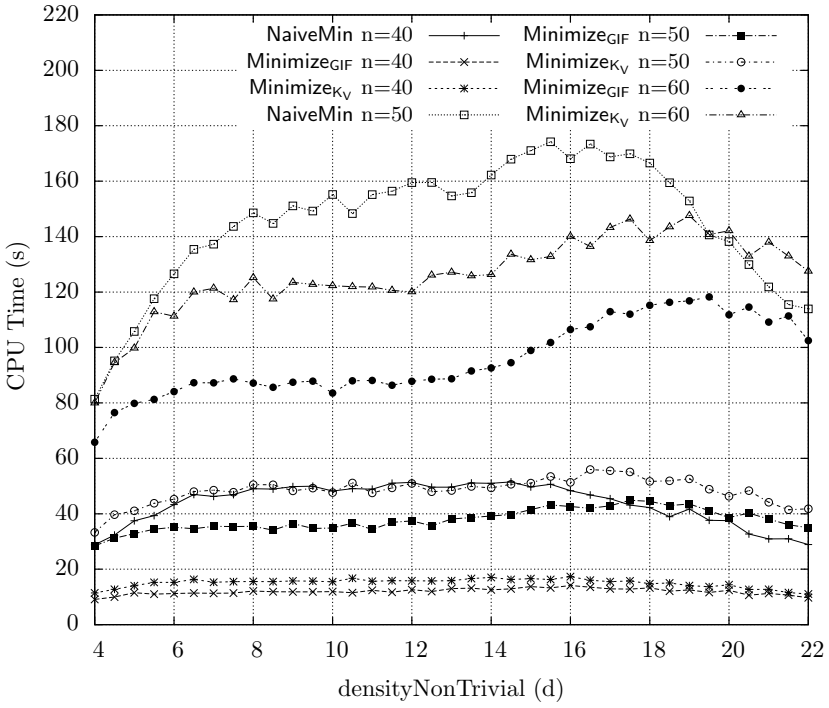


Figure 5.18: CPU time for series $\mathsf{S}(n, d, 6.5)$ of IA

Technical Specifications The experimentation was carried out on a computer with an Intel Core 2 Quad Q9400 processor with a CPU frequency of 2.66 GHz, 8 GB RAM, and the Precise Pangolin x86_64 OS (Ubuntu Linux). The implementation of the different functions was done using the *C* programming language, and the gcc/g++ 4.6.3 compiler was used for compilation. Regarding the implementation, and as we will deal with small-sized QCNs in the evaluation that follows (because of the computational difficulty posed by the MLP), we precomputed all triangles of constraints for each particular use case of the evaluation and reserved a special queue structure for storing them; that way, the underlying PWC algorithm will run as efficiently as possible.[26] Finally, only one of the CPU cores was used for the experiments.

Dataset and Measures In order to study the behavior of algorithm Minimize, we conducted experiments with QCNs of IA and RCC-8. These QCNs were generated from the model S [Nebel, 1996; Nebel, 1997]. This model can generate random satisfiable QCNs according to three parameters, n, d, and s, where n is the number of variables of the generated QCNs, d is the average number of variables connected with a non-trivial constraint (i.e., a constraint defined by a relation other than B), and s is the average number of base relations of a non-trivial constraint. For this model, the satisfiability of a generated QCN is guaranteed by augmenting it with a scenario. A set of QCNs generated according the model S using the parameters n, d and s will be denoted by $\mathsf{S}(n, d, s)$. We will present experiments with instances issued from the series $\mathsf{S}(n, d, |\mathsf{B}|/2)$,

[26] In relation to the implementation note, we remind the reader of our discussion in Section 5.2.1 where we explained the complexity of obtaining a triangle of constraints with respect to a graph when given a particular edge of that graph, and advocated the precomputation of all triangles of constraints with respect to that graph when the size of the QCN allows it.

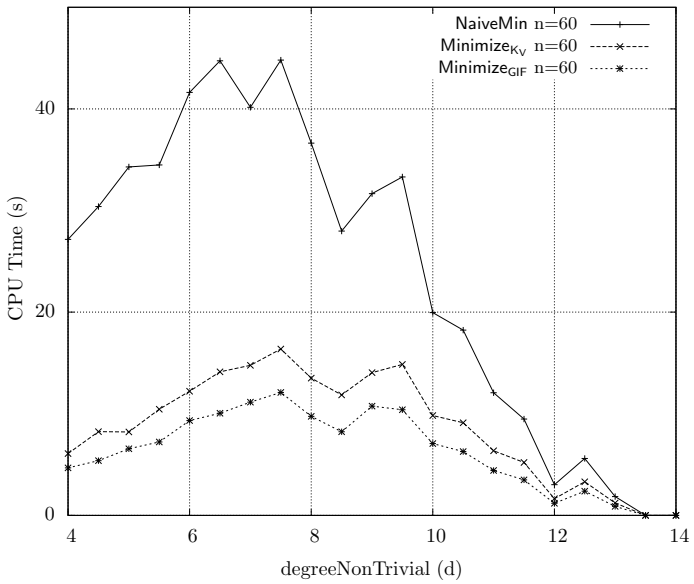


Figure 5.19: CPU time for instances of RCC-8 not forced to be consistent

with n varying from 30 to 60 with an incremental step of 10, and d varying from 4 to 22 with an incremental step of 0.5. For each series, we generated 50 QCNs. The subclasses of relations used as parameters for function Minimize are the two maximal tractable subclasses $\mathcal{H}_{\mathsf{IA}}$ for IA and $\hat{\mathcal{H}}_8$ for RCC-8. The CPU time per each series was considered as a comparison measure. Further, a timeout of 5 hours was given for each series.

As a side note, and as also discussed earlier in this chapter, a linear technique to triangulate a graph consists of adding extra edges produced by eliminating vertices one by one. Many heuristics have been proposed to order the vertices (cf. [Cano and Moral, 1994]), here we use the *GreedyFillIn* heuristic (GIF) [Bodlaender and Koster, 2010] to triangulate the constraint graph of a QCN (line 3 of the Minimize algorithm) and obtain a chordal graph.

Results In what follows, Minimize$_{\mathsf{GIF}}$ denotes the function Minimize with the use of GIF triangulation, whereas Minimize$_{\mathsf{K_V}}$ denotes the function Minimize where the chordal graph used is the complete graph K_V.

For every series, we note that the general approach used by Minimize greatly outperforms the naive approach followed by NaiveMin. As an example, consider Figure 5.18 which illustrates the CPU time required by the three functions to solve the series $\mathsf{S}(40, d, 6.5)$ and $\mathsf{S}(50, d, 6.5)$ of IA. Note that for $n > 50$, function NaiveMin cannot solve the series of IA in the 5 hours given as timeout. Further, by comparing the CPU time required by Minimize$_{\mathsf{K_V}}$ and Minimize$_{\mathsf{GIF}}$, the latter approach has significantly better performance. Using judicious triangulations as the one offered by the GIF heuristic increases the efficiency of the detection of the feasability or the unfeasibility of the base relations belonging to a constraint associated with an edge of the graph issued by the triangulation (in Step 2 of Minimize).

Similar results were obtained for QCNs of RCC-8, for instances not forced to be satisfiable (see Figure 5.19).

Algorithm 25: Delphys$\triangledown(\mathcal{N})$

in : An all-different QCN $\mathcal{N}$ defined over a distributive subclass of relations of a qualitative constraint language that is a relation algebra and for which every $\diamond$-consistent atomic QCN is satisfiable.

output : χ, the set of non-redundant relations in $\diamond(\mathcal{N})$.

```
1  begin
2      χ ← ∅;
3      G ← Triangulation(G(N));
4      N' ← PWC(N, G);
5      Q ← {(v_i, v_j) | {v_i, v_j} ∈ E(G(N)) with 0 ≤ i < j < |V|};
6      while Q ≠ ∅ do
7          (v_i, v_j) ← Q.pop();
8          τ ← ∅;
9          foreach v_k such that {v_i, v_k}, {v_k, v_j} ∈ E(G) do
10             t ← N'[v_i, v_k] ⋄ N'[v_k, v_j];
11             foreach b ∈ B do
12                 if b ∉ t then
13                     τ ← τ ∪ {b};
14         if τ ∪ N'[v_i, v_j] ≠ B then
15             χ ← χ ∪ {N'[v_i, v_j]};
16     return χ;
```

5.6 Efficient Algorithms for the Redundancy Problem of QCNs

In the case where a QCN $\mathcal{N}$ is defined over a distributive subclass of relations, and since algorithm PWC is able to enforce partial $\diamond$-consistency on a given QCN of a relation algebra due to Proposition 24 (at page 97), we can devise an algorithm based on PWC that will effiently extract the set of non-redundant constraints in the closure under weak composition of $\mathcal{N}$, viz., $\diamond(\mathcal{N})$, by implementing Proposition 23 (at page 95). That set of non-redundant relations will be essentially the same as the one provided by Lemma 2 (at page 36). Such an algorithm is presented in Algorithm 25, called Delphys$\triangledown$. We have the following result with respect to Delphys$\triangledown$:

Proposition 43 *Let $\mathcal{N} = (V, C)$ be an all-different* QCN *defined over a distributive subclass of relations of a qualitative constraint language that is a relation algebra and for which every $\diamond$-consistent atomic* QCN *is satisfiable. Then, algorithm* Delphys$\triangledown$ *terminates and returns the set of non-redundant relations in $\diamond(\mathcal{N})$.*

Then, due to Proposition 43, and Propositions 1 (at page 21) and 3 (at page 34), we have the following result:

Corollary 29 *Let $\mathcal{N} = (V, C)$ be a* QCN *defined over a distributive subclass of relations of Point Algebra, Cardinal Direction Calculus, Interval Algebra, Block Algebra, or* RCC-*8. We have that algorithm* Delphys$\triangledown$ *terminates and returns the set of non-redundant relations in $\diamond(\mathcal{N})$.*

Clearly, given a QCN $\mathcal{N} = (V, C)$, algorithm Delphys runs in $O(\delta|E||\mathsf{B}|)$ time, where δ denotes the maximum vertex degree of a triangulation G of $G(\mathcal{N})$, as the PWC algorithm call in line 4 dominates the overall execution time and the triangulation procedure in line 3 is linear in the size of G [Parter, 1961; Diestel, 2012].

Algorithm 26: extractPrimeQCN$_{\triangledown}(\mathcal{N})$

in : A QCN $\mathcal{N} = (V, C)$ defined over a subclass of relations of a qualitative constraint language that is a relation algebra and has patchwork for not trivially inconsistent and $\diamond$-consistent QCNs defined over that subclass of relations.
output : A prime QCN of $\mathcal{N}$.

```
1  begin
2      C' ← map({((v, v') : (B if v ≠ v' else {Id})) | v, v' ∈ V});
3      G ← Triangulation(G(N));
4      Q ← {(v_i, v_j) | {v_i, v_j} ∈ E(G(N)) with 0 ≤ i < j < |V|};
5      while Q ≠ ∅ do
6          (v_i, v_j) ← Q.pop();
7          flag ← True;
8          foreach b ∈ B \ C(v_i, v_j) do
9              if PWC(N_{[v_i,v_j]/{b}}, G) ≠ ⊥^V then
10                 C'(v_i, v_j) ← C(v_i, v_j); C'(v_j, v_i) ← (C(v_i, v_j))^{-1};
11                 flag ← False; break;
12         if flag = True then
13             C(v_i, v_j) = C(v_j, v_i) = B;
14     return (V, C');
```

Then, due to Propositions 43, 13 (at page 50), 1 (at page 21), and 3 (at page 34), and by taking into account the fact that given a QCN $\mathcal{N}$ we have that $\mathcal{N}_{\mathsf{min}}$ is unique, we can construct the prime QCN of a given QCN of Point Algebra, Cardinal Direction Calculus, or RCC-8 by exploiting Theorem 7 (at page 50), as it was presented in Section 3.5.4.

Theorem 7 (at page 50) allows us to construct a unique prime QCN out of a given QCN only for QCNs that satisfy certain conditions, i.e., QCNs defined over a distributive subclass of relations of Point Algebra, Cardinal Direction Calculus, or RCC-8. For QCNs of other qualitative constraint languages, or for QCNs defined over non-distributive subclasses of relations, we have to follow a different approach for extracting their (non-unique in the general case) prime networks. Let us consider the case where a QCN $\mathcal{N} = (V, C)$ is defined over a subclass of relations of a qualitative constraint language that has patchwork for not trivially inconsistent and $\diamond$-consistent QCNs defined over that subclass of relations. To test if a constraint $C(u, v)$ for some $\{u, v\} \in E(G(\mathcal{N}))$ is non-redundant in $\mathcal{N}$, we need to check if there exists a base relation $b \in \mathsf{B} \setminus C(u, v)$ such that the QCN $\mathcal{N}_{[u,v]/\{b\}}$ is satisfiable; this satisfiability check can be achieved through partial $\diamond$-consistency provided that a triangulation of the constraint graph of the given QCN is considered. (Note that due to Lemma 1 at page 31 we already know that given a QCN $\mathcal{N} = (V, C)$, a relation $\mathcal{N}[v, v']$ with $v, v' \in V$ is redundant if $\{v, v'\} \notin E(G(\mathcal{N}))$.) We can apply this procedure iteratively to construct a prime QCN out of a given QCN and, thus, devise algorithm extractPrimeQCN$_{\triangledown}$, presented in Algorithm 26. We have the following result:

Proposition 44 *Let $\mathcal{A} \in 2^{\mathsf{B}}$ be a subclass of relations of a qualitative constraint language that is a relation algebra and has patchwork for not trivially inconsistent and $\diamond$-consistent* QCN*s defined over that subclass of relations. Then, given a* QCN $\mathcal{N} = (V, C)$ *defined over $\mathcal{A}$, algorithm* extractPrimeQCN$_{\triangledown}$ *terminates and returns a prime* QCN *of $\mathcal{N}$.*

Then, due to Propositions 44, 1 (at page 21), and 16 (at page 54), we have the following result:

Algorithm 27: extractPrimeQCNSC$_{\bigtriangledown}(\mathcal{N},\mathcal{A})$

in : A QCN $\mathcal{N} = (V, C)$, and a subclass $\mathcal{A}$.
output : A QCN (V, C').

```
1  begin
2      C' ← map({((v, v') : (B if v ≠ v' else {Id})) | v, v' ∈ V});
3      G ← Triangulation(G(N));
4      Q ← {(v_i, v_j) | {v_i, v_j} ∈ E(G(N)) with 0 ≤ i < j < |V|};
5      while Q ≠ ∅ do
6          (v_i, v_j) ← Q.pop();
7          r ← B \ C(v_i, v_j);
8          if PartialConsistency(N_[v_i,v_j]/r, G, A) ≠ Null then
9              C'(v_i, v_j) ← C(v_i, v_j); C'(v_j, v_i) ← (C(v_i, v_j))^-1;
10         else
11             C(v_i, v_j) = C(v_j, v_i) = B;
12     return (V, C');
```

Corollary 30 *Let $\mathcal{N} = (V, C)$ be a* QCN *of Point Algebra, Cardinal Direction Calculus, Interval Algebra, Block Algebra, or* RCC-*8, defined over one of the classes $\mathcal{P}_{\mathsf{PA}}$, $\mathcal{P}_{\mathsf{CDC}}$, $\mathcal{H}_{\mathsf{IA}}$, $\mathcal{H}^n_{\mathsf{IA}}$, or $\hat{\mathcal{H}}_8$, $\mathcal{C}_8$, and $\mathcal{Q}_8$ respectively. We have that algorithm* extractPrimeQCN$_{\bigtriangledown}$ *terminates and returns a prime* QCN *of $\mathcal{N}$.*

Given a QCN $\mathcal{N} = (V, C)$, algorithm extractPrimeQCN runs in $O(\delta|E||E'||\mathsf{B}|^2)$ time, where E' denotes the set of edges of the constraint graph $G(\mathcal{N})$, E the set of edges of a triangulation G of that graph, and δ the maximum vertex degree of G. The extractPrimeQCN$_{\bigtriangledown}$ algorithm iterates a number of $O(|E'|)$ constraints and calls algorithm PWC for each one of the $O(|\mathsf{B}| - 1)$ base relations not belonging to a constraint; each such call requiring $O(\delta|E||\mathsf{B}|)$ time. It should be clear that the construction of a prime network depends on the order in which the constraints are processed, as we already noted that a prime network of a given QCN is in general not unique.

In the case of an arbitrary QCN, i.e., a QCN $\mathcal{N} = (V, C)$ that is defined over 2^{B}, we have to devise a different algorithm than the one presented earlier. Such an algorithm will build on the PartialConsistency algorithm presented in Algorithm 18 and will use a subclass of relations of a qualitative constraint language that has patchwork for not trivially inconsistent and $\diamond$-consistent QCNs defined over that subclass of relations. To test if a constraint $C(u, v)$ for some $\{u, v\} \in E(G(\mathcal{N}))$ is non-redundant in $\mathcal{N}$, we need to check if the QCN that results by replacing relation $C(u, v)$ with relation $r = \mathsf{B} \setminus C(u, v)$ in $\mathcal{N}$ is satisfiable; this satisfiability check can be achieved through the PartialConsistency algorithm along with a subclass of relations of a qualitative constraint language that has patchwork for not trivially inconsistent and $\diamond$-consistent QCNs defined over that subclass of relations. We can apply this procedure iteratively to construct a prime QCN out of a given QCN and, thus, devise algorithm extractPrimeQCNSC$_{\bigtriangledown}$, presented in Algorithm 27. We have the following result:

Proposition 45 *Let $\mathcal{A} \in 2^{\mathsf{B}}$ be a subclass of relations of a qualitative constraint language that is a relation algebra and has patchwork for not trivially inconsistent and $\diamond$-consistent* QCN*s defined over that subclass of relations. Then, given $\mathcal{A}$, and a* QCN *$\mathcal{N}$ defined over 2^{B}, we have that algorithm* extractPrimeQCNSC$_{\bigtriangledown}$ *terminates and returns a prime* QCN *of $\mathcal{N}$.*

Then, due to Propositions 45, 1 (at page 21), and 16 (at page 54), we have the following result:

Table 5.4: Performance comparison on CPU time

network	Delphys	Delphys$\bigtriangledown$	speedup (%)
nuts	0.26s	0.19s	26.9%
adm1	25 536.93s	222.33s	99.1%
gadm1	140 685.26s	329.43s	99.8%
gadm2	9.18s	2.34s	74.5%
adm2	∞	1 069.51s	$\sim$ 100%

Table 5.5: Effect on obtaining non-redundant relations

network	initial # of relations	non-redundant # of relations	decrease (%)
nuts	3 176	2 249	29.19%
adm1	44 832	44 601	0.52%
gadm1	159 600	158 440	0.73%
gadm2	589 573	292 331	50.42%
adm2	5 236 270	1 798 132	65.66%

Corollary 31 *Given one of the classes $\mathcal{P}_{\mathsf{CDC}}$, $\mathcal{H}_{\mathsf{IA}}$, $\mathcal{H}^n_{\mathsf{IA}}$, or $\hat{\mathcal{H}}_8$, $\mathcal{C}_8$, and $\mathcal{Q}_8$, and a* QCN $\mathcal{N}$ *of Cardinal Direction Calculus, Interval Algebra, Block Algebra, or* RCC-*8 respectively, we have that algorithm* extractPrimeQCNSC$\bigtriangledown$ *terminates and returns a prime* QCN *of $\mathcal{N}$.*

Given a QCN $\mathcal{N} = (V, C)$, and a subclass $\mathcal{A}$, the runtime of algorithm extractPrimeQCNSC$\bigtriangledown$ is $O(\alpha^{|E|}|E'|)$, where E' denotes the set of edges of the constraint graph $G(\mathcal{N})$ and E the set of edges of a triangulation of that graph, as it iterates a number of $O(|E'|)$ constraints and calls algorithm PartiaConsistency for each one of the constraints; each such call operating on a $O(\alpha^{|E|})$ search space. We remind the reader that α is the branching factor provided by subclass $\mathcal{A}$, e.g., $\alpha = 1.4375$ for subclass $\hat{\mathcal{H}}_8$ for RCC-8 [Renz and Nebel, 2001]).

5.6.1 Experimental Evaluation

In this section, we compare the performance of Delphys$\bigtriangledown$ with that of Delphys (presented in Section 3.5.4).

Technical Specifications The experiments were carried out on a computer with an Intel Core 2 Quad Q9400 processor with a CPU frequency of 2.66 GHz per core, 8 GB RAM, and the Precise Pangolin x86_64 OS. Both Delphys$\bigtriangledown$ and Delphys were written in pure Python and run with PyPy 2.4.0 (`http://pypy.org/`). Only one of the CPU cores was used.

Dataset and Measures We considered the real-world RCC-8 datasets that we used in Section 5.4.4, namely, `nuts`, `adm1`, `gadm1`, `gadm2`, and `adm2`. The aforementioned network instances are satisfiable. They comprise relations that are properly contained in any of the two maximal distributive subclasses $\mathcal{D}_8^{41}$ and $\mathcal{D}_8^{64}$ for RCC8. Also, some identical regions were properly amalgamated to satisfy the uniqueness property. We use the CPU time and the reduction in the number of constraints as our measures.

Results The results on the performance of Delphys$\bigtriangledown$ and Delphys are shown in Table 5.4. Note that symbol ∞ signifies that a reasoner hits the memory limit. The speedup for Delphys$\bigtriangledown$ reaches as high as nearly 100% for the cases where Delphys was actually able to reason with the

networks (e.g., `gadm1`). Table 5.5 shows the decrease that we can achieve with respect to the total number of non-redundant relations that we can obtain from an initial network, which allows one to construct sparse constraint graphs that boost various graph related tasks such as storing, querying, representing, and reasoning. Note that the constraint graphs of the initial networks are sparse, thus, a lot of redundancy is already avoided. Still, for the biggest network of the dataset, namely, `adm2`, the decrease is around 66%, which yields a number of non-redundant relations that is almost linear to the number of the vertices of that network, confirming a similar observation in [Li *et al.*, 2015a].[27]

5.7 Towards Efficient Utilization of Parallelism

In this section, we present a simple decomposition scheme that exploits the sparse and loosely connected structure of the constraint graphs of very large real-world QCNs, which have been of high interest in the recent literature [Nikolaou and Koubarakis, 2014; Sioutis and Condotta, 2014b; Sioutis, 2014], and paves the way for efficient utilization of parallelism. Our approach is based on extracting the smaller QCNs that correspond to the biconnected components of the constraint graph of a given large QCN and reasoning with these smaller biconnected QCNs completely separately, in a parallel or serial fashion, which, as our experimentation suggests, significantly decongests search when solving non-tractable QCNs.

However, before we proceed with the description of our method, we will first address certain issues with the decomposition-based approach presented in [Nikolaou and Koubarakis, 2014] that utilizes parallelism to check the satisfiability of QCNs of RCC-8. In particular, we will prove that the aforementioned approach lacks soundness, as the notion of a *partitioning graph* defined in that work is not coherent with the use of patchwork upon which it solely relies, in two ways, which we enumerate and analyse.

5.7.1 Partitioning Graphs and Non-Soundness

Let $G = (V, E)$ be a graph and k a positive integer. If $U \subseteq V$, then $G(U)$ will denote the subgraph of G that is induced by the set of vertices U. A set $\{V_i \subseteq V \mid 1 \leq i \leq k\}$ with k pairwise-disjoint elements such that $\bigcup_{i=1}^{k} V_i = V$, is called a k-way partitioning of G [Bichot and Siarry, 2011]. Finally, let $\emptyset$ denote the empty, edgeless, graph. We recall the following definition of a partitioning graph from [Nikolaou and Koubarakis, 2014]:

Definition 41 *Let $G = (V, E)$ be a graph and $\{V_1, \ldots, V_k\}$ a k-way partitioning of G for some positive integer k. A* partitioning graph *P of G is a tuple $(V_P, E_P, \lambda_P, G_P)$, where $V_P = \{v_1, \ldots, v_k\}$ is the set of its nodes, E_p the set of its edges, $\lambda_P : V_P \rightarrow 2^V$ a function that maps each node of P to a partition (subset of V) of G, and G_P a set of k subgraphs (parts) of G. The following conditions must be satisfied:*

- *If $G_i \in G_P$ then the set of vertices of G_i is a superset U of $\lambda_P(v_i)$ and the set of its edges is $E(G(U))$.*
- *Any edge in G should be present in at least one subgraph $G_i \in G_P$.*

[27] Of course, Delphys would have achieved the same decrease, had it been able to reason with such large networks.

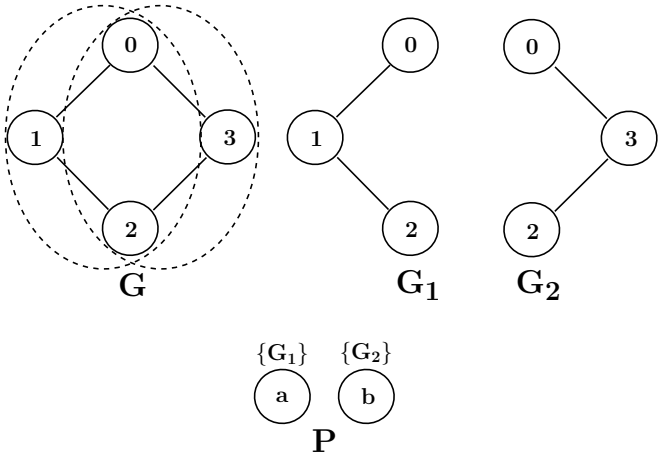


Figure 5.20: A graph and its partitioning graph with the parts comprising it (also contained in dashed circles in the initial graph)

- *An edge $\{v_i, v_j\}$ belongs to E_P if and only if $G_i \cap G_j \neq \emptyset$ (i.e., if and only if the subgraphs G_i and G_j corresponding to nodes v_i and v_j respectively share a common edge).*

Let G be a graph and $P = (V_P, E_P, \lambda_P, G_P)$ its partitioning graph. Then, an edge e of G present in more than one subgraph $G_i \in G_P$ is called a *global edge*. An edge e of G present in exactly one subgraph $G_i \in G_P$ is called a *local edge*.

We will now enumerate the issues that lead to non-soundness and provide counter-examples for each case. The reader is kindly asked to refer to [Nikolaou and Koubarakis, 2014] and check that the flaws pointed out here are actually present in [Nikolaou and Koubarakis, 2014].

The issues that we will enumerate will allow us to infer the following fact:

Proposition 46 *The main result of [Nikolaou and Koubarakis, 2014] with respect to checking the satisfiability of a* QCN *of* RCC-*8 does not hold. In particular, Propositions* 2 *and* 3 *in [Nikolaou and Koubarakis, 2014] do not hold.*

We begin with the first issue.

Issue 1 The first issue has to do with the fact that a complete agreement on the constraints between the common variables of two networks is not achieved in order to allow the applicability of patchwork. Let us consider the example of Figure 5.20. Graph G is partitioned into two parts, namely, G_1 and G_2. The partitioning graph is shown in the lower part of the figure, and it comprises the set of nodes $\{a, b\}$ and an empty set of edges. Node a corresponds to subgraph G_1 and node b to subgraph G_2. Its set of edges E_P is empty as subgraphs G_1 and G_2 do not share a common edge (that would otherwise be the global edge $(0, 2)$), thus, the only possible edge (a, b) does not exist. In [Nikolaou and Koubarakis, 2014] the authors perform $\diamond$-consistency on the subgraphs of a graph separately, in a parallel fashion, and then rely on the set of edges E_P to identify the subgraphs among which a complete agreement has to be ensured (the reader is kindly asked to refer to line 7 in the function of Algorithm 2 in [Nikolaou and Koubarakis, 2014]). If, as in this example, such an edge does not exist, a complete agreement is never achieved. This can be the cause of failing to identify inconsistencies. Let us assume that graph G, as depicted in Figure 5.20, is the constraint graph of a given QCN comprising constraints $C_{01} = C_{12} = C_{23} = C_{30} = \{TPP\}$.

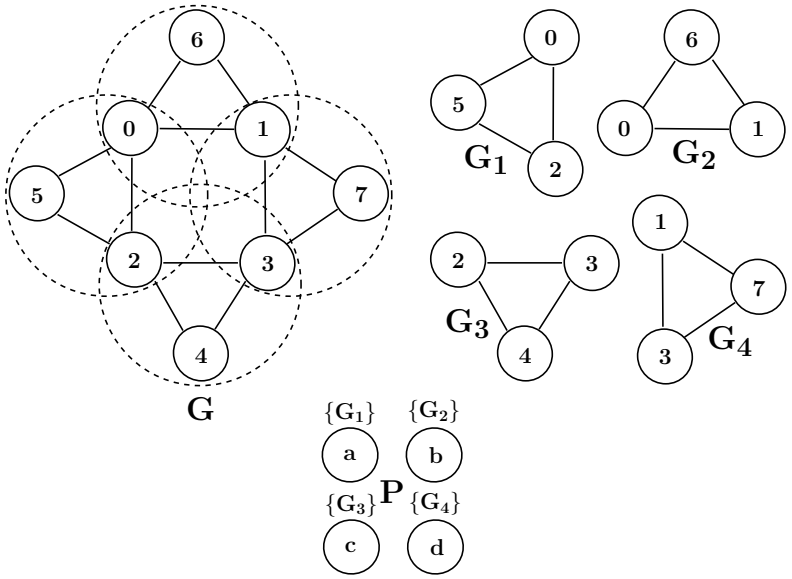


Figure 5.21: A graph and its partitioning graph with the parts comprising it (also contained in dashed circles in the initial graph)

This yields an unsatisfiable network, as it basically infers that region 0 is properly contained in region 2, and vice versa. Applying $\diamond$-consistency on that network would result in the empty relation assignment for constraint C_{02} (inconsistency). However, that constraint is never checked in our example. Although the authors implicitly complete subgraphs G_1 and G_2 in order to apply $\diamond$-consistency, they do not complete these subgraphs when computing their intersection, as clearly specified in the last bullet of Definition 41. Even if they did implicitly consider complete subgraphs for that part of the definition, and the edge (a, b) indeed existed, line 7 in the function of Algorithm 2 in [Nikolaou and Koubarakis, 2014] still requires that an agreement should only be achieved for every common edge of G_1 and G_2 (the initial non-complete subgraphs), which is none. If they implicitly considered complete subgraphs for that part of the algorithm too, then this particular issue for a 2-way partitioning would be resolved. We have also verified this issue experimentally with the implementation used in [Nikolaou and Koubarakis, 2014].

Before proceeding to the next issue, let us assume that the first issue is fixed with everything that we propose, and a 2-way partitioning is actually valid for applying patchwork. We mean to show, that the concept of a partitioning graph is beyond repair, unless it is structured in a way that it defines a tree decomposition, which defeats the purpose of having to define a partitioning graph in the first place.

The second issue follows.

Issue 2 This issue has to do with the fact that even if the first issue is resolved, the partitioning graph can suffer from the existence of cycles that are created by subgraphs of a given graph. Let us consider the example of Figure 5.21. Graph G is partitioned into four parts, namely, G_1, G_2, G_3, and G_4. The partitioning graph is shown in the lower part of the figure, and the correspondence between its sets of nodes and edges with the different subgraphs should be clear up to this point. Note that all subgraphs are complete, thus, they completely overlap with each other on the common vertices. For example, graph G_1 completely overlaps with graph G_2 on the edges of the graph defined on the single common vertex 0, as their intersection yields the complete

graph on singe vertex 0. Although such an overlap is trivial, as a complete graph on a single vertex (singleton graph) does not have any edges, it is sufficient to ensure the applicability of patchwork for the corresponding constraint networks. (Our example can be easily extended to non-trivial overlaps.) However, due to the last bullet of Definition 41, the partitioning graph is unable to obtain any edges, as there can exist no global edges. In fact, even if some edges existed in E_P, in any possible combination and amount, the partitioning graph would still fail to detect the cycle that is constructed by the complete subgraphs G_1, G_2, G_3, and G_4, namely, the cycle defined by vertices 0, 1, 2, and 3. This cycle, as shown in the example of Figure 5.20, can harbor an inconsistency. Such a cycle exists also in [Nikolaou and Koubarakis, 2014, Fig. 1] between vertices 3, 4, 5, and 7 there. Patchwork alone is only valid for tree decompositions, as tree decompositions guarantee acyclicity of cliques and, thus, do not harbor cycles with potential inconsistencies that cannot be detected by the application of $\diamond$-consistency on the different cliques. This issue was again verified experimentally.

Essentially, the approach defines a partial algorithm; a given satisfiable QCN will be shown to be satisfiable, as the approach in [Nikolaou and Koubarakis, 2014] due to disregarding constraints operates on a less restrictive constraint graph of the input network where constraint propagation and consistency checks are limited, whilst an unsatisfiable QCN may be shown to be satisfiable.

Impact on Performance The main contribution of [Nikolaou and Koubarakis, 2014] lies in the performance of its offered implementation, as it promises efficiency that goes well beyond the state of the art. Computing a good k-way partitioning[28] alone is among the graph partitioning problems that fall under the category of NP-hard problems [Garey *et al.*, 1976], and solutions to these problems are generally derived using heuristics and approximation algorithms, such as the ones offered by the METIS[29] software employed in [Nikolaou and Koubarakis, 2014]. We leave aside any extra computational complexity that would result from needing to restrict a partitioning graph to being a tree decomposition (e.g., by identifying cycles or using some recursion as in [Huang *et al.*, 2013]), and focus on native search. As explained in Section 3.5.2, native search in qualitative spatial and temporal reasoning is bound to the number of constraints of a given QCN, and not to its number of variables as in "traditional" constraint programming. This is because, in a sense, the constraints of a given QCN are the *true* variables for which we have to assign some relation. Indeed, the search space defined in [Nikolaou and Koubarakis, 2014] relies mainly on the number of constraints of a given QCN. In particular, we can recall the following proposition from [Nikolaou and Koubarakis, 2014]:

Proposition 47 ([Nikolaou and Koubarakis, 2014]) *Let $G = (V, E)$ be the constraint graph of a* QCN *of* RCC-*8 and P a partitioning graph of G with k parts. The search space of algorithm* DConsistency *[Nikolaou and Koubarakis, 2014] is $O(|\mathsf{B}|^g(|\mathsf{B}|gkm^3 + k\alpha^l m^3))$, where g is the number of global edges, l and m the maximum number of local edges and vertices respectively among all parts of P, and α the branching factor of the subclass of relations employed. If Π denotes the aforementioned search space, then given p processing units and assuming a balanced partitioning among the k parts (i.e., $m = |V|/k$), the elapsed running time of algorithm* DConsistency *is $O(\frac{\Pi}{p})$.*

We showed earlier that some global edges can be disregarded, thus, parameter g as defined in Proposition 47 leads to a significantly reduced search space for the implementation considered

[28] *Good* in terms of obtaining smaller components that meet specific properties, for example, a good partitioning can be defined as one in which the number of edges running between separated components is small.

[29] `http://glaros.dtc.umn.edu/gkhome/views/metis`

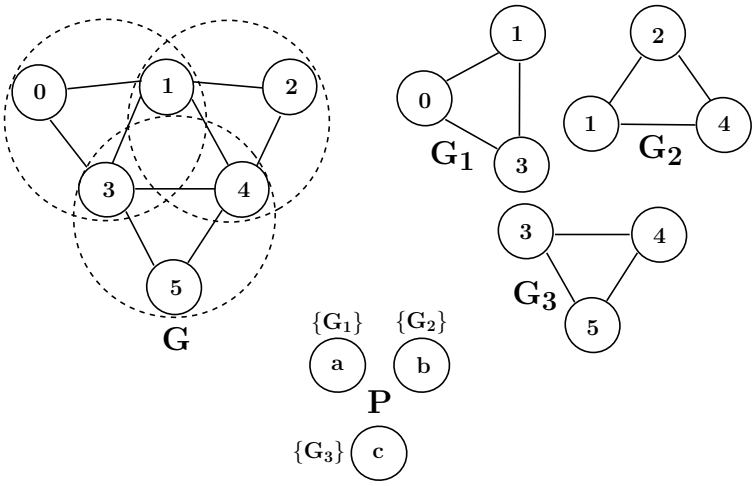


Figure 5.22: A chordal graph and its partitioning graph with the parts comprising it (also contained in dashed circles in the initial graph)

in [Nikolaou and Koubarakis, 2014] with respect to the one that should normally be considered, as g has an exponential contribution. However, even in that case, a re-evaluation of the implementation used in [Nikolaou and Koubarakis, 2014] against state of the art solvers, showed that it performs very poorly with respect to the state of the art [Sioutis, 2014]. The work in [Sioutis, 2014] does not deal with any of the issues that we dealt with here as it assumes a partitioning graph to implicitly define a tree decomposition, thus, [Sioutis, 2014] presents mostly lower bounds on the performance of the implementation used in [Nikolaou and Koubarakis, 2014].

Fixing the Issues We noted earlier that the concept of a partitioning graph is beyond repair, unless it is structured in a way that it defines a tree decomposition. It may seem tempting as a quick hack to triangulate the constraint graph of a given QCN of RCC-8 and, thus, obtain a chordal constraint graph of that QCN, and feed it directly to the partitioning algorithm described in [Nikolaou and Koubarakis, 2014]. This may still yield non-soundness; we explain as follows. Consider the example shown in Figure 5.22 where the chordal graph G is partitioned into 3 subgraphs. The partitioning graph P is such that it is unable to capture/break the cycle defined by vertices 1, 3, and 4. This cycle may harbor an inconsistency, which will not be detected by the application of $\diamond$-consistency on the different parts of the partitioning graph.

One way to force a partitioning graph into defining a tree decomposition is using METIS in a recursive manner, as it is done in [Huang *et al.*, 2013]. In particular, one has to initially partition a given graph G into two parts, and then recursively apply the same procedure on the obtained parts, until no further partitioning can occur. However, this can be a costly operation. A faster way is to rely on chordal graphs (tree decompositions into cliques), which can both be constructed and also yield a natural tree decomposition of their cliques in linear time [Diestel, 2012]. The graphs induced by the cliques can then be collected at no extra cost and serve as the parts of the partitioning graph; consequently, the approach described in [Nikolaou and Koubarakis, 2014] can then be carried out with soundness and completeness.

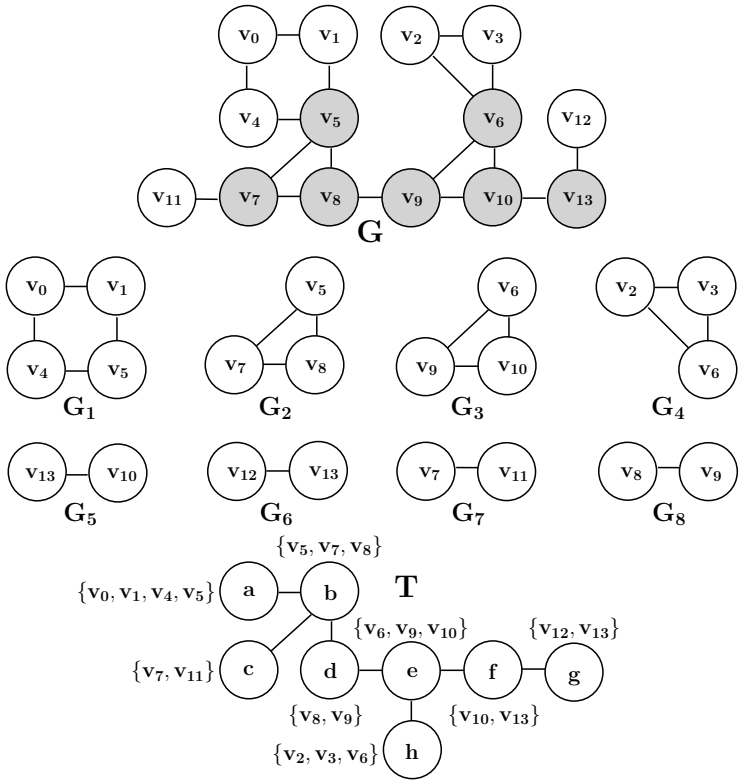


Figure 5.23: A graph G (top) with its biconnected components (middle) and its tree decomposition (bottom)

5.7.2 A Simple Decomposition Scheme for Sound and Efficient Use of Parallelism

As noted earlier, in [Nikolaou and Koubarakis, 2014] the authors provided a parallel implementation for checking the satisfiability of a QCN of RCC-8, which however lacks soundness (Proposition 46 at page 148). Now, we present a simple decomposition scheme that exploits the sparse and loosely connected structure of the constraint graphs of very large real-world QCNs and allows for sound and efficient utilization of parallelism. Our approach is based on extracting the smaller QCNs that correspond to the biconnected components of the constraint graph of a given large QCN and reasoning with these smaller biconnected QCNs completely separately, in a parallel or serial fashion, which, as our experimentation suggests, significantly decongests search when solving non-tractable QCNs.

First, we recall a definition from [Dechter, 2003] regarding biconnected graphs and components.

Definition 42 *A connected graph G is said to have an* articulation vertex *u if there exist vertices v and v' such that all paths connecting v and v' pass through u. A graph that has an articulation vertex is called* separable*, and one that has none is called* biconnected*. A maximal subgraph with no articulation vertices is called a* biconnected component*.*

Intuitively, an articulation vertex is any vertex whose removal increases the number of connected components in a given graph. From [Dechter, 2003] we also have the following property:

Property 1 ([Dechter, 2003]) *Let G be a graph and $\{G_1, \ldots, G_n\}$ its biconnected components. Then, there exists a* tree decomposition *$(T, \{X_1, \ldots, X_n\})$ of G, where cluster $X_i \subseteq V(G)$ induces the biconnected component G_i of G, for every $i \in \{1, \ldots, n\}$.*

Let us now view the discussed notions in an example. Figure 5.23 depicts a graph G, along with its biconnected components, and its tree decomposition. Vertices in grey color are the articulation vertices of G. The tree decomposition comprises a tree $T = (I, F)$ and a cluster X_i for every node $i \in I$ of that tree, e.g., $X_a = \{v_0, v_1, v_4, v_5\}$. We can obtain the following proposition:

Proposition 48 *Let $\mathcal{N}$ be a* QCN *defined on a language that has patchwork for satisfiable atomic* QCN*s, and let $\{G_1, \ldots, G_k\}$ be the biconnected components of its constraint graph $G(\mathcal{N})$. Then, $\mathcal{N}$ is satisfiable iff $\mathcal{N}_i$ is satisfiable for every $i \in \{1, \ldots, k\}$, where $\mathcal{N}_i$ is $\mathcal{N}{\downarrow_{V(G_i)}}$.*

Proof. By Property 1, the constraint graph $G(\mathcal{N})$ has a tree decomposition $(T, \{X_1, \ldots, X_k\})$, where cluster X_i induces G_i, for every $i \in \{1, \ldots, k\}$. We can also infer by Definition 42, that $\forall i, j \in \{1, \ldots, k\}$ with $i \neq j$, $V(G_i) \cap V(G_j)$ contains at most one vertex u. If $\mathcal{N}_i$ is satisfiable for every $i \in \{1, \ldots, k\}$, we can obtain a satisfiable atomic sub-QCN of $\mathcal{N}_i$, i.e., a scenario S_i of $\mathcal{N}_i$, for every $i \in \{1, \ldots, k\}$. For any possible scenarios and any $i, j \in \{1, \ldots, k\}$ with $i \neq j$, we will have that $\mathcal{S}_i = (V(G_i), C_i)$ and $\mathcal{S}_j = (V(G_j), C_j)$ will always agree on the single unary constraint that is defined by a single vertex $u \in V(G_i) \cap V(G_j)$ whenever we have that $V(G_i) \cap V(G_j) \neq \emptyset$, i.e., $C_i(u, u) = C_j(u, u) = \{\mathsf{Id}\}$, as by Definition 4 (at page 26) we have that for any QCN $\mathcal{M} = (V, C)$, $C(v, v) = \{\mathsf{Id}\}$ $\forall v \in V$. Similarly to the proof of Proposition 19 (at page 90), we can apply patchwork to patch together all the satisfiable atomic QCNs $\mathcal{S}_i$ with $i \in \{1, \ldots, k\}$ in a sequential manner and, thus, derive the satisfiability of $\mathcal{N}$. If $\mathcal{N}$ is satisfiable, then, clearly, $\mathcal{N}_i$ will be satisfiable for every $i \in \{1, \ldots, k\}$. ⊣

Consequently, by Propositions 48, 3 (at page 34), and 16 (at page 54), we have the following result:

Corollary 32 *Let $\mathcal{N}$ be a* QCN *of Point Algebra, Cardinal Direction Calculus, Interval Algebra, Block Algebra, or* RCC-8*, and let $\{G_1, \ldots, G_k\}$ be the biconnected components of its constraint graph $G(\mathcal{N})$. Then, $\mathcal{N}$ is satisfiable iff $\mathcal{N}_i$ is satisfiable for every $i \in \{1, \ldots, k\}$, where $\mathcal{N}_i$ is $\mathcal{N}{\downarrow_{V(G_i)}}$.*

It is important to note that the proof of Proposition 48 is based on tree decompositions whose nodes correspond to clusters where any two clusters share at most one vertex with each other. In case two clusters share more than one vertex with each other, the involved QCNs should be, for instance (and among other conditions), not trivially inconsistent, $\diamond$-consistent, and defined over a subclass of relations of a qualitative constraint language that has patchwork for not trivially inconsistent and $\diamond$-consistent QCNs defined over that subclass of relations, as it is specified in Proposition 19 (at page 90) and considered in [Sioutis and Koubarakis, 2012] for RCC-8 and in [Chmeiss and Condotta, 2011] for IA respectively. A simple algorithm for obtaining a collection of QCNs that correspond to the biconnected components of the constraint graph of a given QCN is presented in Algorithm 28. Note that in lines 2–3 we immediately return the input QCN if it is trivially inconsistent, as it would not make any sense to continue with the decomposition procedure. Function BCSubgraphs(G) in line 4 returns the biconnected components of a graph $G = (V, E)$ and has a runtime of $O(|E|)$ [Dechter, 2003]. Note that in line 4 we keep only the components of order greater than 2, as any not trivially inconsistent qualitative constraint

Algorithm 28: Decomposer($\mathcal{N}$)

```
in      : A QCN N = (V, C).
output  : A collection of QCNs.
1  begin
2      if ∃{v, v'} ∈ E(G(N)) with C(v, v') = ∅ then
3          return {(V, C)};
4      S ← {g | g ∈ BCSubgraphs(G(N)); and |V(g)| > 2};
5      χ ← ∅;
6      while S ≠ ∅ do
7          g ← S.pop();
8          V_g ← V(g); E_g ← E(g);
9          C_g ← map({((v, v') : (B if v ≠ v' else {Id})) | v, v' ∈ V_g});
10         foreach {v, v'} ∈ E_g do
11             C_g(v, v') ← C(v, v'); C_g(v', v) ← C(v', v);
12         χ ← χ ∪ {(V_g, C_g)};
13     return χ;
```

Algorithm 29: Solver+($\mathcal{N}$)

```
in      : A QCN N = (V, C).
output  : True, or False.
1  begin
2      foreach n ∈ Decomposer(N) do
3          if not ‖ Solver(n) ‖ then
4              return False;
5      return True;
```

network of less than 3 variables is trivially satisfiable (by definition of a base relation). In what follows, we always consider components of order greater than 2. Based on algorithm Decomposer, we can obtain an algorithm to increase the performance of any given state of the art solver that is sound and complete for checking the satisfiability of a given QCN $\mathcal{N}$ defined on a language that has patchwork for satisfiable atomic QCNs; that algorithm is presented in Algorithm 29. Let us denote any such given state of the art solver by Solver. Then, Algorithm 29 will use Solver to decide the satisfiability of the QCNs that correspond to the biconnected components of the constraint graph of $\mathcal{N}$. The enclosure with symbol ‖ for Solver denotes the fact that Solver can be used in a parallel or serial fashion.

Regarding the MLP, we can have the following result:

Proposition 49 *Let $\mathcal{N} = (V, C)$ be a satisfiable* QCN *defined on a language that has patchwork for satisfiable atomic* QCN*s, and let $\{G_1, \ldots, G_k\}$ be the biconnected components of its constraint graph $G(\mathcal{N})$. Then, a base relation $b \in C(u, v)$, with $u, v \in V(G_i)$, is feasible (resp. unfeasible) iff there exists (resp. there does not exist) a scenario $\mathcal{S}_i = (V_i, C_i')$ of $\mathcal{N}_i = (V_i, C_i)$ such that $C_i'(u, v) = \{b\}$, where $\mathcal{N}_i$ is $\mathcal{N}{\downarrow_{V(G_i)}}$, for some $i \in \{1, \ldots, k\}$.*

Proof. Let $\mathcal{N}' = (V, C')$ be the QCN defined by $C'(u, v) = \{b\}$, $C'(v, u) = \{b\}^{-1}$, and $C'(y, w) = C(y, w)\ \forall (y, w) \in (V \times V) \setminus \{(u, v), (v, u)\}$. Further, let $\mathcal{N}_i' = (V_i, C_i')$ be the restriction of $\mathcal{N}'$ to $V(G_i)$, viz., $\mathcal{N}'{\downarrow_{V(G_i)}}$. Then, by Proposition 48 and as G_i is a biconnected component of $G(\mathcal{N})$,

we know that $\mathcal{N}'$ is satisfiable iff $\mathcal{N}_i'$ is satisfiable; in addition, any scenario $S_i = (V_i, C_i'')$ of $\mathcal{N}_i'$ is the restriction of some scenario $S = (V, C'')$ of $\mathcal{N}'$ to V_i, and any scenario $S = (V, C'')$ of $\mathcal{N}'$ is the extension of some scenario $S_i = (V_i, C_i'')$ of $\mathcal{N}_i'$ to V. As such, the feasibility of b can be characterized by considering $\mathcal{N}_i$ instead of $\mathcal{N}$. ⊣

Given a satisfiable QCN $\mathcal{N} = (V, C)$, Proposition 49 allows one to quickly characterize the feasibility of a base relation $b \in C(u, v)$, with $u, v \in V(G')$, where G' is a biconnected component of the constraint graph $G(\mathcal{N})$. If $u, v \notin V(G')$ for any biconnected component G' of $G(\mathcal{N})$, then b belongs to a constraint that is labeled with the universal relation B and its feasibility can still be efficiently characterized under certain conditions by a function similar to extractFeasible as described in [Amaneddine *et al.*, 2013].

Consequently, by Propositions 49, 3 (at page 34), and 16 (at page 54), we have the following result:

Corollary 33 *Let $\mathcal{N}$ be a satisfiable* QCN *of Point Algebra, Cardinal Direction Calculus, Interval Algebra, Block Algebra, or* RCC-*8, and let $\{G_1, \ldots, G_k\}$ be the biconnected components of its constraint graph $G(\mathcal{N})$. Then, a base relation $b \in C(u, v)$, with $u, v \in V(G_i)$, is feasible (resp. unfeasible) iff there exists (resp. there does not exist) a scenario $\mathcal{S}_i = (V_i, C_i')$ of $\mathcal{N}_i = (V_i, C_i)$ such that $C_i'(u, v) = \{b\}$, where $\mathcal{N}_i$ is $\mathcal{N}{\downarrow_{V(G_i)}}$, for some $i \in \{1, \ldots, k\}$.*

Regarding the redundancy problem, we can have the following result:

Proposition 50 *Let $\mathcal{N} = (V, C)$ be a satisfiable* QCN *defined on a language that has patchwork for satisfiable atomic* QCN*s, and let $\{G_1, \ldots, G_k\}$ be the biconnected components of its constraint graph $G(\mathcal{N})$. Then, a relation $C(u, v)$, with $u, v \in V$, is non-redundant in $\mathcal{N}$ iff $\{u, v\} \in E(G_i)$ and $C(u, v)$ is non-redundant in $\mathcal{N}_i = (V_i, C_i)$, where $\mathcal{N}_i$ is $\mathcal{N}{\downarrow_{V(G_i)}}$, for some $i \in \{1, \ldots, k\}$.*

Proof. Clearly, a relation $C(u, v)$ is redundant in $\mathcal{N}$ if $\{v, v'\} \notin E(G_i)$ for any $i \in \{1, \ldots, k\}$, as it will correspond to the universal relation B. Let us consider a relation $C(u, v)$ where $\{u, v\} \in E(G_i)$ for some $i \in \{1, \ldots, k\}$. Let $\mathcal{N}' = (V, C')$ be the QCN defined by $C'(u, v) = \mathsf{B} \setminus C(u, v)$, $C'(v, u) = \mathsf{B} \setminus (C(u, v))^{-1}$, and $C'(y, w) = C(y, w)\ \forall (y, w) \in (V \times V) \setminus \{(u, v), (v, u)\}$. Further, let $\mathcal{N}_i' = (V_i, C_i')$ be the restriction of $\mathcal{N}'$ to $V(G_i)$, viz., $\mathcal{N}'{\downarrow_{V(G_i)}}$. Then, by Proposition 48 and as G_i is a biconnected component of $G(\mathcal{N})$, we know that $\mathcal{N}'$ is satisfiable iff $\mathcal{N}_i'$ is satisfiable; in addition, any scenario $S_i = (V_i, C_i'')$ of $\mathcal{N}_i'$ is the restriction of some scenario $S = (V, C'')$ of $\mathcal{N}'$ to V_i, and any scenario $S = (V, C'')$ of $\mathcal{N}'$ is the extension of some scenario $S_i = (V_i, C_i'')$ of $\mathcal{N}_i'$ to V. Finally, since for any scenario there exists a solution that satisfies all of its base relations, the redundancy of $C(u, v)$ can be characterized by considering $\mathcal{N}_i$ instead of $\mathcal{N}$. ⊣

Consequently, by Propositions 50, 3 (at page 34), and 16 (at page 54), we have the following result:

Corollary 34 *Let $\mathcal{N}$ be a satisfiable* QCN *of Point Algebra, Cardinal Direction Calculus, Interval Algebra, Block Algebra, or* RCC-*8, and let $\{G_1, \ldots, G_k\}$ be the biconnected components of its constraint graph $G(\mathcal{N})$. Then, a relation $C(u, v)$, with $u, v \in V$, is non-redundant in $\mathcal{N}$ iff $\{u, v\} \in E(G_i)$ and $C(u, v)$ is non-redundant in $\mathcal{N}_i = (V_i, C_i)$, where $\mathcal{N}_i$ is $\mathcal{N}{\downarrow_{V(G_i)}}$, for some $i \in \{1, \ldots, k\}$.*

Reviewing a Real-World Dataset

We review the real-world RCC-8 datasets that we used in Section 5.4.4, and which were also used in [Nikolaou and Koubarakis, 2014], namely, `nuts`, `adm1`, `gadm1`, `gadm2`, and `adm2`. The

Table 5.6: Characteristics of real RCC-8 networks

network	# of nodes	# of edges	avg. degree
nuts	2 235	3 176	2.84
adm1	11 762	44 832	7.62
gadm1	42 749	159 600	7.47
gadm2	276 729	589 573	4.26
adm2	1 732 999	5 236 270	6.04

Table 5.7: Biconnected components of real RCC-8 networks

network	# of components	max order	median order	min order
nuts	64	52	8	3
adm1	5	11 666	30	3
gadm1	166	19 864	6	3
gadm2	2 285	2 371	18	3
adm2	2 889	22 808	579	4

aforementioned network instances are tractable and contain at most two base RCC-8 relations per edge. The characteristics of the constraint graphs of these networks are presented in Table 5.6.

As it can be seen, the constraint graphs of the networks vary in order, but they are all relatively sparse. This comes as no surprise, as real-world graphs often present a scale-free structure [Barabasi and Bonabeau, 2003], which results in them being sparse [Del Genio *et al.*, 2011]. This was also discussed again in detail in Section 5.2.2 Thus, we expect these constraint graphs to be loosely connected and yield a high number of biconnected components. We can view information regarding the biconnected components of the constraint graphs of our networks in Table 5.7 (where by *max order*, *median order*, and *min order* we refer to the maximum, median, and minimum number of vertices, respectively, met among the biconnected components).

The findings are quite impressive, in the sense that the maximum order among the biconnected components of a constraint graph is significantly smaller than the order of that graph. For example, the constraint graph of the biggest real RCC-8 network, namely, adm2, has an order of value 1 732 999, but the maximum order among its biconnected components is only of value 22 808. Note also that, as the median metric suggests, most of the biconnected components of a graph have an order much closer to the minimum order than the maximum order among the biconnected components of that graph.

Instances for evaluating the satisfiability checking performance of the reasoners for non-tractable QCNs, which are of our interest here, were constructed in [Nikolaou and Koubarakis, 2014] with the introduction of $\mathcal{NP}_8$ relations [Renz and Nebel, 2001] in the networks' edges. These instances will be denoted by hard-nuts, hard-adm1, and hard-gadm1 in the evaluation that follows, and are structurally identical to networks nuts, adm1, and gadm1 respectively, i.e., their constraint graphs have the same characteristics as those presented in Tables 5.6 and 5.7.

As [Nikolaou and Koubarakis, 2014] suggests, some state of the art reasoners, such as GQR [Gantner *et al.*, 2008], use a matrix to represent a QCN $\mathcal{N} = (V, C)$, which has a $O(|V|^2)$ memory requirement. It would be impossible to store a graph of the order of adm2 in a matrix as we would need $\sim 3TB$ of memory. Even if memory was not the issue, the time complexity alone of a $\diamond$-consistency algorithm would explode, while the backtracking algorithm that is typically used for tackling non-tractable QCNs and makes use of $\diamond$-consistency as a forward checking step, would suffer from an increased search space. Heuristics for the backtracking algorithm would also have a hard time distinguishing between biconnected components. Consider for example a

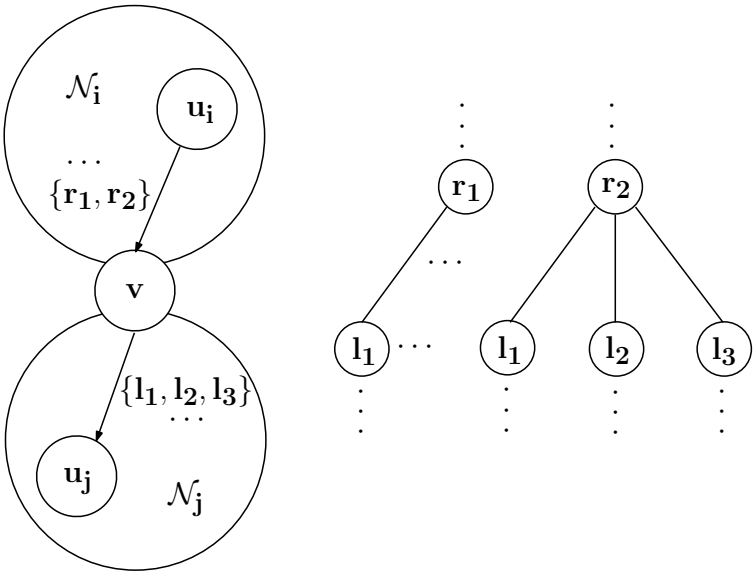


Figure 5.24: A separable constraint graph with an articulation vertex v

situation where the backtracking algorithm backtracks from an instatiation of a constraint in a biconnected component to an instantiation of a constraint in a different biconnected component. Since the constraints belong to different biconnected components, we have already shown that they are completely unrelated to each other (i.e., satisfying one constraint does not affect the other in any way); nevertheless, they might still define a huge branch in the search-tree that is spawned by the backtracking algorithm. Such a situation is depicted in Figure 5.24, which presents two QCNs $\mathcal{N}_i = (V_i, C_i)$ and $\mathcal{N}_j = (V_j, C_j)$ such that $V_i \cap V_j = \{v\}$. Let us assume that their constraint graphs are biconnected. Then, the constraint graph of $\mathcal{N}_i \cup \mathcal{N}_j$ has $G(\mathcal{N}_i)$ and $G(\mathcal{N}_j)$ as its biconnected components. It is clear that the valuation of constraint $C_i(u_i, v)$ with any of the values r_1 or r_2 does not affect the satisifiability or unsatisfiability of the valuation of constraint $C_j(v, u_j)$ with any of the values l_1, l_2, or l_3, and vice versa. However, if we choose not to treat the biconnected components separately, a huge branch might be defined, as viewed in Figure 5.24, that could otherwise be entirely avoided. Proposition 48 (at page 153) allows us to treat the QCNs that correspond to biconnected components completely separately, in a parallel or serial fashion, and avoid the aforementioned bothersome issues.

5.7.3 Experimental Evaluation

In this section, we evaluate our decomposition scheme with a variety of different solvers in the literature. If Solver is the name of a reasoner, Solver+ denotes the use of Algorithm 29 with that reasoner.

Technical Specifications The experiments were carried out on a computer with an Intel Core 2 Quad Q9400 processor with a CPU frequency of 2.66 GHz per core, 8 GB RAM, and the Precise Pangolin x86_64 OS. GQR (under version 1500) was compiled with gcc/g++ 4.6.3 and Sarissa, Phalanx, and Phalanx$_\bigtriangledown$ [Sioutis and Condotta, 2014b] (all under version 0.2) were run with PyPy 2.4.0[30], which fully implements Python 2.7.8. For all reasoners, the best performing

[30] `http://pypy.org`

Table 5.8: Performance comparison based on elapsed time

solver	GQR	GQR+	Pha.	Pha.+	Sar.	Sar.+	Pha.$\triangledown$	Pha.$\triangledown$+
hard-nuts	2.0s	0.1s	4.0s	0.6s	0.8s	0.6s	0.9s	0.6s
hard-adm1	4.7E3s	5.2E3s	3.4E3s	3.7E3s	161.5s	137.7s	98.3s	97.4s
hard-gadm1	1.4E4s	1.2s	1.0E5s	3.5s	2.0E3s	3.4s	1.1E3s	3.0s

heuristics were enabled. (Obviously, we did not consider the implementation of [Nikolaou and Koubarakis, 2014] in our evaluation as it is not sound.) We chose to reason in a serial fashion, from smaller to bigger QCN, so as to stress how much more $\diamond$-consistency and the backtracking algorithm that utilizes it along with the heuristics in each reasoner benefit from reasoning with the smaller biconnected QCNs than reasoning with the initial large and loosely connected QCN, when both approaches are offered the same computational power. Thus, only one CPU core was used in our experiments.

Dataset and Measures We consider the *hard* network instances `hard-nuts`, `hard-adm1`, and `hard-gadm1` from [Nikolaou and Koubarakis, 2014] that comprise $\mathcal{NP}_8$ relations [Renz and Nebel, 2001] to utilize the whole reasoning engine of a reasoner. As noted earlier, the constraint graphs of these networks have the same characteristics as those presented in Tables 5.6 and 5.7. We use the CPU time as our measure.

Results The results are shown in Table 5.8 and make clear that our simple decomposition scheme aids the performance of each reasoner substantially, with the more apparent case being that of `hard-gadm1`, which is unsatisfiable. Networks `hard-nuts` and `hard-adm1` are satisfiable. In particular, GQR decides `gadm1` in ~ 4 hours, while GQR+ in 1.2 seconds, and similar results are obtained for the other reasoners too. When an inconsistency is detected in a QCN n that corresponds to some biconnected component of the constraint graph of an input QCN $\mathcal{N}$, each reasoner backtracks only within the search space defined by n, and considers a very small search-tree to either verify or dispute that inconsistency with respect to the search-tree that would have been obtained by the input QCN $\mathcal{N}$. Obviously, the time obtained for reasoner Solver+ is the time that it took it to serially reason with every QCN n, until it reached an unsatisfiable QCN (thus, assuring that the input QCN $\mathcal{N}$ is also unsatisfiable by Corollary 32 at page 153).

It is worth commenting on the performance of the reasoners with respect to network `hard-adm1`. Reasoners Sarissa+ and Phalanx$\triangledown$+ present a performance that is slightly better than that of reasoners Sarissa and Phalanx$\triangledown$ respectively. On the other hand, reasoners GQR+ and Phalanx+ present a performance that is slightly worse than that of reasoners GQR and Phalanx respectively. This is due to the fact that the maximum order among the biconnected components of the constraint graph of `adm1` is very close to the order of the entire graph itself (see Table 5.7). Thus, in such cases, the use of Algorithm 25 may not lead to drastically improved performance, while sometimes due to the randomness of the heuristics in a reasoner, even slightly worse performance may be observed, as in this particular case.

Finally, we note that the results presented in Table 5.8 do not take into account the time needed for decomposing the networks with Algorithm 29, but only the time needed for performing satisfiability checks on the networks. However, the time needed for decomposing `hard-nuts`, `hard-adm1`, and `hard-gadm1` was negligible, and does not change the results qualitatively. In particular, a simple Python script that makes use of the networkx[31] library was able to decompose

[31] `https://networkx.github.io/`

`hard-nuts`, `hard-adm1`, and `hard-gadm1` in 0.2, 1.4, and 7.6 seconds respectively.

5.8 Conclusion and Future Work

In this chapter, we presented our contributions with respect to qualitative constraint-based spatial and temporal reasoning, which involved novel and efficient algorithms that go beyond the state of the art algorithms for reasoning with qualitative constraint networks (QCNs). In particular, we defined new local consistency conditions and new algorithms for enforcing those conditions, which we compared both theoretically and experimentally to the local consistency conditions and their respective algorithms that were presented in Section 3.5.

Notably, our contributions ranged over the entire spectrum of fundamental reasoning problems in qualitative constraint-based spatial and temporal reasoning. Specifically, we contributed novel and effiecient techniques towards solving the *satisfiability problem*, the *minimal labeling problem*, and the *redundancy problem* of a given QCN. Furthermore, we addressed an issue in the literature regarding a non-sound approach that utilizes parallelism to check the satisfiability of RCC-8 networks. To this end, we provided the appropriate fixes for that approach, but also presented our own approach of a simple decomposition scheme that exploits the sparse and loosely connected structure of the constraint graphs of very large real-world QCNs and paves the way for efficient utilization of parallelism to solve all the aforementioned fundamental reasoning tasks.

Regarding future work, our contributions leave many options to be considered. Of particular interest is the method of directional algebraic closure and the related notion of directional $\diamond$-consistency presented in Section 5.3. Currently, with respect to that method and its related notion, we have already considered the satisfiability problem of QCNs restricted to a particular subclass of relations (viz., a *distributive* subclass of relations) in this thesis, and we have also made a first step towards using directional $\diamond$-consistency to establish $\diamond$-consistency in such QCNs in [Long *et al.*, 2016]. We would like to explore whether directional $\diamond$-consistency can be efficiently used as the backbone of a backtracking algorithm for checking the satisfiability of arbitrary qualitative constraint networks, i.e., networks defined over any of the relations of a qualitative constraint language. Our experimentation that took place in Section 5.4.4 suggests that we should be able to have better performance in some cases, as any such backtracking algorithm defined in the literature largely utilizes its core local consistency enforcing algorithm; on the other hand, the pruning capacity of directional $\diamond$-consistency is significantly limited when compared with the pruning capacity of partial $\diamond$-consistency or $\diamond$-consistency.

We would also like to explore the implication of directional $\diamond$-consistency in the novel algorithms that we presented here for the *minimal labeling problem* and the *redundancy problem*, as these problems exhibit functions that build on the local consistency enforcing algorithms used for checking the satisfiability of a given qualitative constraint network.

Another option to be considered is the implemention of online algorithms that can we used for qualitative spatial and temporal stream reasoning [Heintz and de Leng, 2014; de Leng and Heintz, 2016], i.e., evaluating spatial and temporal formulas over incrementally available streams of temporal states. To this end, the vertex-incremental partial $\diamond$-consistency algorithm that we presented in Section 5.2.2, viz., the iPWC algorithm, can come particularly handy alongside our hash table based adjacency list representation of a QCN presented in Section 5.4.3, as it can lead to efficient online partial $\diamond$-consistency algorithm implementations. It would be nice to have this functionality for arbitrary QCNs as well.

Other future work consists of further exploring and optimizing on the hierarchical structure

that real datasets present, as argued in [Koubarakis *et al.*, 2011]. In particular, it would be interesting to explore which relations are used more than others in real datasets and whether this could be of some use or not.

Finally, we would like to consider possible applications of our results in fields outside the field of qualitative constraint-based spatial and temporal reasoning, such as the field of *quantitative* constraint-based spatial and temporal reasoning. As a matter of fact, we have already done so with respect to the redundancy problem for instances of the Simple Temporal Problem (STP), which is a well-known and studied problem in the field of quantitative temporal reasoning [Dechter *et al.*, 1991]. In particular, with respect to the STP, and inspired from our contributions in this chapter, we have made a step towards dealing with redundant information in instances of the STP by simplifying their structure to a unique and minimal representation [Lee *et al.*, 2016]. As explained in [Lee *et al.*, 2016], the concise representation of STP instances can allow for more efficient scheduling, whilst retaining storage requirements to a minimum.

Chapter 6

Enriching Qualitative Spatio-Temporal Reasoning

6.1 Introduction

In this chapter, we present our contributions with respect to formalisms that combine spatial and temporal reasoning in an interrelated manner, which enrich the field of qualitative spatio-temporal reasoning in general.

In particular, we study the qualitative spatio-temporal logic that results by combining the propositional temporal logic (PTL) with a qualitative spatial constraint language, namely, the $\mathcal{L}_1$ logic that was presented in Section 4.2, and investigate the implication of the constraint properties of *compactness* and *patchwork* in qualitative spatio-temporal reasoning (cf. Section 3.6). We use these properties to strengthen results regarding the complexity of the satisfiability problem in $\mathcal{L}_1$, by replacing the stricter global consistency property used in literature and, consequently, generalizing to more qualitative spatial constraint languages.

Further, we identify fragments of the $\mathcal{L}_1$ logic that capture significant aspects of spatio-temporal change. In particular, we address the issue of periodical, and smoothness and continuity constraints between spatial configurations, and obtain results on their computational properties. Regarding periodicity, we use the properties of compactness and patchwork to strengthen related results that exist in the literature and were presented in Section 4.5, by re-establishing conditions that allow for tractability and, again, generalizing to a larger class of qualitative spatial constraint languages.

Moreover, we present a first semantic tableau method that given a $\mathcal{L}_1$ formula ϕ systematically searches for a model for ϕ. Our approach builds on Wolper's tableau method for PTL, while the ideas provided can be carried to other tableau methods for PTL as well. We prove the correctness of our tableau method for $\mathcal{L}_1$ using the aforementioned strengthened results regarding the satisfiability problem in $\mathcal{L}_1$.

Additionally, and with respect to the discussion in Sections 4.3 and 4.4, we investigate the task of ordering a temporal sequence of qualitative spatial configurations to meet certain transition constraints. This ordering is constrained by the use of conceptual neighbourhood graphs defined on qualitative spatial constraint languages. Specifically, we show that the problem of ordering a sequence of qualitative spatial configurations to meet such transition constraints is NP-complete for the the well-known languages of RCC-8, Interval Algebra, and Block Algebra. Based on this result, we also propose a framework where the temporal aspect of a sequence of qualitative spatial configurations is constrained by a Point Algebra network, and again show that the enhanced

problem is in NP when considering the aforementioned languages. These results lie within the area of Graph Traversal [Rosenkrantz *et al.*, 1977] and allow for many practical and diverse applications, such as identifying optimal routes in mobile robot navigation, modelling changes of topology in biological processes, and computing sequences of segmentation steps used in image processing algorithms.

The contributions to be presented in this chapter draw from the published works in [Sioutis *et al.*, 2015b; Sioutis *et al.*, 2015e; Sioutis *et al.*, 2014; Sioutis *et al.*, 2015c; Sioutis *et al.*, 2015a; Sioutis *et al.*, 2015d].

Finally, in Section 6.6 we conclude the chapter and give some directions for future work. In particular, we discuss the implication of using determined entities (constants) for a given qualitative spatial constraint language (cf. [Li *et al.*, 2013; Liu *et al.*, 2011]) and whether qualitative spatio-temporal reasoning can benefit from a recent advancement regarding the modal logic S5 [Salhi and Sioutis, 2015].

6.2 Revisiting the Satisfiability Problem in $\mathcal{L}_1$

In this section, we revisit a result regarding the satisfiability of $\mathcal{L}_1$ formulas in a ST-structure, using patchwork and compactness. These properties strengthen previous results, in that we do not longer need to restrict atomic QCNs to being globally consistent as in [Balbiani and Condotta, 2002; Demri and D'Souza, 2007], but we can consider atomic QCNs that have compactness and patchwork. As explained in Section 3.6, compactness and patchwork combined are less strict than global consistency alone.

Given a $\mathcal{L}_1$ formula ϕ, Balbiani and Condotta in [Balbiani and Condotta, 2002] show that the satisfiability of formula ϕ can be checked by characterizing a particular infinite sequence of finite satisfiable atomic QCNs representing an infinite consistent valuation of ϕ. Each of the QCNs of such a sequence represents a set of spatial constraints in a fixed-width window of time. The set of spatial constraints at point of time i, is given by the i-th QCN in the infinite sequence, and shares spatial constraints with the next QCN. Moreover, in such a sequence, there exists a point of time after which the corresponding QCNs replicate the same set of spatial constraints. The global consistency property is then used for the following two tasks:

(i) to prove that by considering all the QCNs of the aforementioned sequence we obtain a consistent set of constraints;

(ii) to prove that in such an infinite sequence, a sub-sequence which begins and ends with two QCNs representing the same set of spatial constraints can be reduced to just considering the first QCN.

In the sequel, we formally show that tasks (i) and (ii) can be performed using the properties of patchwork and compactness instead. As a consequence, we can generalize a result regarding the satisfiability of a $\mathcal{L}_1$ formula ϕ to a larger class of calculi than the previously considered in the literature.

We now introduce the two aforementioned tasks in the form of two propositions. Each proposition will be accompanied by a visual representation in the form of a figure to make the related discussion easier.

Proposition 51 *Let $V = \{v_0, \ldots, v_n\}$ be a set of entities, $w \geq 0$ an integer, and $\mathcal{S} = (\mathcal{N}_0 = (V_0, C_0), \mathcal{N}_1 = (V_1, C_1), \ldots)$ a countably infinite sequence of satisfiable atomic QCNs, such that:*

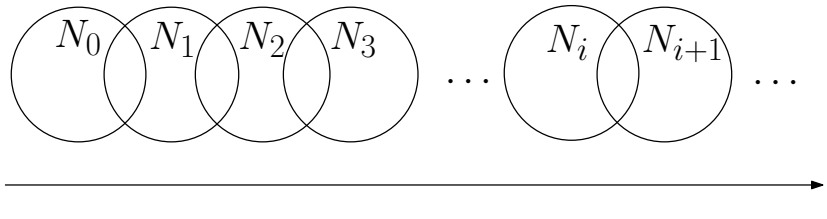


Figure 6.1: A countably infinite sequence of satisfiable atomic QCNs that agree on their common part

- *for each $i \geq 0$, V_i is defined by the set of variables $\{v_0^0,\ldots,v_n^0,\ldots,v_0^w,\ldots,v_n^w\}$,*
- *for each $i \geq 0$, for all $m, m' \in \{0,\ldots,n\}$, and for all $k, k' \in \{1,\ldots,w\}$, $C_i(v_m^k, v_{m'}^{k'}) = C_{i+1}(v_m^{k-1}, v_{m'}^{k'-1})$.*

We have that if the constraint language considered has compactness and patchwork for satisfiable atomic QCN*s, then $\mathcal{S}$ defines a consistent set of qualitative constraints.*

Proof. Given $\mathcal{N}_i$, we rewrite its set of variables to $\{v_0^i,\ldots,v_n^i,\ldots,v_0^{w+i},\ldots,v_n^{w+i}\}$. Then, by patchwork we can assert that for each integer $k \geq 0$, $\bigcup_{k\geq i\geq 0}\mathcal{N}_i$ is a consistent set of qualitative constraints. Suppose though, that $\bigcup_{i\geq 0}\mathcal{N}_i$ is an inconsistent set. By compactness we know that there exists an integer $k' \geq 0$ for which $\bigcup_{k'\geq i\geq 0}\mathcal{N}_i$ is inconsistent. This is a contradiction. Thus, $\mathcal{S}$ defines a consistent set of qualitative constraints. ⊣

A visual representation of Proposition 51 is presented in Figure 6.1. The second proposition follows.

Proposition 52 *Let $V = \{v_0,\ldots,v_n\}$ be a set of entities, $w \geq 0, t > t' \geq 0$ three integers, and $\mathcal{S} = (\mathcal{N}_0 = (V_0, C_0), \mathcal{N}_1 = (V_1, C_1), \ldots)$ a countably infinite sequence of satisfiable atomic* QCN*s, such that:*

- *for each $i \geq 0$, V_i is defined by the set of variables $\{v_0^0,\ldots,v_n^0,\ldots,v_0^w,\ldots,v_n^w\}$,*
- *for each $i \geq 0$, for all $m, m' \in \{0,\ldots,n\}$, and for all $k, k' \in \{1,\ldots,w\}$, $C_i(v_m^k, v_{m'}^{k'}) = C_{i+1}(v_m^{k-1}, v_{m'}^{k'-1})$,*
- *for all $m, m' \in \{0,\ldots,n\}$ and all $k, k' \in \{0,\ldots,w\}$, $C_{t'}(v_m^k, v_{m'}^{k'}) = C_t(v_m^k, v_{m'}^{k'})$.*

Let $\mathcal{S}' = (\mathcal{N}_0' = (V_0', C_0'), \mathcal{N}_1' = (V_1', C_1'), \ldots)$ be the infinite sequence defined by:

- *for all $i \in \{0,\ldots,t'\}$, $\mathcal{N}_i' = \mathcal{N}_i$,*
- *for all $i > t'$, $V_i' = V_i$, and for all $m, m' \in \{0,\ldots,n\}$ and all $k, k' \in \{0,\ldots,w\}$, $C_i'(v_m^k, v_{m'}^{k'}) = C_{i+(t-t')}(v_m^k, v_{m'}^{k'})$.*

We have that if the constraint language considered has compactness and patchwork for satisfiable atomic QCN*s, then $\mathcal{S}'$ defines a consistent set of qualitative constraints.*

Proof. We have $\mathcal{N}_i$ which is a satisfiable QCN for all $i \geq 0$. From this, we can deduce that $\mathcal{N}_i'$ is a satisfiable QCN for all $i \geq 0$, since $\mathcal{N}_i' = \mathcal{N}_i$ for all $i \in \{0,\ldots,t'\}$, $V_i' = V_i$ for all $i > t'$, and $C_i'(v_m^k, v_{m'}^{k'}) = C_{i+(t-t')}(v_m^k, v_{m'}^{k'})$ for all $m, m' \in \{0,\ldots,n\}$ and all $k, k' \in \{0,\ldots,w\}$. By

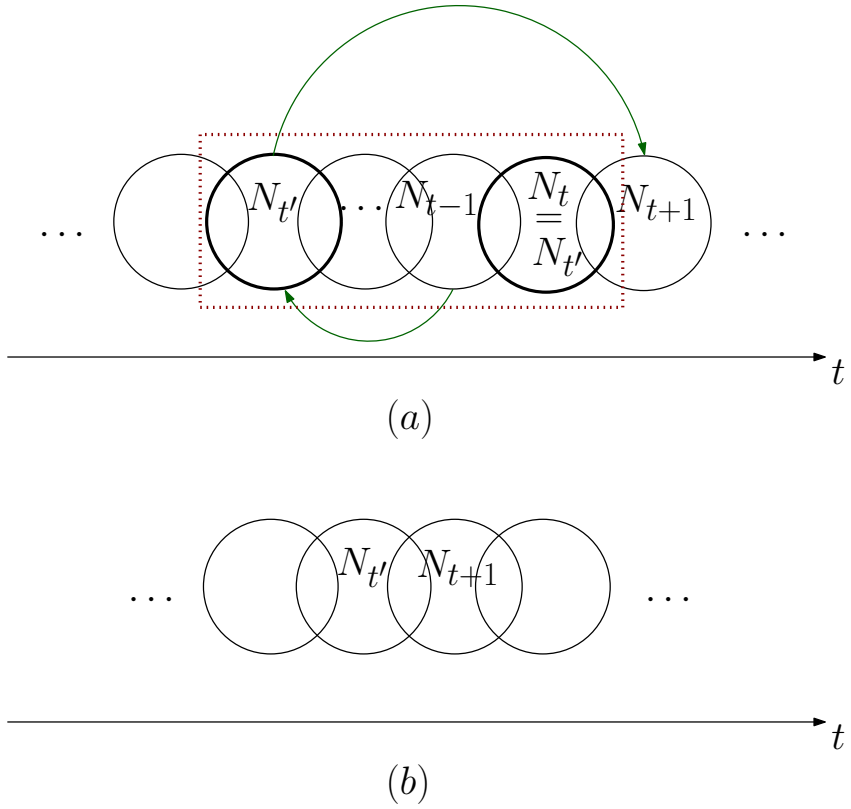


Figure 6.2: A countably infinite sequence of satisfiable atomic QCNs that contains a sub-sequence which begins and ends with two QCNs representing the same set of spatial constraints (i.e., a sub-sequence which defines a loop between two QCNs) (a); we can reduce the sub-sequence to just considering the first QCN and patch it with the QCN following the sub-sequence (b)

Proposition 51 we can deduce that $\mathcal{S}'$ defines a consistent set of qualitative constraints, as it satisfies the conditions listed in that proposition. ⊣

A visual representation of Proposition 52 is presented in Figure 6.2. We now can obtain the following result:

Theorem 33 *Checking the satisfiability of a* $\mathcal{L}_1$ *formula* ϕ *in a* ST*-structure is* PSPACE*-complete if the qualitative spatial constraint language considered has compactness and patchwork for satisfiable atomic* QCN*s.*

Proof. (Sketch) Consider the approach in [Balbiani and Condotta, 2002] where a proof of PSPACE-completeness is given for a logic that considers qualitative constraint languages for which satisfiable atomic QCNs are globally consistent (see Theorem 1 in [Balbiani and Condotta, 2002]). To be able to replace the use of global consistency with the use of patchwork and compactness, we need to use Propositions 51 and 52 in the proofs of Lemmas 3 and 4 in [Balbiani and Condotta, 2002]. The interested reader can verify that the aforementioned proofs make use of global consistency to perform exactly the tasks described by Propositions 51 and 52. Since these propositions build on compactness and patchwork, we can prove PSPACE-completeness using these properties instead. ⊣

Theorem 33 allows us to consider more calculi than the ones considered in literature for which

the combination with PTL yields PSPACE-completeness. Due to the lack of global consistency for RCC-8 [Renz and Ligozat, 2005], in [Gabelaia *et al.*, 2003] the authors restrict themselves to a very particular domain interpretation of RCC-8 to prove that the $\mathcal{ST}_1^-$ logic is PSPACE-complete; the $\mathcal{ST}_1^-$ logic is the $\mathcal{L}_\mathbf{1}$ logic when the considered qualitative constraint language is RCC-8. $\mathcal{L}_\mathbf{1}$ does not rely on the semantics of the qualitative constraint language used, but rather on the constraint properties of *compactness* and *patchwork* [Lutz and Milicic, 2007]. Therefore, $\mathcal{L}_\mathbf{1}$ is by default able to consider all calculi that have these properties, such as RCC-8 [Randell *et al.*, 1992], Cardinal Direction Calculus (CDC) [Frank, 1991; Ligozat, 1998], Block Algebra (BA) [Balbiani *et al.*, 2002], and even Interval Algebra (IA) [Allen, 1983] when viewed as a spatial calculus. The most notable languages that have patchwork and compactness are listed in [Huang, 2012].

In particular, due to Theorem 33 and Propositions 16 (at page 54), 3 (at page 34), and 17 (at page 54) we obtain the following result:

Corollary 35 *Checking the satisfiability of a $\mathcal{L}_\mathbf{1}$ formula in a* ST*-structure is* PSPACE*-complete for the qualitative constraint languages of Point Algebra, Interval Algebra, Cardinal Direction Calculus, Block Algebra, and* RCC*-8.*

It is interesting to note that Theorem 33 can also be proved by following the approach of Demri and D'Souza in [Demri and D'Souza, 2007; Demri and D'Souza, 2002]. In particular, in [Demri and D'Souza, 2007] it is shown that one can check the satisfiability of $\mathcal{L}_\mathbf{1}$ formulas by a simple, almost modular, combination of the satisfiability checking algorithm for PTL [Sistla and Clarke, 1985] and any satisfiability checking algorithm for the qualitative constraint language at hand. It is modular in that an automaton can be constructed as the intersection of two separate automata, one for the underlying temporal logic language PTL, and the other one for the qualitative constraint language under consideration. Then, if the qualitative constraint language under consideration has the global consistency property for atomic networks, the satisfiability problem for $\mathcal{L}_\mathbf{1}$ remains PSPACE-complete [Demri and D'Souza, 2007].

Let us delve into more detail about the approach of Demri and D'Souza in [Demri and D'Souza, 2007], and let the automaton for the underlying temporal logic language PTL be denoted by $\mathcal{A}_\phi^t$ and the automaton for the qualitative constraint language under consideration be denoted by $\mathcal{A}_\phi^s$. Then, $\mathcal{A}_\phi^t$ is a Vardi-Wolper automaton [Vardi and Wolper, 1986] and $\mathcal{A}_\phi^s$ a Büchi automaton $(Q, q_0, \rightarrow, F)$, where Q is the set of *maximally* consistent sets of atomic constraints over variables in ϕ,[32] q_0 a separate start state, $\rightarrow$ a transition relation that starting initially from q_0 moves to a next state by patching together the set of atomic constraints provided by each state into a unique atomic consistent network, and $F = Q$. To prove PSPACE-completeness for $\mathcal{L}_\mathbf{1}$, Demri and D'Souza show that it suffices to be able to compute the transitions of $\mathcal{A}_\phi^s$ in PSPACE. They are able to do so by using the global consistency property as follows. Given the initial atomic constraint network corresponding to state q_0, they consider a valuation (i.e., a solution) of its set of variables. Whenever they move to a next state and augment the atomic constraint network with a new set of atomic constraints while retaining consistency locally, they know that the augmented atomic constraint network will be satisfiable as the valuation of the previous state can be extended to a valuation of the new state due to the property of global consistency. However, we proved earlier that consistency of such sequences of constraint networks can be retained through the properties of patchwork and compactness instead (cf. Proposition 51). Hence, the main decidability result of Theorem 4 in [Demri and D'Souza, 2007, Chapter 4] can be revised accordingly.

[32] In a sense, we have all possible satisfiable atomic QCNs over a given set of variables.

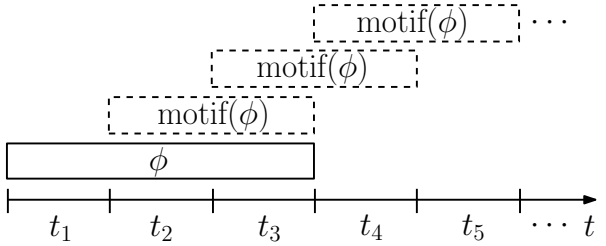


Figure 6.3: A $\mathcal{L}_{\mathsf{UPQCN}}$ formula ϕ over timeline t

6.3 Capturing Spatio-Temporal Behaviour in $\mathcal{L}_1$

In this section, we use particular fragments of the $\mathcal{L}_1$ logic to capture properties that deal with spatial behaviour in a temporal universe, such as periodicity, and continuity and smoothness. Note that the *temporal window size* (or simply *size*) of a $\mathcal{L}_1$ formula ϕ, denoted by $|\phi|$, is defined inductively as follows: $P(\bigcirc^l v, \bigcirc^m v') = \max\{l, m\}$; $|\neg\phi| = |\phi|$; $|\phi \vee \psi| = |\phi\,\mathcal{U}\,\psi| = \max\{|\phi|, |\psi|\}$. The size of a set of $\mathcal{L}_1$ formulas $\chi = \{\phi, \psi, \ldots\}$, will be the maximum size among its formulas, i.e., $|\chi| = \max\{|\phi|, |\psi|, \ldots\}$. We first define a sublanguage of $\mathcal{L}_1$ that will be of use in studying our fragments. In particular, let $\mathcal{L}_{\mathsf{QCN}}$ be the sublanguage defined by statements of the form $\phi = \bigwedge(R(\bigcirc^n x_i, \bigcirc^m x_j))$, where $R \in 2^{\mathsf{B}}$. It is easy to see that a $\mathcal{L}_{\mathsf{QCN}}$ formula ϕ can be expressed by a QCN $\mathcal{N} = (V, C)$ as follows. If formula ϕ comprises the set of variables $\{x_0, \ldots, x_k\}$, V will be the set $\{v_0^0, \ldots, v_k^0, \ldots, v_0^{|\phi|}, \ldots, v_k^{|\phi|}\}$. Then, $C(v_i^n, v_j^m) = R$ if $R(\bigcirc^n x_i, \bigcirc^m x_j)$ is a subformula of ϕ, and $C(v_i^n, v_j^m) = (\mathsf{B}$ if $v_i^n \neq v_j^m$ else $\{\mathsf{Id}\})$ otherwise. Expressing a QCN as a $\mathcal{L}_{\mathsf{QCN}}$ formula is also straight-forward. If $\mathcal{N} = (V, C)$ is a QCN, the $\mathcal{L}_{\mathsf{QCN}}$ formula ϕ corresponding to $\mathcal{N}$ is defined as $\phi = \bigwedge(C(u, v))$, where $u, v \in V$. As such, checking the satisfiability of a $\mathcal{L}_{\mathsf{QCN}}$ formula in a ST-structure has the same complexity as the satisfiability problem for the corresponding QCN in the considered qualitative constraint language. In particular, checking the satisfiability of a $\mathcal{L}_{\mathsf{QCN}}$ formula in a ST-structure is NP-complete for RCC-8, Cardinal Direction Calculus, Block Algebra, and Interval Algebra, and can be performed in polynomial time for Point Algebra.

6.3.1 Spatio-Temporal Periodicity

In this section, we capture the notion of an ultimately periodic qualitative constraint network (UPQCN) [Condotta *et al.*, 2005] that was presented in Section 4.5. As a brief reminder, a UPQCN is a temporalized QCN that evolves over time with a recurrent pattern. Specifically, a UPQCN is a QCN that extends over a fixed-width window of time, and contains a smaller QCN as a recurrent pattern evolving over time. We can define fragment $\mathcal{L}_{\mathsf{UPQCN}}$ to capture periodicity as follows.

Definition 43 *Given a* UPQCN $\mathcal{U} = (V, C, t_{min}, t_{max})$*, fragment* $\mathcal{L}_{\mathsf{UPQCN}}$ *comprises formulas of the following form:*

$$\phi \wedge \bigcirc^m \Box\phi_{motif(\mathcal{U})}$$

where ϕ *is a* $\mathcal{L}_{\mathsf{QCN}}$ *formula that corresponds to* $\mathcal{U}$*,* $\phi_{motif(\mathcal{U})}$ *is a* $\mathcal{L}_{\mathsf{QCN}}$ *formula that corresponds to the periodical part of* $\mathcal{U}$*, namely,* $motif(\mathcal{U})$*, and* $m = t_{min}$ *defines the beginning of the recurrent pattern.*

Clearly, a UPQCN $\mathcal{U}$ is satisfiable if and only if the $\mathcal{L}_{\mathsf{UPQCN}}$ formula representing it is satisfiable. Given a UPQCN $\mathcal{U}$, the relationship between $\mathcal{L}_{\mathsf{QCN}}$ formulas ϕ and $\phi_{motif(\mathcal{U})}$, as provided

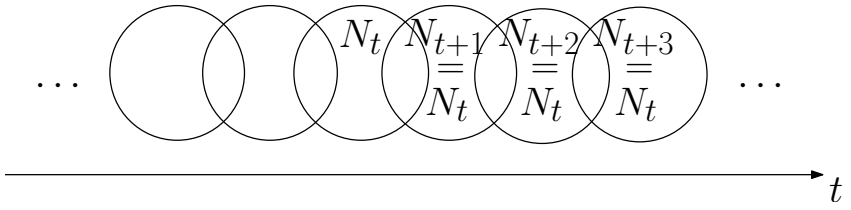


Figure 6.4: A countably infinite sequence of not trivially inconsistent and $\diamond$-consistent QCNs, where there exists a point of time t after which the QCNs in the sequence represent the same set of constraints

in the aforementioned definition, is depicted in Figure 6.3. As ϕ stretches over the timeline, it forms a pattern, denoted by $\phi_{\text{motif}(\mathcal{U})}$, which holds at every consecutive period of time after its first appearance, as clearly observed in the figure.

As an example, let us consider the $\mathcal{L}_{\mathsf{UPQCN}}$ formula $PO(X,Y) \wedge EC(X, \bigcirc X) \wedge \bigcirc \Box DC(X, \bigcirc Y)$. Assuming a timeline t, formula ϕ corresponds to a UPQCN $\mathcal{U}$ that extends over three consecutive points of time $t = 0$, $t = 1$, and $t = 2$. At $t = 0$ we have the constraint $PO(X,Y)$, between points of time $t = 0$ and $t = 1$ we have the constraint $EC(X, \bigcirc X)$, and between points of time $t = 1$ and $t = 2$ we have the constraint $DC(X, \bigcirc Y)$. It is easy to see that due to the $\Box$ operator, constraint $DC(X, \bigcirc Y)$ must hold over the period defined by points of time $t = 1$ and $t = 2$, but also over all consecutive periods of time in t. In fact, $DC(X, \bigcirc Y)$ is the motif of $\mathcal{U}$, captured by $\phi_{\text{motif}(\mathcal{U})}$, where $m = 1$ in our example $\mathcal{L}_{\mathsf{UPQCN}}$ formula.

The main result of [Condotta *et al.*, 2005] concerns the satisfiability problem for a UPQCN where the qualitative constraints belong to a class for which all $\diamond$-consistent and not trivially inconsistent QCNs are globally consistent. More precisely, it was shown that the satisfiability of a UPQCN $\mathcal{U}$ can be checked by characterizing a particular infinite sequence of finite $\diamond$-consistent and not trivially inconsistent QCNs representing an infinite consistent valuation of $\mathcal{U}$. Each of the QCNs of such a sequence represents a set of spatial constraints in a fixed-width window of time. The set of spatial constraints at point of time i, is given by the i-th QCN in the infinite sequence, and shares spatial constraints with the next QCN. Moreover, in such a sequence, there exists a point of time after which every QCN replicates the same set of spatial constraints with the previous QCN in the sequence. Global consistency was then used to prove that by considering all the QCNs of the aforementioned sequence we obtain a consistent set of constraints. We can generalize the result of [Condotta *et al.*, 2005] with the following proposition:

Proposition 53 *Let $V = \{v_0, \ldots, v_n\}$ be a set of entities, $w \geq 0$, $t \geq 0$ two integers, and $\mathcal{S} = (\mathcal{N}_0 = (V_0, C_0), \mathcal{N}_1 = (V_1, C_1), \ldots)$ a countably infinite sequence of not trivially inconsistent and $\diamond$-consistent QCNs, such that:*

- *for each $i \geq 0$, V_i is defined by the set of variables $\{v_0^0, \ldots, v_n^0, \ldots, v_0^w, \ldots, v_n^w\}$,*
- *for each $i \geq 0$, for all $m, m' \in \{0, \ldots, n\}$, and for all $k, k' \in \{1, \ldots, w\}$, $C_i(v_m^k, v_{m'}^{k'}) = C_{i+1}(v_m^{k-1}, v_{m'}^{k'-1})$,*
- *for all $m, m' \in \{0, \ldots, n\}$, all $k, k' \in \{0, \ldots, w\}$, and all $t' > t$, $C_{t'}(v_m^k, v_{m'}^{k'}) = C_t(v_m^k, v_{m'}^{k'})$.*

We have that if the qualitative spatial constraint language considered has compactness, patchwork for not trivially inconsistent and $\diamond$-consistent QCNs, and $\diamond$-consistency which implies satisfiability, then $\mathcal{S}$ defines a consistent set of qualitative constraints.

Proof. Since $\diamond$-consistency implies satisfiability, for each $i \geq 0$ we have that $\mathcal{N}_i$ is a satisfiable QCN. Given $\mathcal{N}_i$, we rewrite its set of variables to $\{v_0^i,\ldots,v_n^i,\ldots,v_0^{w+i},\ldots,v_n^{w+i}\}$. Then, by patchwork we can assert that for each integer $k \geq 0$, $\bigcup_{k\geq i\geq 0}\mathcal{N}_i$ is a consistent set of qualitative constraints. Suppose though, that $\bigcup_{i\geq 0}\mathcal{N}_i$ is an inconsistent set. By compactness we know that there exists an integer $k' \geq 0$ for which $\bigcup_{k'\geq i\geq 0}\mathcal{N}_i$ is inconsistent. This is a contradiction. Thus, $\mathcal{S}$ defines a consistent set of qualitative constraints. ⊣

A visual representation of Proposition 53 is presented in Figure 6.4. Using Proposition 53 we can prove the following theorem based on the fact that a $\mathcal{L}_{\mathsf{UPQCN}}$ defines a sequence of QCNs such as the one shown in Figure 6.4:

Theorem 34 *The satisfiability problem for a $\mathcal{L}_{\mathsf{UPQCN}}$ formula that corresponds to a* UPQCN *defined over a subclass of relations of a qualitative constraint language that has compactness and patchwork for not trivially inconsistent and $\diamond$-consistent* QCN*s defined over that subclass of relations can be decided in polynomial time.*

Then, due to Propositions 16 (at page 54) and 17 (at page 54) and Theorem 34 we obtain the following result:

Corollary 36 *The satisfiability problem for a $\mathcal{L}_{\mathsf{UPQCN}}$ formula that corresponds to a* UPQCN *of Point Algebra, Interval Algebra, Cardinal Direction Calculus, Block Algebra, or* RCC-8 *defined over one of the classes $\mathcal{P}_{\mathsf{PA}}$, $\mathcal{P}_{\mathsf{CDC}}$, $\mathcal{H}_{\mathsf{IA}}$, $\mathcal{H}^n_{\mathsf{IA}}$, or $\hat{\mathcal{H}}_8$, $\mathcal{C}_8$, and $\mathcal{Q}_8$ respectively can be decided in polynomial time.*

Theorem 34 is a significant strengthening of the main result obtained in [Condotta *et al.*, 2005], as we no longer need to restrict ourselves to a small class of relations satisfying global consistency (if such a class exists), but we can use a maximal tractable subclass of relations for the considered calculi here. For example, for RCC-8 there does not exist a class of relations containing all singleton relations that satisfies global consistency (as explained in Section 3.6 and stated in [Renz and Ligozat, 2005]), but the class of relations satisfying patchwork and compactness can be any of its maximal tractable subclasses $\hat{\mathcal{H}}_8, \mathcal{C}_8$, and $\mathcal{Q}_8$ [Renz, 1999], which comprise up to $\sim 60\%$ of the whole set of RCC-8 relations.

6.3.2 Spatio-Temporal Smoothness and Continuity

In [Westphal *et al.*, 2013] the authors study the problem of transition constraints in Point Algebra (PA) [van Beek, 1992]. In particular, they take a relational approach to the problem and define global constraints that capture smoothness and continuity. Here, we will make a similar contribution for spatio-temporal logics, as we will define statements that can capture smoothness and continuity within the context of $\mathcal{L}_1$.

Smoothness and continuity in a qualitative spatial constraint language can be encoded by a *conceptual neighborhood graph* [Freksa, 1991], the formal definition of which is presented in Section 4.3. In short, conceptual neighbourhood graphs capture conceptually neighbouring relations in any given qualitative constraint language. (As an example, a conceptual neighbourhood graph of RCC-8 is depicted in Figure 4.2 in Section 4.3). For instance, in RCC-8 the base relations DC and EC are conceptually proximal with respect to a pair of entities (v, v'), since a movement of the spatial entity v towards spatial entity v' may cause a direct transition from relation DC to relation EC; on the other hand, and again with respect to the pair of entities (v, v'), the relations DC and PO are not conceptually proximal since a transition between

those relations must go through relation EC. A subset of conceptual neighbourhood graphs for various qualitative constraint languages is found in [Freksa, 1991; Santos and Moreira, 2009; Egenhofer, 2010].

Definition 44 *Given a conceptual neighbourhood graph* $\Gamma = (\mathsf{B}, E)$*, the* conceptual neighborhood *of a vertex* $b \in \mathsf{B}$ *is the set* $N_C(b) = \{b' \mid \{b, b'\} \in E\}$.

We can capture transition constraints in the $\mathcal{L}_{\mathsf{QCN}}$ logic, by defining a particular formula ϕ_Γ comprising certain statements as follows.

Definition 45 *Given the set of spatial variables* V *of a* $\mathcal{L}_{\mathsf{QCN}}$ *formula, and a conceptual neighbourhood graph* $\Gamma = (\mathsf{B}, E)$*, formula* ϕ_Γ *is defined for all* $(v, v') \in V \times V$ *as a conjunction of the following* $\mathcal{L}_1$ *statements:*

$$\bigwedge_{b \in \mathsf{B}} \Box(b(v, v') \rightarrow \bigcirc(\bigvee_{b' \in N_C(b)} (b'(v, v'))))$$

We can obtain the following theorem:

Theorem 35 *Given a* $\mathcal{L}_{\mathsf{QCN}}$ *formula* ϕ *defined on a qualitative spatial constraint language, and a conceptual neighbourhood graph* Γ *of that language, checking the satisfiability of formula* $\phi \wedge \phi_\Gamma$ *in a* ST*-structure is in* NP *if the satisfiability problem for the corresponding* QCN *of* ϕ *in that language is in* NP*.*

Proof. Formula ϕ_Γ can be seen as a set of integrity constraints over each pair of the same spatial entities appearing at adjacent points of time. Suppose that you are provided with a model of formula ϕ, i.e., a solution of the QCN expressed by ϕ (see introduction of Section 6.3); the validity of that model can be verified in polynomial time. The length of the timeline is defined by $|\phi|$, yielding $|\phi| + 1$ points of time. Hence, we can have at most $|\phi|$ pairs of adjacent time points. For each such pair we can check in $O(|V|^2)$ time if the relations that hold between each pair of the same spatial entities appearing at the adjacent points of time satisfy the transition constraints with respect to formula ϕ_Γ, where V is the set of variables in ϕ. Thus, we will need an extra $O(|\phi||V|^2)$ time for deciding satisfiability of $\phi \wedge \phi_\Gamma$. Note that the $\Box$ operator propagates the transition constraints indefinitely, but, due to compactness and its implication regarding infinite sequences of finite satisfiable extensions of a network, we only need to propagate the constraints up to point of time $|\phi|$, and assume to always have the same satisfiable valuation afterwards. $\dashv$

Then, due to Theorem 35 and the discussion in Section 3.3.1 we can obtain the following result:

Corollary 37 *Given a* $\mathcal{L}_{\mathsf{QCN}}$ *formula* ϕ *defined on* RCC*-8, Cardinal Direction Calculus, Interval Algebra, or Block Algebra, and a conceptual neighbourhood graph* Γ *of the considered language, checking the satisfiability of formula* $\phi \wedge \phi_\Gamma$ *in a* ST*-structure is* NP*-complete.*

In the particular case of Point Algebra, we can infer from Theorem 35 that checking the satisfiability of formula $\phi \wedge \phi_\Gamma$ in a ST-structure, where formula ϕ is defined on Point Algebra and Γ is some conceptual neighbourhood graph of Point Algebra, is also in NP. This is because, although checking the satisfiability of a QCN of Point Algebra can be done in polynomial time, introducing transition constraints can impact that polynomial complexity. In particular, the complexity depends on the conceptual neighbourhood graph to be used. In the case where the

conceptual neighbourhood graph of Point Algebra considered is a complete graph for instance, it is clear that the satisfiability problem of a $\mathcal{L}_{\mathsf{QCN}}$ formula defined on Point Algebra and augmented with the corresponding set of transition constraints remains polynomial-time decidable.

We proceed with a small example of a spatio-temporal formula ϕ for which we will analyze its different components and the steps taken to decide its satisfiability. Let $\phi = NTPP(X,Y) \wedge \bigcirc TPP(X,Y) \wedge \bigcirc^2 PO(X,Y) \wedge EC(X,\bigcirc^2 X) \wedge NTPP(X,\bigcirc Y)$ be a spatio-temporal formula. (Note that it would be equivalent to write ϕ as $NTPP(X,Y) \wedge TPP(\bigcirc X,\bigcirc Y) \wedge PO(\bigcirc^2 X,\bigcirc^2 Y) \wedge EC(X,\bigcirc^2 X) \wedge NTPP(X,\bigcirc Y)$.) The size of the formula, i.e., the temporal window size, is $|\phi| = 2$, thus, it yields 3 distinct points of time t_1, t_2, t_3 totally ordered in the timeline t. At t_1, region X is a non-tangential proper part of region Y, i.e., $NTPP(X,Y)$. At time t_2, region X is a tangential proper part of region Y, i.e., $TPP(X,Y)$. And, at time t_3, region X partially overlaps region Y, i.e., $PO(X,Y)$. We can interpret these constraints as a motion of region X from the interior to the exterior of region Y. Further, we have that region X at time t_1 is externally connected to region X at time t_3, i.e., $EC(X,\bigcirc^2 X)$, which indeed implies a motion in space as time goes by for region X, and we also have that region X at time t_1 is a non-tangential proper part of region Y at time t_2, i.e., $NTPP(X,\bigcirc Y)$, which implies that region Y has not moved in the first two points of time, at least relatively and with respect to region X. The atomic qualitative constraint networks specified at each point of time are consistent, they are essentially base relation constraints over two regions. The constraints specified between regions at different points of time, viz., $EC(X,\bigcirc^2 X)$ and $NTPP(X,\bigcirc Y)$, are also consistent. Finally, for all pairs of adjacent points of times, viz., (t_1,t_2) and (t_2,t_3), the transition constraints are also respected.

6.4 Semantic tableau for $\mathcal{L}_1$

Proof theory is a syntactic in nature mathematical logic that represents proofs as formal mathematical objects and facilitates their analysis through mathematical techniques. In particular, proofs are typically presented as inductively-defined data structures, such as graphs or trees, which are constructed according to the axioms and rules of inference of the logical system at hand. As such, proof theory can be described as the study of the general structure of mathematical proofs and of *demonstrative arguments* (i.e., arguments whose conclusions follow necessarily from the assumptions made) as they are encountered in logic. Such demonstrative arguments lie in the basis of the work of Aristotle in [Aristotle, 2004]:

> *"if to understand something is what we have posited it to be, then demonstrative understanding in particular must proceed from items which are true and primitive and immediate and more familiar than and prior to and explanatory of the conclusions. (In this way the principles will also be appropriate to what is being proved.) There can be a deduction even if these conditions are not met, but there cannot be a demonstration–for it will not bring about understanding."*

Hence, Aristotle defines the deductive science as a science that is organised around a number of basic concepts that are assumed understood without further explanation, and a number of basic truths or axioms that can be seen as true immediately. A subdiscipline of proof theory that studies the specifics of proof calculi, i.e., formal systems that use a common style of formal inference for their inference rules, is structural proof theory. An important research topic in structural theory includes the *semantic tableau method* [Beth, 1955], which applies the central ideas of structural proof theory to provide decision procedures and semi-decision procedures for

a wide range of logics. More specifically, the semantic tableau method is a decision procedure for sentential and related logics, and a proof procedure for formulas of first-order logic. As we will also see in this section, the tableau method can also determine the satisfiability of finite sets of formulas of various logics. Notably, the tableau method is the most popular proof procedure for modal logics [Girle, 2000].

In this section, we present a semantic tableau method that given a $\mathcal{L}_\mathbf{1}$ formula ϕ systematically searches for a model for ϕ. The method builds on the tableau method for PTL of Wolper [Wolper, 1985], and makes use of the results of Section 6.2 to ensure soundness and completeness. It is important to note, that Wolper's method serves as the basis to illustrate our line of reasoning, and that the techniques to be presented can be carried to other more efficient tableau methods for PTL as well, such as the systematic semantic tableaux for PTL presented in [Gaintzarain *et al.*, 2008].

6.4.1 Rules for Constructing a Semantic Tableau

The decomposition rules of the temporal operators are based on the following identities, which are called *eventualities* (where $\Box$ abbreviates $\neg\Diamond\neg$):

- $\Diamond\phi \equiv \phi \vee \bigcirc\Diamond\phi$
- $\phi \,\mathcal{U}\, \psi \equiv \psi \vee (\phi \wedge \bigcirc(\phi \,\mathcal{U}\, \psi))$

Note that decomposing eventualities can lead to an infinite tableau. However, we will construct a finite tableau by identifying nodes that are labeled by the same set of formulas, thus, ensuring that infinite periodicity will not exist. To test a $\mathcal{L}_\mathbf{1}$ formula ϕ for satisfiability, we will construct a directed graph. Each node n of the graph will be labeled by a set of formulas, and initially the graph will contain a single node, labeled by $\{\phi\}$. Similarly to Wolper [Wolper, 1985], we distinguish between *elementary* and *non-elementary* formulas:

Definition 46 *A $\mathcal{L}_\mathbf{1}$ formula is elementary if its main connective is $\bigcirc$ (viz., $\bigcirc$-formula), or if it corresponds to a base relation $P \in \mathsf{B}$.*

Then, the construction of the graph proceeds by using the following decomposition rules which map each non-elementary formula ϕ into a set of sets of formulas:

- $\neg P(\bigcirc^n v, \bigcirc^m v') \rightarrow \{\{P'(\bigcirc^n v, \bigcirc^m v')\} \mid P' \in \mathsf{B} \setminus \{P\}\}$
- $\neg\neg\phi \rightarrow \{\{\phi\}\}$
- $\neg \bigcirc \phi \rightarrow \{\{\bigcirc\neg\phi\}\}$
- $\phi \wedge \psi \rightarrow \{\{\phi, \psi\}\}$
- $\neg(\phi \wedge \psi) \rightarrow \{\{\neg\phi\}, \{\neg\psi\}\}$
- $\Diamond\phi \rightarrow \{\{\phi\}, \{\bigcirc\Diamond\phi\}\}$
- $\neg\Diamond\phi \rightarrow \{\{\neg\phi, \neg \bigcirc \Diamond\phi\}\}$
- $\phi \,\mathcal{U}\, \psi \rightarrow \{\{\psi\}, \{\phi, \bigcirc(\phi \,\mathcal{U}\, \psi)\}\}$
- $\neg(\phi \,\mathcal{U}\, \psi) \rightarrow \{\neg\psi, \neg\phi \vee \neg \bigcirc (\phi \,\mathcal{U}\, \psi)\}$

During the construction, we *mark* formulas to which a decomposition rule has been applied to avoid decomposing the same formula twice. If ψ is a formula, $\psi*$ denotes ψ marked.

6.4.2 Systematic Construction of a Semantic Tableau

A tableau $\mathcal{T}$ can be seen as a directed graph where each of its nodes n is labeled with a set of formulas $\mathcal{T}(n)$. The root node is labeled with the singleton set $\{\phi\}$ for the $\mathcal{L}_\mathbf{1}$ formula ϕ whose satisfiability we wish to check. The children of the nodes are obtained by applying the rules presented in Section 6.4.1.

Given a set of $\mathcal{L}_\mathbf{1}$ formulas χ over the set of variables $\{x_0, \ldots, x_l\}$, we denote by $expandVars(\chi)$ the set $\{\bigcirc^0 x_0, \ldots, \bigcirc^0 x_l, \ldots, \bigcirc^{|\chi|} x_0, \ldots, \bigcirc^{|\chi|} x_l\}$. We first define a translation of a node of a tableau to a QCN.

Definition 47 *Let n be a node of a tableau $\mathcal{T}$ for a $\mathcal{L}_\mathbf{1}$ formula ϕ, and $\{x_0, \ldots, x_l\}$ the set of variables in ϕ. Then, $\mathcal{N}(n)$ will denote the* QCN $= (V, C)$*, where $V = \{v_0^0, \ldots, v_l^0, \ldots, v_0^{|\phi|}, \ldots, v_l^{|\phi|}\}$, and $C(v_m^k, v_{m'}^{k'}) = \{P(\bigcirc^k x_m, \bigcirc^{k'} x_{m'})\}$ if $P(\bigcirc^k x_m, \bigcirc^{k'} x_{m'}) \in \mathcal{T}(n)$, and $C(v_m^k, v_{m'}^{k'}) =$ (B if $v_m^k \neq v_{m'}^{k'}$ else $\{\mathsf{Id}\}$) otherwise, $\forall m, m' \in \{0, \ldots, l\}$ and $\forall k, k' \in \{0, \ldots, |\phi|\}$.*

Let us also define the notions of a *state* and a *pre-state*, which we will be referring to a lot in what follows.

Definition 48 *A node n that contains only elementary and marked formulas* and *for which we have that $\mathcal{N}(n)$ is atomic is called a* state*, and a node m that is either the root node or the direct child node of a state (which leaps to the next point of time) is called a* pre-state*.*

We give a definition of eventuality fulfillment that will be of use later on.

Definition 49 *Let $\mathcal{T}$ be a tableau, and π a path in $\mathcal{T}$ defined from nodes n_1, n_2, ..., n_j. Any eventuality $\Diamond\epsilon_2$ or $\epsilon_1 \mathcal{U} \epsilon_2 \in \mathcal{T}(n_i)$, with $1 \leq i \leq j$, is* fulfilled *in π if there exists k, with $i \leq k \leq j$, such that $\epsilon_2 \in \mathcal{T}(n_k)$.*

We now present Clotho, an algorithm that constructs a semantic tableau $\mathcal{T}$ for a given formula ϕ, as shown in Algorithm 30. At any given point of time, we construct all the possible atomic QCNs comprising base relations that extend from the given point of time to a future point of time. This is achieved by repeatedly applying the decomposition rules to a node comprising unmarked non-elementary formulas (lines 4 to 9), and sequentially populating a node comprising only elementary and marked formulas with the universal relation B (lines 10 to 19) so that it may lead to a state. The universal relation B is only introduced on a pair of variables, if there does not exist any base relation on that same pair. The universal relation B, as well as any other relation $r \in 2^\mathsf{B}$, is essentially the disjunction of base relations, as noted in Section 2. In particular, B is the disjunction of all the base relations of a given qualitative constraint language. As such, by decomposing B into base relations using the disjunctive tableau rule, this approach allows us to obtain one or more nodes harboring atomic QCNs for a given point of time (viz., states), that represent a set of atomic spatial constraints in a fixed-width window of time. Once we have obtained our atomic QCNs for a given point of time, and assuming that the states that harbor them contain $\bigcirc$-formulas, we can leap to the next point of time and create pre-states, including all the atomic spatial constraints of the aforementioned QCNs that extend from the new point of time to a future point of time (lines 20 to 24). This can be seen as making a +1 time shift and maintaining all possible knowledge offered by previous states that extends from the new point of time to a future point of time. It is important to note that when we create a child node m of a node n (lines 7, 17, and 21), we only create a new node if there does not

Algorithm 30: Clotho(ϕ)

in : A $\mathcal{L}_1$ formula ϕ.
output : A semantic tableau $\mathcal{T}$ for ϕ.

begin
  create root node $\{\phi\}$ and mark it unprocessed;
  while $\exists$ *unprocessed node* n **do**
    if $\mathcal{T}(n)$ *contains an unmarked non-elementary formula* ψ **then**
      mark node n processed;
      foreach $\gamma \in \Gamma$, *where* Γ *is the result of applying a decomposition rule to* ψ **do**
        create a child node m;
        $\mathcal{T}(m) \leftarrow (\mathcal{T}(n) - \{\psi\}) \cup \gamma \cup \{\psi*\}$;
        mark node m unprocessed;
    else if $\mathcal{T}(n)$ *contains only elementary and marked formulas* **then**
      mark node n processed;
      filling $\leftarrow \emptyset$;
      foreach $u, v \in expandVars(\phi)$ **do**
        if $\nexists\ P(u, v) \in \mathcal{T}(n)$ **then**
          filling $\leftarrow$ filling $\cup \{\mathsf{B}(u, v)\}$;
      if filling $\neq \emptyset$ **then**
        create a child node m;
        $\mathcal{T}(m) \leftarrow \mathcal{T}(n) \cup$ filling;
        mark node m unprocessed;
      else if $\mathcal{T}(n)$ *contains* $\bigcirc$*-formulas* **then**
        create a child node m;
        $\mathcal{T}(m) \leftarrow \{\psi \mid \bigcirc \psi \in \mathcal{T}(n)\}$;
        $\mathcal{T}(m) \leftarrow \mathcal{T}(m) \cup \{P(\bigcirc^{i-1}u, \bigcirc^{j-1}v) \mid P(\bigcirc^i u, \bigcirc^j v) \in \mathcal{T}(n)$ if $i, j \geq 1\}$;
        mark node m unprocessed;

Algorithm 31: Atropos($\mathcal{T}$)

in/out : A semantic tableau $\mathcal{T}$.
output : True or False.

begin
  do
    flag $\leftarrow$ False;
    if *there is a node* n *such that* $\mathcal{N}(n)$ *is an unsatisfiable* QCN **then**
      eliminate node n; flag $\leftarrow$ True;
    if *all the children of a node* n *have been eliminated* **then**
      eliminate node n; flag $\leftarrow$ True;
    if *a node* n *is a* pre-state **and not** Lachesis($\mathcal{T}, n$) **then**
      eliminate node n; flag $\leftarrow$ True;
  while *flag*;
  if $\nexists$ *node* $n \in \mathcal{T}$ **then return** False **else return** True;

already exist a node in the graph labeled by $\mathcal{T}(m)$. Otherwise, we just create an arc from node n to the existing node.

A tableau $\mathcal{T}$ for a $\mathcal{L}_1$ formula ϕ that has resulted after the application of algorithm Clotho is

Algorithm 32: Lachesis($\mathcal{T}, n$)

in : A semantic tableau $\mathcal{T}$, and a node n.
output : True or False.

```
1 begin
2     foreach eventuality ϵ ∈ T(n) do
3         if ϵ is not fulfilled in any path π = ⟨n, ...⟩ then return False;
4     return True;
```

finite; if ϕ is over a set of l variables, then $\mathcal{T}$ has at most $O(|B|^{l^2(|\phi|+1)^3}2^{length(\phi)})$ nodes.

To decide the satisfiability of a $\mathcal{L}_1$ formula ϕ using the tableau that is generated by Clotho, we have to eliminate unsatisfiable nodes inductively, until a fixed point is reached. We present Atropos, an algorithm that achieves this goal, shown in Algorithm 31. Note that function Lachesis (see Algorithm 32) essentially searches for a path from a given pre-state to a node that fulfills an eventuality of the pre-state, as defined in Definition 49.

Definition 50 *Let $\mathcal{T}$ be a tableau for a $\mathcal{L}_1$ formula ϕ that has resulted after the application of algorithms* Clotho *and* Atropos. *If $\mathcal{T}$ has no root node, we call $\mathcal{T}$* closed*, and* open *otherwise.*

Let us consider formula $\phi = \{EQ(x\ ,y)$, $PO(\bigcirc x, \bigcirc y)$, $TPP(x, \bigcirc x)$, $TPP(y, \bigcirc y)$, $TPP(x, \bigcirc y)$, $\Diamond DC(x, y)\}$. (For simplicity we assume that the decomposition rule for $\wedge$ has already been applied and resulted in the current set form for formula ϕ.) The tableau obtained by the application of algorithms Clotho and Atropos for this formula is shown in Figure 6.5. Horizontal dotted lines distinguish between different points in time, thus, our tableau extends over three points of time. The root node is 1, the states are 5 to 12, 14, 15, 17, and 18, and the pre-states are 1, 13, and 16. By decomposing the initial formula using the tableau rules and populating it with universal relations where appropriate, we reach states 5 to 12, each one of which harbors a set of base relations that correspond to an atomic QCN. (Inverse relations are not shown to save space.) These atomic QCNs represent a set of atomic spatial constraints in a fixed-width window of time. After leaping to the next point of time and, consequently, obtaining pre-state 13, we include all the atomic spatial constraints of the aforementioned QCNs that extend from the new point of time to a future point of time. In this particular case, the atomic spatial constraints of interest narrow down to the single atomic constraint $PO(\bigcirc x, \bigcirc y)$, common for all states 5 to 12. Of course, since we are now at the next point of time, the constraint is rewritten to $PO(x, y)$. Again, we apply the rules and reach states 14 and 15, each one of which harbors an atomic QCN. We continue repeating the process until all our child nodes are labeled by sets of formulas already met in nodes of the tableau. In this case, the unique child node of state 18 would be labeled by the set of formulas of node 16, thus, we create an arc from 18 to 16. After having constructed our tableau, we delete unsatisfiable nodes 2, 5 to 11, and 14 using the $\diamond$-consistency operation on QCNs $\mathcal{N}(2)$, $\mathcal{N}(5)$ to $\mathcal{N}(11)$, and $\mathcal{N}(14)$ respectively. Inconsistencies stemming from nodes 2 and 14 are apparent, as there exist different base relations on a same pair of variables, whereas inconsistencies in nodes 5 to 11 stem from the fact that relation $TPP(y, \bigcirc x)$ is inferred by $\diamond$-consistency, which contradicts with the base relation that is defined on variables y and $\bigcirc x$ in states 5 to 11. Formula ϕ is satisfiable, as the tableau is open, and a model can be constructed out of the sequence of states 12,15,17 which contains a loop on 17 as relation $DC(x, y)$ repeats itself. These states harbor satisfiable atomic QCNs that completely agree on their common part due to our construction. In particular, we have the sequence $\mathcal{N}(12)\rightarrow\mathcal{N}(15)\rightarrow\mathcal{N}(17)\circlearrowleft$ that satisfies the prerequisites of Proposition 52, hence, satisfiability is met.

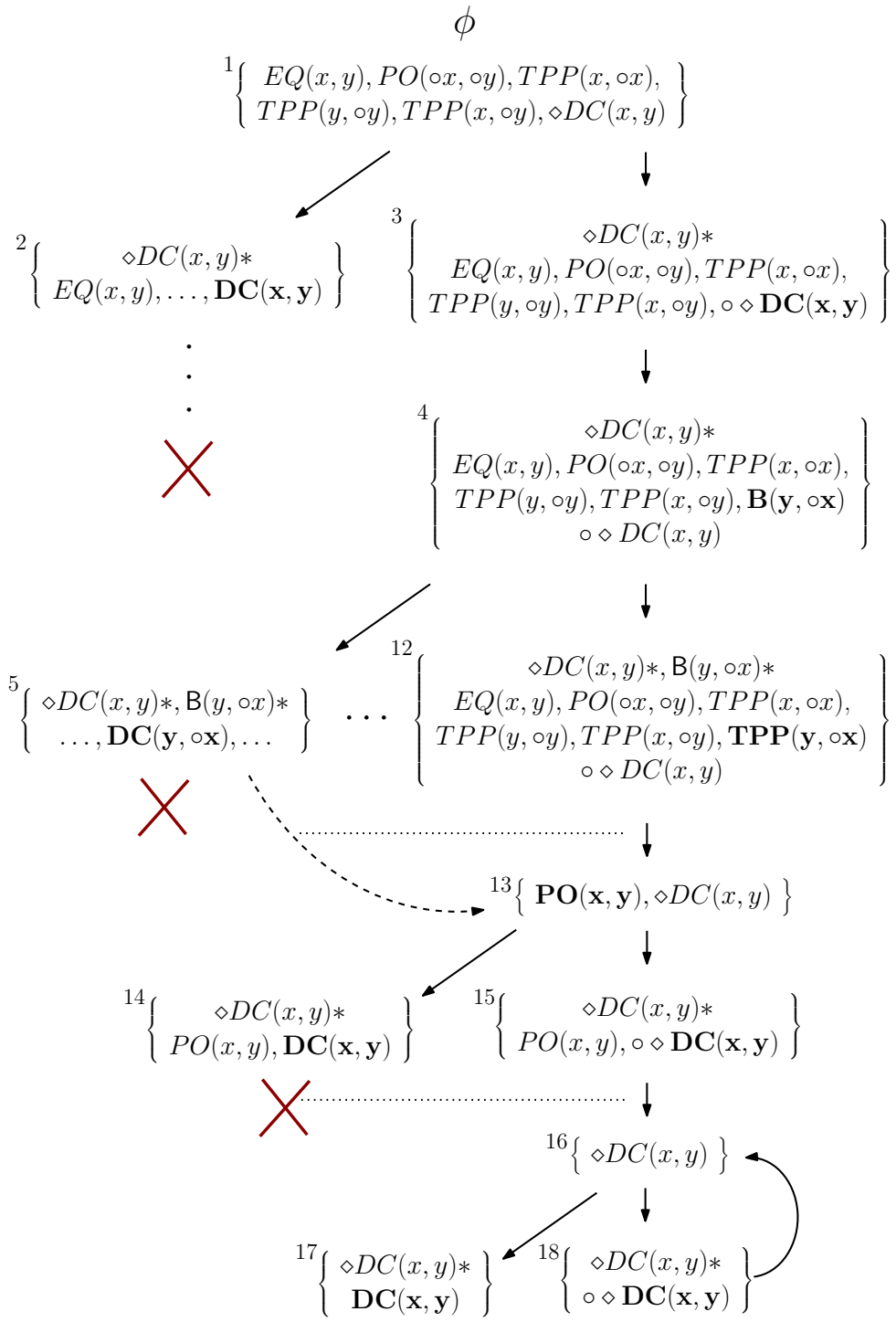


Figure 6.5: A $\mathcal{L}_1$ formula and its simplified tableau

6.4.3 Soundness and Completeness of our Semantic Tableau Method

In this section, we prove that the tableau method as defined by algorithms Clotho and Atropos is sound and complete for checking the satisfiability of a $\mathcal{L}_1$ formula ϕ.

Theorem 36 (soundness) *If ϕ has a closed tableau, then ϕ is unsatisfiable.*

Proof. Let $\mathcal{T}$ be a closed tableau for ϕ, that has resulted after the application of algorithms Clotho and Atropos. We prove by induction that if a node n is eliminated, then $\mathcal{T}(n)$ is an unsatisfiable set of formulas. We distinguish three cases:

(i) a node n is eliminated because $\mathcal{N}(n)$ is an unsatisfiable QCN (lines 4 to 5 in Atropos), thus, $\mathcal{T}(n)$ is an unsatisfiable set of formulas; unsatisfiability of $\mathcal{N}(n)$ can de detected by use of $\diamond$-consistency, which also disallows the conjunction of two or more base relations to be defined on a same pair of variables (base relations are jointly exhaustive and pairwise disjoint as noted in Section 2).

(ii) a node n is eliminated because all of its child nodes are unsatisfiable and have been eliminated (lines 6 to 7 in Atropos). Child nodes can be created in the following three cases:

  (a) the decomposition rule $\psi \rightarrow \Gamma$, where $\psi \in \mathcal{T}(n)$, is applied and a child node is created for each $\gamma \in \Gamma$ (lines 4 to 9 in Clotho); we have that ψ is satisfiable iff $\exists \gamma \in \Gamma$ that is satisfiable.

  (b) implicit knowledge in the parent node n is made explicit in the child node m through the introduction of the universal relation B (lines 10 to 19 in Clotho); by Definition 47 we have that $\mathcal{N}(n) = \mathcal{N}(m)$, thus, $\mathcal{N}(n)$ is satisfiable iff $\mathcal{N}(m)$ is satisfiable, and the same holds for the set of formulas $\mathcal{T}(n)$ and $\mathcal{T}(m)$.

  (c) node n is a state and generates pre-state m with $\mathcal{T}(m) = \{\psi \mid \bigcirc \psi \in \mathcal{T}(n)\} \cup \{P(\bigcirc^{i-1}u, \bigcirc^{j-1}v) \mid P(\bigcirc^{i}u, \bigcirc^{j}v) \in \mathcal{T}(n) \text{ if } i, j \geq 1\}$ (lines 20 to 24 in Clotho); clearly, $\mathcal{T}(n)$ is a satisfiable set of formulas iff $\{\psi \mid \bigcirc \psi \in \mathcal{T}(n)\}$ is a satisfiable set of formulas and iff $\mathcal{N}(m)$ is satisfiable.

(iii) a node n is eliminated if it contains an eventuality that cannot be fulfilled in any path in the tableau (lines 8 to 9 in Atropos); since any model will correspond to a path in the tableau, we have that $\mathcal{T}(n)$ is an unsatisfiable set of formulas.

As we have considered all possible cases, at this point we conclude our proof. ⊣

Let us obtain a proposition that denotes that two successive states in a path of an open tableau harbor QCNs that completely agree on their common part.

Proposition 54 *Let π be a path going through an open tableau $\mathcal{T}$ for a $\mathcal{L}_1$ formula ϕ that has resulted after the application of algorithms* Clotho *and* Atropos*, s_t and s_{t+1} two states of π belonging to points of time t and $t+1$ respectively, and $\{x_0, \ldots, x_l\}$ the set of variables in ϕ. Then we have that $\mathcal{N}(s_t)[v_m^k, v_{m'}^{k'}] = \mathcal{N}(s_{t+1})[v_m^{k-1}, v_{m'}^{k'-1}]$ $\forall m, m' \in \{0, \ldots, l\}$ and $\forall k, k' \in \{1, \ldots, |\phi|\}$.*

Proof. State s_t at point of time t is followed by a pre-state p at point of time $t+1$ in path π, whose set of base relations is $\{P(\bigcirc^{i-1}u, \bigcirc^{j-1}v) \mid P(\bigcirc^{i}u, \bigcirc^{j}v) \in \mathcal{T}(s_t) \text{ if } i, j \geq 1\} \cup \{P(\bigcirc^{i}u, \bigcirc^{j}v) \mid \bigcirc P(\bigcirc^{i}u, \bigcirc^{j}v) \in \mathcal{T}(s_t)\}$ by construction of our tableau (lines 20 to 24 in Clotho). The set of base relations of $\mathcal{T}(p)$ is carried over, possibly enriched, to state s_{t+1} at point of time $t+1$. As such, let us assume that there exists an additional base relation $b(\bigcirc^{i-1}u, \bigcirc^{j-1}v)$ in the set of base relations of s_{t+1}, with $i, j \in \{1, \ldots, |\phi|\}$, such that $b(\bigcirc^{i}u, \bigcirc^{j}v) \notin \mathcal{T}(s_t)$. In this case, $\mathcal{N}(s_{t+1})$ is a QCN with two base relations defined on a same pair of variables. This QCN would have been deleted by the application of Atropos as specified also in the proof of Theorem 36. Thus, state s_{t+1} could not have been in path π, resulting in a contradiction. Therefore, we have that $\mathcal{N}(s_t)[v_m^k, v_{m'}^{k'}] = \mathcal{N}(s_{t+1})[v_m^{k-1}, v_{m'}^{k'-1}]$ $\forall m, m' \in \{0, \ldots, l\}$ and $\forall k, k' \in \{1, \ldots, |\phi|\}$, and, as such, $\mathcal{N}(s_t)$ and $\mathcal{N}(s_{t+1})$ completely agree on their common part. ⊣

Theorem 37 (completeness) *If ϕ has an open tableau, then ϕ is satisfiable.*

Proof. Let $\mathcal{T}$ be an open tableau for ϕ, that has resulted after the application of algorithms Clotho and Atropos. We need to show that there exists a path of nodes π which defines a model for ϕ. We distinguish two cases:

(i) if no eventualities need to be fulfilled, path π can be simply a path starting from the root node and going through the tableau, defining a sequence of states s_0,s_1,...,s_t, with $t \in \mathbb{N}$, and, consequently, yielding a sequence of QCNs as follows.

$$\mathcal{N}(s_0) \rightarrow \mathcal{N}(s_1) \ldots \rightarrow \mathcal{N}(s_t)$$

The sequence of QCNs is such that for all states s_i and s_{i+1}, with $i \in \{0, \ldots, t-1\}$, along with a set of variables $\{x_0, \ldots, x_l\}$ in ϕ, we have that $\mathcal{N}(s_i)[v_m^k, v_{m'}^{k'}] = \mathcal{N}(s_{i+1})[v_m^{k-1}, v_{m'}^{k'-1}]$ $\forall m, m' \in \{0, \ldots, l\}$ and $\forall k, k' \in \{1, \ldots, |\phi|\}$ by Proposition 54. Thus, the sequence of QCNs corresponds to the sequence shown in Figure 6.1 (at page 163), satisfies the prerequisites of Proposition 51 (at page 162), and is therefore satisfiable.

(ii) if eventualities need to be fulfilled, we show how we can construct a path π that fulfills all eventualities as follows. For each pre-state $p \in \mathcal{T}$ containing an eventuality, we must find a path $\pi_p = \langle p, \ldots \rangle$ starting from p, such that all the eventualities contained in p are fulfilled in π_p. We fulfill all the eventualities of p, one by one, as follows. For a selected eventuality $\epsilon \in \mathcal{T}(p)$, it is possible to find a path $\pi_p = \langle p, \ldots, p' \rangle$ in which ϵ is fulfilled and whose last node is a pre-state p', as otherwise the node would have been deleted by the application of Atropos. By construction of our tableau, p' will also contain the rest of the eventualities that need to be fulfilled (they are carried over from p to p'), and it follows that we can extend path π_p to fulfill a second one, and so on, until all the eventualities of p are fulfilled. By linking together all paths π_p $\forall$ pre-states $p \in \mathcal{T}$, we can obtain a path π starting from the initial node and going through the tableau, defining a sequence of states s_0,s_1,...,s_{t-1}, with $t \in \mathbb{N}$, with a final loop between state s_{t-1} and a state $s_{t'}$, with $0 \leq t' \leq t-1$. The loop exists due to the fact that at point of time $t-1$ there exists a node n, whose child node m is such that $\mathcal{T}(m) = \mathcal{T}(o)$, where o is a node at point of time t'. In particular, we can view the sequence of states as a sequence of QCNs as follows.

$$\mathcal{N}(s_0) \rightarrow \mathcal{N}(s_1) \ldots \rightarrow \underbrace{\mathcal{N}(s_{t'}) \ldots \rightarrow \mathcal{N}(s_{t-1})}$$

The sequence of QCNs is such that for all states s_i and s_{i+1}, with $i \in \{0, \ldots, t-2\}$, along with a set of variables $\{x_0, \ldots, x_l\}$ in ϕ, we have that $\mathcal{N}(s_i)[v_m^k, v_{m'}^{k'}] = \mathcal{N}(s_{i+1})[v_m^{k-1}, v_{m'}^{k'-1}]$ $\forall m, m' \in \{0, \ldots, l\}$ and $\forall k, k' \in \{1, \ldots, |\phi|\}$ by Proposition 54. Further, if we were to extend path π, we would obtain a state s_t with $\mathcal{N}(s_t)[v_m^k, v_{m'}^{k'}] = \mathcal{N}(s_{t'})[v_m^k, v_{m'}^{k'}]$ $\forall m, m' \in \{0, \ldots, l\}$ and $\forall k, k' \in \{0, \ldots, |\phi|\}$ (i.e., s_t replicates the same set of spatial constraints with $s_{t'}$, hence, the loop). Thus, the sequence of QCNs corresponds to the sequence shown in Figure 6.2 (at page 164), satisfies the prerequisites of Proposition 52 (at page 163), and is therefore satisfiable.

As we have considered all possible cases, at this point we conclude our proof. ⊣

6.5 Ordering Spatio-Temporal Sequences to meet Transition Constraints

In this section, we focus on a particular spatio-temporal reasoning problem that lies within the area of Graph Traversal [Rosenkrantz *et al.*, 1977], which is one of the oldest areas of inquiry in Graph Theory. Graph Traversal commonly deals with visiting all the nodes in a graph in a particular manner, updating and/or checking their values along the way. We are interested in a problem related to the Hamiltonian path problem for a given graph, which is the graph traversal problem of finding a path in the graph that visits each vertex exactly once. Hamiltonian path related problems naturally extend into use cases where routes need to be ordered or optimised, minimising the traversal of paths and vertices already visited.

In the context of spatio-temporal reasoning, and spatio-temporal sequences in particular in the manner they have been presented in Section 4.3, our problem of interest in a Hamiltonian path related problem where we want to order a sequence of qualitative spatial configurations to meet certain transition constraints. This ordering is constrained by the use of conceptual neighbourhood graphs defined on qualitative spatial constraint languages. For this problem, we consider several well-known qualitative spatial and temporal constraint languages, such as RCC-8 [Randell *et al.*, 1992], Interval Algebra [Allen, 1983], Block Algebra [Balbiani *et al.*, 2002], Rectangle Algebra [Guesgen, 1989][33], and Point Algebra [Vilain *et al.*, 1990; van Beek and Cohen, 1990; van Beek, 1992]. Two closely related contributions that deal with sequences of qualitative spatial or temporal configurations consist of the works of Westphal et al. in [Westphal *et al.*, 2013] and Cui et al. in [Cui *et al.*, 1992], both of which were presented in our state of the art part of the thesis in Section 4.3. In both of these works, qualitative configurations extracted follow a predefined ordering, where all the pairs of consecutive qualitative configurations in the sequence produced meet certain transition constraints with respect to an assumed conceptual neighbourhood graph. In our case, our knowledge base already comprises a set of qualitative configurations, and the problem is that of finding an ordering of those qualitative configurations when positioned in a sequence, such that all the pairs of consecutive qualitative configurations in the ordered sequence meet the aforementioned transition constraints. Thus, we define a novel problem in the context of qualitative spatio-temporal reasoning, whose computational properties we are the first to study.

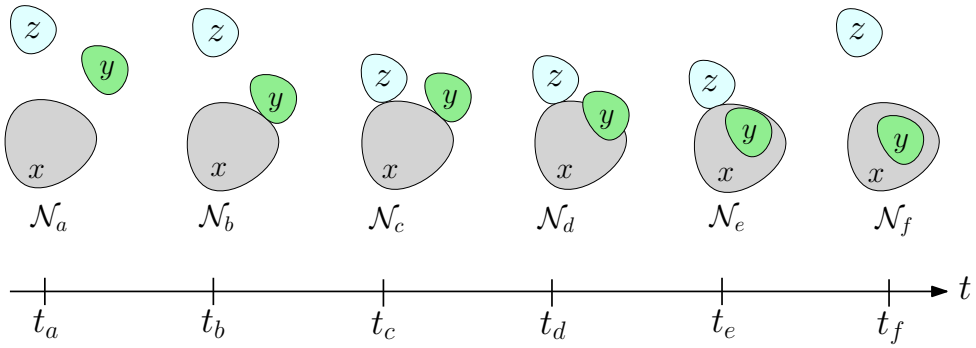
In particular, we will make the following contributions in what follows:

(i) we consider a sequence of qualitative spatial configurations of RCC-8, Interval Algebra, Block Algebra, or Rectangle Algebra, and show that it is NP-complete to order the configurations in a way such that the transition constraints are met with respect to the conceptual neighbourhood graph of the considered language;

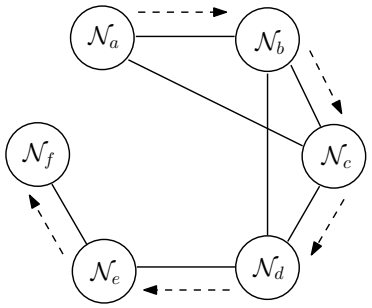
(ii) we study a particular optimization problem that results from the qualitative constraint language of RCC-8 when restricting the size of its regions, and obtain some interesting computational propeties;

(iii) and we introduce a framework where the temporal aspect of a sequence of qualitative spatial configurations of RCC-8, Interval Algebra, Block Algebra, or Rectangle Algebra is replaced with a Point Algebra network, which further restricts the desired ordering of the configurations, but which, nevertheless, allows the problem of finding such an ordering to be in NP.

[33] A qualitative constraint language that is *isomorphic* [Ladkin and Maddux, 1994] to a sublanguage of Interval Algebra (see also [Guesgen, 1989, Section 2]).



(a) Example of a spatio-temporal sequence based on RCC-8



(b) Transition graph of the above spatio-temporal sequence

Figure 6.6: The example spatio-temporal sequence and its corresponding transition graph of Section 4.3

6.5.1 Spatio-Temporal Sequence Ordering Problems

Let us recall the definition of a Hamiltonian path before we proceed with an example that introduces and explains our main problem.

Definition 51 *Given a graph $G = (V, E)$, a* Hamiltonian path *in G is a path between two vertices of G that visits each vertex in V exactly once, where a path is an ordered sequence v_0, v_1, ..., v_k of graph vertices $v_i \in V$ such that for $1 \leq i \leq k$, $\{v_{i-1}, v_i\} \in E$.*

Now, let us revisit the example spatio-temporal sequence and its corresponding transition graph presented in Section 4.3. In particular, the atomic spatio-temporal sequence based on RCC-8 is given in Figure 6.6a. Figure 6.6a depicts the sequence ($\mathcal{N}_a = (V, C_a)$, $\mathcal{N}_b = (V, C_b)$, $\mathcal{N}_c = (V, C_c)$, $\mathcal{N}_d = (V, C_d)$, $\mathcal{N}_e = (V, C_e)$, $\mathcal{N}_f = (V, C_f)$), where $V = \{x, y, z\}$ and $\mathcal{N}_a$, $\mathcal{N}_b$, $\mathcal{N}_c$, $\mathcal{N}_d$, $\mathcal{N}_e$, and $\mathcal{N}_f$ are RCC-8 configurations over V. Specifically, $\mathcal{N}_a$ defines the set of constraints $\{DC(x, y),\ DC(y, z),\ DC(x, z)\}$, $\mathcal{N}_b$ defines the set of constraints $\{EC(x, y),\ DC(y, z),\ DC(x, z)\}$, $\mathcal{N}_c$ defines the set of constraints $\{EC(x, y),\ DC(y, z),\ EC(x, z)\}$, $\mathcal{N}_d$ defines the set of constraints $\{PO(x, y),\ DC(y, z),\ EC(x,\ z)\}$, $\mathcal{N}_e$ defines the set of constraints $\{TPPi(x, y),\ DC(y, z),\ EC(x, z)\}$, and finally $\mathcal{N}_f$ defines the set of constraints $\{NTPPi(x, y),\ DC(y, z),\ DC(x,\ z)\}$. Each spatial QCN in the sequence corresponds to a unique point of time in the timeline t. For example, spatial configuration $\mathcal{N}_c$ corresponds to the point of time t_c in the timeline t. Thus, the ordering of the spatial QCNs in a given sequence yields a spatio-temporal configuration that describes how a spatial configuration evolves over time. The transition graph of the spatio-temporal sequence depicted in Figure 6.6a, defined with respect to the standard

conceptual neighbourhood graph of RCC-8 (Figure 4.2 at page 71), is shown in Figure 6.6b. Indeed, we can have continuous transitions between the spatial QCNs in the pairs $(\mathcal{N}_a,\mathcal{N}_b)$, $(\mathcal{N}_b,\mathcal{N}_c)$, $(\mathcal{N}_c,\mathcal{N}_d)$, $(\mathcal{N}_d,\mathcal{N}_e)$, $(\mathcal{N}_e,\mathcal{N}_f)$ of consecutive QCNs in the sequence ($\mathcal{N}_a$, $\mathcal{N}_b$, $\mathcal{N}_c$, $\mathcal{N}_d$, $\mathcal{N}_e$, $\mathcal{N}_f$), but also continuous transitions between spatial configurations $\mathcal{N}_a$ and $\mathcal{N}_c$ (i.e., the pair $(\mathcal{N}_a,\mathcal{N}_c)$), and $\mathcal{N}_b$ and $\mathcal{N}_d$ (i.e., the pair $(\mathcal{N}_b,\mathcal{N}_d)$). It is easy to see that the pairs of consecutive QCNs in the aforementioned sequence correspond to a Hamiltonian path illustrated with dashed arrows in Figure 6.6b. Another possible sequence that would respect the transition constraints and yield a Hamiltonian path could be, for example, the sequence ($\mathcal{N}_c$, $\mathcal{N}_a$, $\mathcal{N}_b$, $\mathcal{N}_d$, $\mathcal{N}_e$, $\mathcal{N}_f$).

Based on the previous observation, we will now formally introduce the main problem that we are interested in studying here and sketch its relation with the problem of finding a Hamiltonian path in a given graph, which is known to be NP-complete [Garey and Johnson, 1979; Garey and Johnson, 1979]. In fact, we will make a polynomial-time reduction of the Hamiltonian path problem to our problem. We call our problem the sequence ordering problem (SOP) and define it as follows.

Definition 52 *Given a qualitative constraint language, a conceptual neighbourhood graph* Γ *of that language, and a satisfiable atomic qualitative spatio-temporal sequence (*QSS*)* $\mathcal{S} = (\mathcal{N}_1, \mathcal{N}_2, \ldots, \mathcal{N}_k)$*, the* SOP *for* $\mathcal{S}$ *is the problem of obtaining an ordered sequence of the* QCN*s of* $\mathcal{S}$ *such that the spatial* QCN*s* $\mathcal{N}_i$ *and* $\mathcal{N}_j$ *in every pair of consecutive* QCN*s* $(\mathcal{N}_i, \mathcal{N}_j)$ *in the ordered sequence are conceptual neighbours with respect to* Γ*.*

The relation between the Hamiltonian path problem and the SOP is as follows.

Lemma 5 *Given a qualitative constraint language, a conceptual neighbourhood graph* Γ *of that language, a satisfiable atomic* QSS $\mathcal{S}$*, and the transition graph* M *of* $\mathcal{S}$ *defined with respect to* Γ*, an ordered sequence of the* QCN*s of* $\mathcal{S}$ *is a solution of the* SOP *for* $\mathcal{S}$ *iff it defines a Hamiltonian path in* M*.*

Proof. As stated in Definition 51, given a graph G, a path in G is completely specified by an ordered sequence of vertices of G. Let $\mathcal{S} = (\mathcal{N}_1, \mathcal{N}_2, \ldots, \mathcal{N}_k)$ be a satisfiable atomic QSS, Γ some conceptual neighbourhood graph of the qualitative constraint language at hand, and $M = (\{\mathcal{N}_1, \mathcal{N}_2, \ldots, \mathcal{N}_k\}, E)$ the transition graph of $\mathcal{S}$ defined with respect to Γ. Let the ordered sequence of vertices $\mathcal{N}_1, \mathcal{N}_2, \ldots, \mathcal{N}_k$ of M be a Hamiltonian path in M. We will prove by contradiction that this path is also a solution of the SOP for $\mathcal{S}$. Hence, let us assume that that path is not a solution of the SOP for $\mathcal{S}$. Then, there exists a pair of consecutive QCNs $(\mathcal{N}_{i-1}, \mathcal{N}_i)$ in the path, with $1 < i \leq k$, for which the QCNs $\mathcal{N}_{i-1}$ and $\mathcal{N}_i$ are not conceptual neighbours with respect to Γ. By definition of a transition graph, this means that $\{\mathcal{N}_{i-1},\mathcal{N}_i\} \notin E$, which is a contradiction as $\{\mathcal{N}_{i-1},\mathcal{N}_i\}$ defines an edge in the considered Hamiltonian path in M. In a simpler manner, we can prove that if the ordered sequence of vertices $\mathcal{N}_1, \mathcal{N}_2, \ldots, \mathcal{N}_k$ is a solution of the SOP for $\mathcal{S}$, then it also defines a Hamiltonian path in M. We have that for all $1 < i \leq k$ the pairs of consecutive QCNs $(\mathcal{N}_{i-1}, \mathcal{N}_i)$ are conceptual neighbours with respect to Γ and, thus, form an edge in the transition graph M, i.e., $\{\mathcal{N}_{i-1},\mathcal{N}_i\} \in E$. In addition, the ordered sequence of vertices $\mathcal{N}_1, \mathcal{N}_2, \ldots, \mathcal{N}_k$ visits each vertex of M exactly once. As such, the ordered sequence defines a Hamiltonian path in M. ⊣

Indeed, as Lemma 5 suggests, a Hamiltonian path in the transition graph of a given qualitative spatio-temporal sequence, will provide us with an ordered sequence of its QCNs such that the QCNs in every pair of consecutive QCNs in the ordered sequence are conceptual neighbours with respect to some conceptual neighbourhood graph, and vice versa, as explained earlier in light of our example concerning Figure 6.6.

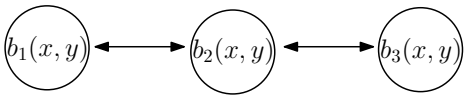


Figure 6.7: A conceptual neighbourhood graph

At this point, we should also highlight the relation between the sequence ordering problem (SOP) presented here and the sequence solving problem (SSP) presented in Section 4.3. With respect to the SSP, we are provided with a qualitative spatio-temporal sequence of QCNs defined on a qualitative constraint language and a conceptual neighbourhood graph of that language, and we must solve the QCNs and extract scenarios of them in a way such that the scenarios in every pair of consecutive scenarios in the obtained sequence of scenarios are conceptual neighbours with respect to the conceptual neighbourhood graph. In a sense, this translates to solving the spatio-temporal sequence in a manner such that the sequence itself *in its default ordering* yields a path in the corresponding transition graph (which will be by default a Hamiltonian path). With respect to the SOP, we are provided with a satisfiable atomic qualitative spatio-temporal sequence of QCNs defined on a qualitative constraint language and a conceptual neighbourhood graph of that language, and we must order the QCNs in a way such that the QCNs in every pair of consecutive scenarios in the obtained ordered sequence of QCNs are conceptual neighbours with respect to the conceptual neighbourhood graph. In a sense, this translates to ordering the spatio-temporal sequence in a manner such that the *ordered* sequence yields a path in the corresponding transition graph (which again will be by default a Hamiltonian path). Hence, the SOP is an incremental variation of the SSP, where we have some transition graph (produced by some scenario of a spatio-temporal sequence) and we want to check if we can obtain a Hamiltonian path in that graph. As such, even though the SSP might be unsolvable for a particular instance because any possible scenario of a sequence of QCNs does not produce a Hamiltonian path in the corresponding transition graph with respect to the default ordering of the sequence, the SOP might still be solvable as we are allowed to change the position of some or all of the QCNs in a given sequence and, therefore, obtain a path in the corresponding transition graph; in other words, we try to find a Hamiltonian path if it is not provided by default due to the initial ordering of a given spatio-temporal sequence.

We provide a definition on graph isomorphism that will be of use in what follows.

Definition 53 *A graph $G_1 = (V_1, E_1)$ is isomorphic to a graph $G_2 = (V_2, E_2)$ iff there is a bijection $f : V_1 \to V_2$ such that $\{u, v\} \in E_1$ iff $\{f(u), f(v)\} \in E_2$.*

It might be tempting at this point to suggest that the SOP for a given QSS $\mathcal{S}$ is NP-complete, as is the case with the Hamiltonian path problem. However, we first need to show that any arbitrary graph G can be translated to an isomorphic to G transition graph M in polynomial time. This is a necessary requirement in our line of reasoning for proving NP-completeness for the SOP, as it could be the case that for a given qualitative constraint language and its conceptual neighbourhood graph, the family of transition graphs that can be constructed allow for obtaining a Hamiltonian path in polynomial time. A trivial case, for example, would be knowing for a fact that any transition graph is a complete graph. Further, to be able to prove NP-completeness for the SOP, we require that a qualitative constraint language has the P_3 property, defined as follows.

Property 2 (Property $\mathbf{P_3}$) *A qualitative constraint language will be said to have the P_3 property if it satisfies the following conditions:*

- *its set of base relations* B *consists of at least three base relations* b_1, b_2, *and* b_3;
- *the conceptual neighbourhood graph defined by base relations* b_1, b_2, *and* b_3 *is the graph* $\Gamma = (\{b_1(u,v), b_2(u,v), b_3(u,v)\}, \{(b_1(u,v), b_2(u,v)), (b_2(u,v), b_3(u,v))\})$, *with* u *and* v *being two entities, as shown in Figure 6.7 (omitting loops);*
- *base relation* b_1 *belongs to all the possible weak compositions among base relations* b_1, b_2, *and* b_3, *viz.,* $b_1 \in b_i \diamond b_j \ \forall i, j \in \{1,2,3\}$;
- *It is a relation algebra and every* $\diamond$*-consistent atomic* QCN *is satisfiable.*

By considering the base relations DC (disconnected), EC (externally connected), and PO (partially overlaps) for RCC-8, the base relations $<$ (before), m (meets), and o (overlaps) for Interval Algebra, and the base relations $<$ (left of), $\leq$ (attached to), and $\Leftarrow$ (overlapping) for Rectangle Algebra, and due to Propositions 1 (at page 21) and 3 (at page 34), we can obtain the following proposition:

Proposition 55 *The qualitative constraint languages* RCC*-8, Interval Algebra, Block Algebra, and Rectangle Algebra have the* P_3 *property.*

Let us go back to being able to construct a transition graph out of any given arbitrary graph in polynomial time. We prove the following proposition:

Proposition 56 *Given a graph* G, *and a qualitative constraint language* $\mathcal{L}$ *that has the* P_3 *property, we have that we can construct a satisfiable atomic* QSS $\mathcal{S}$ *of* $\mathcal{L}$ *that yields an isomorphic to* G *transition graph* M *in polynomial time.*

Proof. Given an arbitrary graph $G = (V, E)$, and a qualitative constraint language $\mathcal{L}$ that has the P_3 property, we can construct a set of satisfiable atomic QCNs of $\mathcal{L}$ that yield a transition graph which is isomorphic to G using algoritm Arachni, depicted in Algorithm 33. We prove the correctness of Arachni as follows.

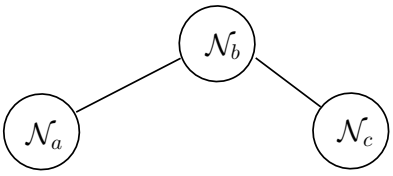
Step 1. If the order of graph G is k, i.e., if $k = |V|$, we create a set $\{\mathcal{N}_1, \mathcal{N}_2, \ldots, \mathcal{N}_k\}$ of k QCNs of $\mathcal{L}$. In fact, we have a bijection between sets V and $\{\mathcal{N}_1, \mathcal{N}_2, \ldots, \mathcal{N}_k\}$, as we consider to have a one-to-one correspondance between an element of V and a QCN in the set of k QCNs of $\mathcal{L}$. This bijection is defined by a dictionary map that given a node $v \in V$ returns the index i of a QCN $\mathcal{N}_i$ in the set of k QCNs of $\mathcal{L}$, i.e., $i = \mathsf{map}[u]$, with $i \in \{1, 2, \ldots, k\}$. For every $i \in \{1, 2, \ldots, k\}$, we have that every QCN $\mathcal{N}_i$ shares the set of variables $\{v_1, v_2, \ldots, v_{k+1}\}$, viz., all k QCNs of $\mathcal{L}$ are defined on the same set of variables $\{v_1, v_2, \ldots, v_{k+1}\}$. We assume first that G is an edgeless graph, therefore, the k QCNs of $\mathcal{L}$ are initially constructed in a manner such that there exists no pair of QCNs where the QCNs in the pair are conceptual neighbours of one another. This is achieved by initializing relation $\mathcal{N}_i[v_i, v_{i+1}]$ for every QCN $\mathcal{N}_i$ with the singleton relation $\{b_3\}$ while initializing all other relations $\mathcal{N}_i[v_j, v_o]$ with $i \neq j$ and $j < o$ with the singleton relation $\{b_1\}$. Then, for any pair of QCNs $(\mathcal{N}_i, \mathcal{N}_j)$ from our set of k QCNs of $\mathcal{L}$, we have that $\mathcal{N}_i$ and $\mathcal{N}_j$ are not conceptual neighbours, since the base relations b_3 and b_1 defined by relations $\mathcal{N}_i[v_i, v_{i+1}]$ and $\mathcal{N}_j[v_i, v_{i+1}]$ respectively (and equivalently, the base relations b_1 and b_3 defined by relations $\mathcal{N}_i[v_j, v_{j+1}]$ and $\mathcal{N}_j[v_j, v_{j+1}]$ respectively) are not conceptual neighbours. Up to this point it should be clear that we have constructed a set of atomic QCNs that yield a transition graph which is isomorphic to an edgeless graph of order k. Since every QCN in our set of k QCNs of $\mathcal{L}$ is defined on $k+1$ entities, and assuming that we use a matrix to represent a given QCN, the construction of our QCNs is achieved in $O(k^3)$ time.

Algorithm 33: Arachni$(G, \mathcal{L})$

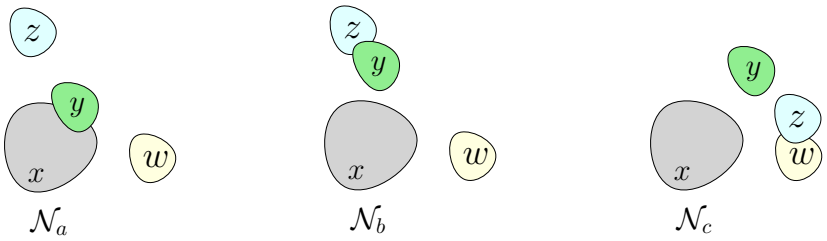
in : A graph $G = (V, E)$, and a qualitative constraint language $\mathcal{L}$ that has the P_3 property.
output : A set of satisfiable atomic QCNs of $\mathcal{L}$ that yield a transition graph which is isomorphic to graph G.

```
1  begin
2      i ← 1;
3      χ ← ∅;
4      map ← dict();
5      while V do
6          map[V.pop()] ← i;
7          V_i ← {v_1, v_2, ..., v_{|V(G)|+1}};
8          foreach v_k, v_l ∈ V_i do
9              if k = l then
10                 C_i(v_k, v_l) ← {Id};
11             else if k < l then
12                 if k = i and l = k + 1 then
13                     C_i(v_k, v_l) ← {b_3}; C_i(v_l, v_k) ← {b_3^-1};
14                 else
15                     C_i(v_k, v_l) ← {b_1}; C_i(v_l, v_k) ← {b_1^-1};
16         N_i ← (V_i, C_i);
17         χ ← χ ∪ {N_i};
18         i ← i + 1;
19     while E do
20         (u, u') ← E.pop();
21         (i, j) ← (map[u], map[u']);
22         N_i[v_j, v_{j+1}] ← {b_2}; N_i[v_{j+1}, v_j] ← {b_2^-1};
23         N_j[v_i, v_{i+1}] ← {b_2}; N_j[v_{i+1}, v_i] ← {b_2^-1};
24     return χ;
```

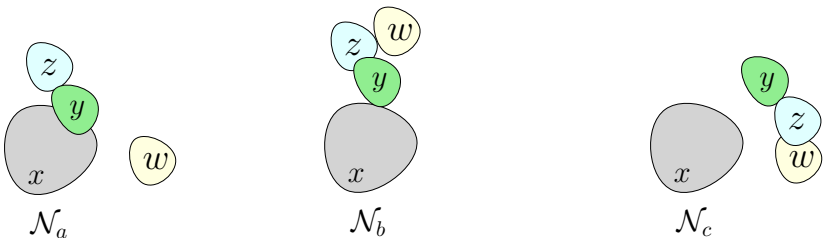
Step 2. Now, we need to iterate the set of edges of graph G and change the QCNs in the corresponding pairs of QCNs into being conceptual neighbours of one another. Using dictionary map, we obtain a pair of QCNs $(\mathcal{N}_i, \mathcal{N}_j)$ for every edge $(u, u') \in E$, where $i = \mathsf{map}[u]$ and $j = \mathsf{map}[u']$. As noted earlier, $\mathcal{N}_i$ and $\mathcal{N}_j$ are not conceptual neighbours, since the base relations b_3 and b_1 defined by relations $\mathcal{N}_i[v_i, v_{i+1}]$ and $\mathcal{N}_j[v_i, v_{i+1}]$ respectively (and equivalently, the base relations b_1 and b_3 defined by relations $\mathcal{N}_i[v_j, v_{j+1}]$ and $\mathcal{N}_j[v_j, v_{j+1}]$ respectively) are not conceptual neighbours. Therefore, we need to change the aforementioned base relations b_1 into being base relation b_2, so that we can achieve conceptual proximity with base relation b_3. In particular, we set relations $\mathcal{N}_j[v_i, v_{i+1}]$ and $\mathcal{N}_i[v_j, v_{j+1}]$ to $\{b_2\}$ from $\{b_1\}$. Note that QCNs $\mathcal{N}_i$ and $\mathcal{N}_j$ become conceptual neighbours only of one another, as any other QCN $\mathcal{N}_o$ with $i \neq o \neq j$ is not a conceptual neighbour of either $\mathcal{N}_i$ or $\mathcal{N}_j$, since relation $\mathcal{N}_o[v_o, v_{o+1}]$ is defined by b_3, and relations $\mathcal{N}_i[v_o, v_{o+1}]$ and $\mathcal{N}_j[v_o, v_{o+1}]$ are still defined by b_1 (and equivalently, relations $\mathcal{N}_o[v_i, v_{i+1}]$ and $\mathcal{N}_o[v_j, v_{j+1}]$ are defined by b_1, and relations $\mathcal{N}_i[v_i, v_{i+1}]$ and $\mathcal{N}_j[v_j, v_{j+1}]$ are defined by b_3). After iterating the whole set of edges of graph G, we will have that any two nodes u and u' of G are adjacent in G if and only if $\mathcal{N}_{\mathsf{map}[u]}$ and $\mathcal{N}_{\mathsf{map}[u']}$ are adjacent in the transition graph that is defined by the set $\{\mathcal{N}_1, \mathcal{N}_2, \ldots, \mathcal{N}_k\}$ of k QCNs of $\mathcal{L}$. Formally, if M is the transition graph defined by the set $\{\mathcal{N}_1, \mathcal{N}_2, \ldots, \mathcal{N}_k\}$ of k QCNs of $\mathcal{L}$, we have that



(a) An arbitrary graph G



(b) A set of QCNs that yield a transition graph which is isomorphic to the empty graph of the order of G



(c) A set of QCNs that yield a transition graph which is isomorphic to graph G

Figure 6.8: Example of the construction of a transition graph through algorithm Arachni

$\{u, u'\} \in E(G)$ iff $\{\mathcal{N}_{\mathsf{map}[u]}, \mathcal{N}_{\mathsf{map}[u']}\} \in E(M)$. Thus, graph M is isomorphic to graph G. To fix the pairs of QCNs that are conceptual neighbours and consequently introduce the edges in our transition graph, we require $O(k^2)$ time, as there can only be $O(k^2)$ edges in a k order graph (and given that our QCNs are represented by matrices, we can alter their relations in $O(1)$ time). In conclusion, algorithm Arachni requires $O(k^3)$ running time in total to process its input and produce an output.

Step 3. Finally, we also need to show that every QCN in the set of k atomic QCNs of $\mathcal{L}$ that we have constructed is satisfiable. Due to our construction, for every QCN $\mathcal{N}_i = (V_i, C_i)$, with $i \in \{1, 2, \ldots, k\}$, we have that every triple of variables v_o, v_o' and v_o'' in V_i, with $o < o' < o''$, defines a set of relations $\mathcal{N}_i[v_o, v_{o'}]$, $\mathcal{N}_i[v_{o'}, v_{o''}]$, and $\mathcal{N}_i[v_o, v_{o''}]$, such that $\mathcal{N}_i[v_o, v_{o''}]$ is always defined by the base relation b_1, and $\mathcal{N}_i[v_o, v_{o'}]$ and $\mathcal{N}_i[v_{o'}, v_{o''}]$ are defined by either of the three base relations b_1, b_2, and b_3. Due to the fact that $b_1 \in b_i \diamond b_j$ $\forall i, j \in \{1, 2, 3\}$, we have that $\mathcal{N}_i[v_o, v_{o''}] \subseteq \mathcal{N}_i[v_o, v_{o'}] \diamond \mathcal{N}_i[v_{o'}, v_{o''}]$. Further, as $\mathcal{L}$ is a relation algebra and therefore satisfies the axioms of $\diamond$-associativity, $^{-1}$-involution, $^{-1}$-involutive distributivity, and Peircean law (sometimes called cycle law [Dylla *et al.*, 2013]), we can deduce that every path of length 2 in $\mathcal{N}_i$ is closed under the weak composition operation defined by operator $\diamond$, thus, $\mathcal{N}_i$ is $\diamond$-consistent. As $\diamond$-consistency decides the satisfiability of atomic QCNs of $\mathcal{L}$, we have that $\mathcal{N}_i$ is a satisfiable QCN of $\mathcal{L}$ for every $i \in \{1, 2, \ldots, k\}$. ⊣

We present a simple example of the construction of a transition graph through algorithm Arachni in Figure 6.8. In particular, let us consider the arbitrary graph shown in Figure 6.8a

and RCC-8 as our qualitative constraint language of choice. First, we construct a set of QCNs that yield a transition graph which is isomorphic to the empty graph of the order of G, as shown in Figure 6.8b. No transition are allows up to this point. Then, we alter the QCNs in a way such that they yield edges in the transition graph that correspond to edges in G, as shown in Figure 6.8c. Clearly, $\mathcal{N}_a$ cannot transition to $\mathcal{N}_c$ as relations $PO(x, y)$ and $DC(x, y)$ are not conceptual neighbours, but all other transitions are valid.

We proceed with obtaining a complexity result for the SOP, for the case where a considered satisfiable atomic QSS is defined on a qualitative constraint language that satisfies property P_3.

Theorem 38 *The* SOP *for any satisfiable atomic* QSS $\mathcal{S}$ *of a qualitative constraint language satisfying property* P_3*, is* NP*-complete.*

Proof. NP-hardness follows from the fact that the Hamiltonian path problem is NP-complete, and we can translate any input of the Hamiltonian path problem, which is an arbitrary graph G, to an isomorphic to G transition graph M of some QSS $\mathcal{S}$ in polynomial time, due to Proposition 56. Further, due to the notion of isomorphism, it is clear that we can have a Hamiltonian path in M iff we can have a Hamiltonian path in G. By Lemma 5, we have that obtaining a Hamiltonian path in M is equivalent to solving the SOP for $\mathcal{S}$, thus, we ultimately have obtained a polynomial-time reduction from the Hamiltonian path problem to the SOP. We can also explicitly define membership in NP due to the fact that provided with a candidate ordered satisfiable atomic QSS $\mathcal{S}$, we can check if the QCNs in every pair of consecutive QCNs in $\mathcal{S}$ are conceptual neighbours in polynomial time. In particular, if $\mathcal{S}$ comprises k QCNs, we can only have $k-1$ pairs of consecutive QCNs in the sequence, and we can check if the QCNs in a pair are conceptual neighbours in $O(n^2)$ time, given the fact than the QCNs are defined over n entities. Thus, the SOP for any satisfiable atomic QSS $\mathcal{S}$ of a qualitative constraint language satisfying property P_3, is NP-complete. ⊣

Due to Theorem 38 and Proposition 55, we can immediately obtain the following result:

Corollary 38 *The* SOP *for any satisfiable atomic* QSS $\mathcal{S}$ *of* RCC*-8, Interval Algebra, Block Algebra, or Rectangle Algebra is* NP*-complete.*

We can obtain a variation of the SOP for a satisfiable atomic QSS $\mathcal{S}$, where we allow one to consider a number of up to m QCNs in addition to the number of QCNs of $\mathcal{S}$ and solve the SOP for the new augmented QSS $\mathcal{S}'$. This is particularly useful if given a QSS $\mathcal{S}$ we are unable to solve the SOP for $\mathcal{S}$, because $\mathcal{S}$, for example, yields a disconnected transition graph and, thus, does not allow obtaining a Hamiltonian path in its transition graph. We provide a very simple, but, nevertheless, sufficient example to better explain this problem.

Let RCC-8 be our qualitative constraint language of choice with its usual conceptual neighbourhood graph as depicted in Figure 4.2, and $(\mathcal{N}_a, \mathcal{N}_b,)$ a QSS $\mathcal{S}$ of RCC-8, where $\mathcal{N}_a$ defines the set of constraints $\{DC(x, y)\}$ and $\mathcal{N}_b$ defines the set of constraints $\{PO(x, y)\}$. Clearly, the transition graph of $\mathcal{S}$ is disconnected as $\mathcal{N}_a$ and $\mathcal{N}_b$ are not conceptual neighbours and, thus, there can be no transition from $\mathcal{N}_a$ to $\mathcal{N}_b$, and vice versa. In particular, the transition graph of $\mathcal{S}$ is the graph $M = (\{\mathcal{N}_a, \mathcal{N}_b\}, \emptyset)$. As such, the SOP for $\mathcal{S}$ is unsolvable, since there can be no Hamiltonian path in M. However, we can augment $\mathcal{S}$ with the QCN $\mathcal{N}_c$ that defines the set of constraints $\{EC(x, y)\}$, and obtain the QSS $\mathcal{S}' = (\mathcal{N}_a, \mathcal{N}_b, \mathcal{N}_c)$. Then, the transition graph of $\mathcal{S}'$ will be the graph $M' = (\{\mathcal{N}_a, \mathcal{N}_b, \mathcal{N}_c\}, \{\{\mathcal{N}_a, \mathcal{N}_b\}, \{\mathcal{N}_b, \mathcal{N}_c\}\})$. The Hamiltonian path $(\mathcal{N}_a, \mathcal{N}_c, \mathcal{N}_b)$ in M' is exactly a solution of the SOP for $\mathcal{S}'$, where we considered one extra QCN with respect to the number of QCNs of $\mathcal{S}$.

We call this new problem the *relaxed* sequence ordering problem (rSOP) and define it as follows.

Definition 54 *Given an integer m, a qualitative constraint language, a conceptual neighbourhood graph Γ of that language, and a satisfiable atomic QSS $\mathcal{S} = (\mathcal{N}_1, \mathcal{N}_2, \ldots, \mathcal{N}_k)$ defined on a set of variables V, the rSOP for $\mathcal{S}$ is the SOP for QSS $\mathcal{S}'$, where $\mathcal{S}'$ is the sequence $\mathcal{S}$ augmented with a set $\{\mathcal{N}'_1, \mathcal{N}'_2, \ldots, \mathcal{N}'_n\}$ of n QCNs defined on V, with $n \leq m$.*

We proceed with obtaining a complexity result for the rSOP, for the case where a considered satisfiable atomic QSS is defined on a qualitative constraint language that satisfies property P_3.

Theorem 39 *The rSOP for any satisfiable atomic QSS $\mathcal{S}$ of a qualitative constraint language satisfying property P_3 and some integer m, is NP-complete.*

Proof. NP-hardness follows from the fact that the SOP, which is NP-complete due to Theorem 38, can be reduced to the rSOP in polynomial time, by just considering an integer value of $m = 0$ for the rSOP. With the aforementioned requirement for integer m, it is clear that any input for the SOP serves as an input for the rSOP, and a solution of the rSOP is also a solution of the SOP and vice versa. Membership in NP follows from the fact that provided with a candidate ordered satisfiable atomic QSS $\mathcal{S}'$ that corresponds to an input satisfiable atomic QSS $\mathcal{S}$ augmented with $\leq m$ QCNs, we can check if $\mathcal{S}'$ is a solution of the SOP for $\mathcal{S}'$ in polynomial time, as the SOP is in NP. Also, we can check if $\mathcal{S}'$ contains $\leq m$ more QCNs than $\mathcal{S}$ in linear time in the number of QCNs of $\mathcal{S}'$. Thus, the rSOP for any satisfiable atomic QSS $\mathcal{S}$ of a qualitative constraint language satisfying property P_3 and some integer m, is NP-complete. ⊣

Due to Theorem 39 and Proposition 55, we can immediately obtain the following result:

Corollary 39 *The rSOP for any satisfiable atomic QSS $\mathcal{S}$ of RCC-8, Interval Algebra, Block Algebra, or Rectangle Algebra and some integer m is NP-complete.*

The rSOP, as is the case with the SOP, is a decision problem where we try to decide if an adequate ordered sequence exists, and if so, present that sequence as a solution of some input instance. However, we can also view the rSOP as an *optimization problem* [Krentel, 1988; Creignou *et al.*, 2001] where we try to minimize the integer value of m. We will formally define and study this optimization problem of the rSOP in a later separate section.

Let us introduce yet another problem that will be also useful for a contribution in a later section in this thesis. We can view the transition graph of a satisfiable atomic QSS as a digraph (also called a directed graph), where the edges, i.e., the pairs of QCNs, have a direction associated with them that specifies which QCN in the pair can transition to the other one. We call the corresponding problem the *directed* sequence ordering problem (dSOP) and define it as follows.

Definition 55 *Given a qualitative constraint language, a conceptual neighbourhood graph Γ of that language, a satisfiable atomic QSS $\mathcal{S} = (\mathcal{N}_1, \mathcal{N}_2, \ldots, \mathcal{N}_k)$, and a transition digraph $M_{\mathsf{d}} = (\{\mathcal{N}_1, \mathcal{N}_2, \ldots, \mathcal{N}_k\}, A)$, with $A = \{(\mathcal{N}_i, \mathcal{N}_j)$ and/or $(\mathcal{N}_j, \mathcal{N}_i) \mid \{\mathcal{N}_i, \mathcal{N}_j\} \in E\}$, where $M = (\{\mathcal{N}_1, \mathcal{N}_2, \ldots, \mathcal{N}_k\}, E)$ is the transition graph of $\mathcal{S}$ defined with respect to Γ, the dSOP for $\mathcal{S}$ is the problem of obtaining an ordered sequence of the QCNs of $\mathcal{S}$ such that the spatial QCNs $\mathcal{N}_i$ and $\mathcal{N}_j$ in every pair of consecutive QCNs $(\mathcal{N}_i, \mathcal{N}_j)$ in the ordered sequence are conceptual neighbours with respect to Γ and $(\mathcal{N}_i, \mathcal{N}_j) \in A$.*

We proceed with obtaining a complexity result for the dSOP, for the case where a considered satisfiable atomic QSS is defined on a qualitative constraint language that satisfies property P_3.

Theorem 40 *The* dSOP *for any satisfiable atomic* QSS $\mathcal{S}$ *of a qualitative constraint language satisfying property* P_3*, is* NP*-complete.*

Proof. NP-hardness follows from the fact that the SOP, which is NP-complete due to Theorem 38, can be reduced to the dSOP in polynomial time, by just considering a transition digraph $M_{\mathsf{d}} = (V, A)$ of $\mathcal{S}$, with $A = \{(\mathcal{N}_i, \mathcal{N}_j)$ and $(\mathcal{N}_j, \mathcal{N}_i) \mid \{\mathcal{N}_i, \mathcal{N}_j\} \in E\}$, where $M = (V, E)$ is the transition graph of $\mathcal{S}$ defined with respect to Γ. Namely, for every edge in M we introduce both directions of this edge, i.e., both arcs, in M_{d}. With the aforementioned requirement for the transition digraph M_{d}, it is clear that any input for the SOP serves as an input for the dSOP, and a solution of the dSOP is also a solution of the SOP and vice versa. Membership in NP follows from the fact that provided with a candidate ordered satisfiable atomic QSS $\mathcal{S}$, we need to check if $\mathcal{S}$ is a solution of the SOP for $\mathcal{S}$ and also check if the QCNs in every pair of $k-1$ pairs of consecutive QCNs in $\mathcal{S}$ form an arc that belongs to the transition digraph M_{d}. We can perform the former check in polynomial time as the SOP is in NP. For the latter check, if we assume that we use a matrix to store the transition digraph M_{d}, we can check if a pair of QCNs forms an arc that belongs to the transition digraph M_{d} in $O(1)$ time, thus, we need $O(k-1)$ time in total for all $k-1$ pairs of QCNs. As such, the dSOP for any satisfiable atomic QSS $\mathcal{S}$ of a qualitative constraint language satisfying property P_3, is NP-complete. ⊣

Due to Theorem 40 and Proposition 55, we can immediately obtain the following result:

Corollary 40 *The* dSOP *for any satisfiable atomic* QSS $\mathcal{S}$ *of* RCC*-8, Interval Algebra, Block Algebra, or Rectangle Algebra is* NP*-complete.*

Optimizing the Relaxed Sequence Ordering Problem

We study the optimization problem of the rSOP, which results from viewing the number of additional QCNs to be considered for a given spatio-temporal sequence as the objective function to be minimized. In particular, we will prove that for the case of RCC-8 where we require that regions do not change size (i.e., the corresponing conceptual neighbourhood graph of RCC-8 is the graph as depicted in Figure 4.2 at page 71, but without the dashed edges), the optimization problem of the rSOP corresponds to a special kind of a NP-optimization problem [Kann, 1995]. First, we recall the following definition of a NP-optimization problem:

Definition 56 *A* NP*-optimization problem (*NPO*)* Π *is a quadruple* $(I, \sigma, g, goal)$ *such that:*

- I *is the set of all input instances of* Π *and is recognizable in polynomial time.*
- *Given an input instance* $x \in I$*,* $\sigma(x)$ *denotes the set of solutions of* x*. For every solution* $y \in \sigma(x)$*, there exists a polynomial* p *such that* $|y| \leq p(|x|)$*, where* $|y|$ *and* $|x|$ *denote the sizes of* y *and* x *respectively. Futher, it is decidable in polynomial time if for any input instance* $x \in I$ *and any candidate solution* y *of* x *with* $|y| \leq p(|x|)$*, we have that* $y \in \sigma(x)$*.*
- *Given an input instance* $x \in I$ *and a solution* $y \in \sigma(x)$*,* $g(x, y)$ *denotes the positive integer* measure *of* y*. Function* g *is called the* objective function *and is computable in polynomial time.*
- $goal \in \{\mathsf{min}, \mathsf{max}\}$.

The goal of a NPO problem $(I, \sigma, g, goal)$ with respect to an input instance $x \in I$ is to find an *optimum solution*, i.e., a solution $y \in \sigma(x)$ such that:

$$g(x, y) = goal\{g(x, y') \mid y' \in \sigma(x)\}$$

The observant reader will note that the first three items in Definition 56 imply that the corresponding decision problem of a NPO problem is in NP. Moreover, a NPO problem $(I, \sigma, g, goal)$ will be said to be *polynomially bounded* if there exists a polynomial q such that for any input instance $x \in I$ and for any solution $y \in \sigma(x)$, we have that $g(x, y) \leq q(|x|)$, i.e., the objective function is polynomially bounded in the size of the input instance [Kann, 1995]. Polynomially bounded NPO problems constitute an important class of optimization problems, as they have many desirable computational properties with respect to approximability [Kann, 1995; Krentel, 1988].

We recall the following lemma from [Gerevini and Nebel, 2002], which was proven using a large computer-generated case analysis:

Lemma 6 ([Gerevini and Nebel, 2002]) *Let $\mathcal{N}_a$ and $\mathcal{N}_b$ be two satisfiable atomic* QCN*s of* RCC*-8 defined on a set of variables V with $|V| = n$, and Γ the conceptual neighbourhood graph of* RCC*-8 under the requirement that regions do not change size. Then, we can introduce $\leq 12n^2$ pairwise distinct satisfiable atomic* QCN*s of* RCC*-8 defined on V, such that the* QSS *$\mathcal{S}' = (\mathcal{N}_a, \mathcal{N}_1, \mathcal{N}_2, \ldots, \mathcal{N}_{O(12n^2)}, \mathcal{N}_b)$ is a solution of the* SOP *for $\mathcal{S}'$, and equivalently a solution of the* rSOP *for* QSS *$\mathcal{S} = (\mathcal{N}_a, \mathcal{N}_b)$ with $m \leq 12n^2$.*

We can now prove the following theorem with respect to RCC-8:

Theorem 41 *The optimization problem of the* rSOP *for any satisfiable atomic* QSS *$\mathcal{S}$ of* RCC*-8 under the requirement that regions do not change size and some integer m, is a polynomially bounded* NPO *problem $(I, \sigma, g, goal)$ where:*

- *I is the set of all satisfiable atomic* QSS*s of* RCC*-8 comprising k* QCN*s over a set of n spatial variables, for some integers k and n.*
- *The non-empty set of solutions $\sigma(x)$ for an input instance $x \in I$ is a set of* QSS*s, such that $\forall y \in \sigma(x)$ we have that $|y| \leq (k-1)12n^4 + kn^2$, where $|y|$ denotes the size of y. Also, it is decidable in $O((k-1)12n^4 + kn^2 - n^2)$ time if for any input instance $x \in I$ and any candidate solution y of x with $|y| \leq (k-1)12n^4 + kn^2$, we have that $y \in \sigma(x)$.*
- *The objective function g yields the measure of integer m for the* rSOP *and is computable in $O((k-1)12n^2)$ time.*
- *$goal =$* min*.*

Proof. We will prove that given a satisfiable atomic QSS $\mathcal{S} = (\mathcal{N}_1, \mathcal{N}_2, \ldots, \mathcal{N}_k)$ of RCC-8 under the requirement that regions do not change size, comprising k QCNs over a set of n spatial variables V, the rSOP for $\mathcal{S}$, along with some integer m, meets the claimed bounds of Theorem 41. We consider the worst-case scenario for defining the bounds, which will apply to all other scenarios as well. The worst-case scenario for the rSOP, is when we have that the transition graph M of $\mathcal{S}$ is the edgeless graph of order k. Therefore, we have $k-1$ pairs of consecutive QCNs in any ordered sequence of the k QCNs of $\mathcal{S}$, such that the QCNs in every pair are not conceptual neighbours of one another. Due to Lemma 6, we can introduce $\leq 12n^2$ pairwise

distinct satisfiable atomic QCNs of RCC-8 over V for every pair of $k-1$ pairs of QCNs to obtain a QSS $\mathcal{S}' = (\mathcal{N}_1, \mathcal{N}_1^1, \mathcal{N}_2^1, \ldots, \mathcal{N}_{O(12n^2)}^1, \mathcal{N}_2, \mathcal{N}_1^2, \mathcal{N}_2^2, \ldots, \mathcal{N}_{O(12n^2)}^2, \mathcal{N}_3, \mathcal{N}_1^3, \mathcal{N}_2^3, \ldots, \mathcal{N}_{O(12n^2)}^3, \ldots, \mathcal{N}_{k-1}, \mathcal{N}_1^{k-1}, \mathcal{N}_2^{k-1}, \ldots, \mathcal{N}_{O(12n^2)}^{k-1}, \mathcal{N}_k)$, such that the rSOP for $\mathcal{S}$ with $m = (k-1)12n^2$ is solvable, with $\mathcal{S}'$ being the solution. The size of solution $\mathcal{S}'$ is $\leq (k-1)12n^4 + kn^2$, as we have augmented the input QSS $\mathcal{S}$ of k QCNs with $\leq (k-1)12n^2$ QCNs and we assume that we use a matrix to represent a given QCN which has a $O(n^2)$ size requirement. Also, since $\mathcal{S}'$ comprises at most $(k-1)12n^2 + k$ QCNs, we can only have $(k-1)12n^2 + k - 1$ pairs of consecutive QCNs in the sequence, and we can check if the QCNs in a pair are conceptual neighbours in $O(n^2)$ time, thus, we can verify the feasibility of $\mathcal{S}'$ in $O((k-1)12n^4 + kn^2 - n^2)$ time. Finally, any solution of the rSOP is guaranteed to be obtainable if m is upper bounded by $O((k-1)12n^2)$, as such, we have that the optimization problem of the rSOP for any satisfiable atomic QSS $\mathcal{S}$ of RCC-8 under the requirement that regions do not change size and some integer m, is a polynomially bounded NPO problem. ⊣

6.5.2 Constraining Spatio-Temporal Sequences with Point Algebra

In Section 6.5.1 we studied the problem of ordering a qualitative spatio-temporal sequence (QSS) of QCNs to meet certain transition constraints, i.e., we studied the problem of ordering the sequence in a manner such that the QCNs in every pair of consecutive QCNs in the sequence are conceptual neighbours. A QSS comprises strictly spatial QCNs, but the ordering of the sequence itself constitutes a timeline upon which the spatial QCNs are defined. Therefore, a QSS has an implicit temporal aspect as it describes an evolving spatio-temporal configuration in some timeline. In this section we make this temporal aspect explicit by defining a framework where the timeline is constrained by Point Algebra [Vilain *et al.*, 1990; van Beek and Cohen, 1990; van Beek, 1992] relations. We remind the reader, that PA comprises the set of base relations $\{<, =, >\}$, with $=$ being the identity relation, where the relation symbols display the natural interpretation over time points in $\mathbb{Q}$. This makes the problem even more interesting as we have both constraints propagating from the spatial aspect to the temporal aspect and the other way around, and it also makes it more expressive as we will see in a later example.

We obtain a spatio-temporal framework by defining the concept of a qualitative spatio-temporal constraint network (QSCN) that builds on Point Algebra and allows plugging in any qualitative spatial constraint language, such as RCC-8, Interval Algebra, Block Algebra, or Rectangle Algebra. Intuitively, a QSCN is a QCN of Point Algebra where the set of variables corresponds to a set of spatial QCNs. We formally define a QSCN as follows.

Definition 57 *A* QSCN $\mathcal{N}$ *is a* QCN (W, R) *of Point Algebra, where* W *is a set of variables* $\{\mathcal{N}_1 = (V, C_1), \mathcal{N}_2 = (V, C_2), \ldots, \mathcal{N}_k = (V, C_k)\}$ *of* k *satisfiable atomic* QCN*s of a qualitative spatial constraint language over a set of spatial entities* V*, and* R *the usual constraint mapping in a* QCN *as defined in Definition 4 (at page 26).*

Note that we always regard a QSCN as a complete network. In what follows, given a QSCN $\mathcal{N} = (W, R)$ and $v, v' \in W$, $\mathcal{N}[v, v']$ will denote the relation $R(v, v')$. An atomic QSCN $\mathcal{N}$ is a QSCN whose underlying QCN of Point Algebra is atomic, and a scenario $\mathcal{N}(\sigma)$ of $\mathcal{N}$ is a scenario of its underlying QCN of Point Algebra, where σ is a solution of that QCN.

Definition 58 *Let* $(\mathcal{N}_1, \mathcal{N}_2, \ldots, \mathcal{N}_k)$ *be the* QSS *defined by an atomic* QSCN $\mathcal{N} = (\{\mathcal{N}_1, \mathcal{N}_2, \ldots, \mathcal{N}_k\}, R)$*, denoted by* $\mathcal{S}(\mathcal{N})$*,* Γ *a conceptual neighbourhood graph of the considered qualitative constraint language, and* $M = (\{\mathcal{N}_1, \mathcal{N}_2, \ldots, \mathcal{N}_k\}, E)$ *the transition graph of* $\mathcal{S}(\mathcal{N})$ *defined with*

respect to Γ. *Then,* $\mathcal{S}(\mathcal{N})$ *yields a transition digraph* $(\{\mathcal{N}_1, \mathcal{N}_2, \ldots, \mathcal{N}_k\}, A)$, *denoted by* $M_{\mathsf{d}}(\mathcal{N})$, *where* $\forall \mathcal{N}_i, \mathcal{N}_j \in \{\mathcal{N}_1, \mathcal{N}_2, \ldots, \mathcal{N}_k\}$ *with* $i \leq j$*:*

- $(\mathcal{N}_i, \mathcal{N}_j) \in A$ *and* $(\mathcal{N}_j, \mathcal{N}_i) \in A$, *if* $\{\mathcal{N}_i, \mathcal{N}_j\} \in E$ *and* $\mathcal{N}[\mathcal{N}_i,\mathcal{N}_j] = \{=\}$;
- $(\mathcal{N}_i, \mathcal{N}_j) \in A$, *if* $\{\mathcal{N}_i, \mathcal{N}_j\} \in E$ *and* $\mathcal{N}[\mathcal{N}_i,\mathcal{N}_j] = \{<\}$;
- $(\mathcal{N}_j, \mathcal{N}_i) \in A$, *if* $\{\mathcal{N}_i, \mathcal{N}_j\} \in E$ *and* $\mathcal{N}[\mathcal{N}_i,\mathcal{N}_j] = \{>\}$.

Given a QSCN $\mathcal{N}$ defined on some qualitative constraint language along with a conceptual neighbourhood graph Γ of that language, a solution of $\mathcal{N}$ is a solution of the dSOP for $\mathcal{S}(\mathcal{N}(\sigma))$ with respect to $M_{\mathsf{d}}(\mathcal{N}(\sigma))$, where σ is a solution of the underlying QCN of Point Algebra of $\mathcal{N}$ and $\mathcal{N}(\sigma)$ its corresponding scenario.

Definition 59 *A* QSCN $\mathcal{N}$ *is* satisfiable *if and only if it admits a solution.*

Given a QSCN $\mathcal{N}$, obtaining a solution σ of its underlying QCN of Point Algebra and consequently a scenario $\mathcal{N}(\sigma)$ of that QCN, will provide us with an input for the dSOP. In particular, a scenario of the QCN of Point Algebra (i.e., an atomic satisfiable sub-QCN of the QCN of Point Algebra) constrains the timeline upon which the spatial QCNs of sequence $\mathcal{S}(\mathcal{N}(\sigma))$ are defined, by encoding a particular transition digraph $M_{\mathsf{d}}(\mathcal{N}(\sigma))$ of $\mathcal{S}(\mathcal{N}(\sigma))$, as defined in Definition 58. (Of course, the transitions defined by the obtained transition digraph $M_{\mathsf{d}}(\mathcal{N}(\sigma))$ can be even further restricted upon user preference, by considering an other transition digraph with some of the arcs of $M_{\mathsf{d}}(\mathcal{N})$ removed.) In a sense, a QSCN allows us to describe numerous different transition digraphs, but also transitions that cannot be described by a single transition digraph alone, as we will see in the following simple example.

Let us consider the QSCN $\mathcal{N} = (W, R)$ of RCC-8, where $W = \{\mathcal{N}_a = (\{x, y\}, \{DC(x, y)\}), \mathcal{N}_b = (\{x, y\}, \{EC(x, y)\})\}$ and $R(\mathcal{N}_a, \mathcal{N}_b) = \{>, <\}$. Clearly, $\mathcal{N}$ has two scenarios defined by $R(\mathcal{N}_a, \mathcal{N}_b) = \{>\}$ and $R(\mathcal{N}_a, \mathcal{N}_b) = \{<\}$ respectively. Thus, $\mathcal{N}$ encodes two transition digraphs containing arcs $(\mathcal{N}_a, \mathcal{N}_b)$ and $(\mathcal{N}_b, \mathcal{N}_a)$ respectively, ultimately representing the knowledge that either $\mathcal{N}_a$ will transition to $\mathcal{N}_b$, or $\mathcal{N}_b$ will transition to $\mathcal{N}_a$. This dichotomic behavior cannot be represented by a single transition digraph alone, as any such digraph would either allow both transitions between a pair of QCNs, or a single transition of one QCN to the other one in the pair.

Let us consider all the aforementioned notions around a QSCN in conjuction with a qualitative spatio-temporal sequence of Rectangle Algebra [Guesgen, 1989], as follows.

In Figure 6.9 we can view a QSCN $\mathcal{N} = (W, R)$ of Rectangle Algebra, where $W = \{\mathcal{N}_x, \mathcal{N}_y, \mathcal{N}_z\}$ and R is defined by the set of constraints $R(\mathcal{N}_x, \mathcal{N}_y) = \{>, <\}$, $R(\mathcal{N}_y, \mathcal{N}_z) = \{>, <\}$, and $R(\mathcal{N}_x, \mathcal{N}_z) = ?$, where ? is the common notation in the literature for the universal relation of Point Algebra, viz., $\{<, =, >\}$. Every variable of the underlying QCN of Point Algebra of $\mathcal{N}$ corresponds to a spatial QCN. As noted, for the sake of our example, we can view these spatial configurations as QCNs of Rectangle Algebra. All QCNs of Rectangle Algebra share the same set of spatial variables V as imposed by Definition 57, which in our case comprises the disks[34] of the Moon and the Sun. In fact, our example describes the phenomenon of an eclipse. The QCN $\mathcal{N}_x$ of Rectangle Algebra comprises the set of constraints $\{\Rightarrow (\mathsf{Moon}, \mathsf{Sun})\}$ (the disk of the Moon overlaps the disk of the Sun from right to left). The QCN $\mathcal{N}_y$ of Rectangle Algebra comprises the set of constraints $\{\sqsubset (\mathsf{Moon}, \mathsf{Sun})\}$ (the disk of the Moon contains the disk of the Sun). Finally,

[34] We consider these disks to be enclosed in minimum bounding boxes.

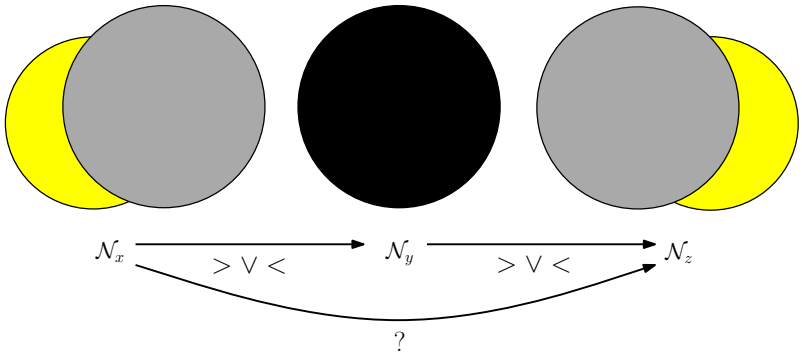


Figure 6.9: Example of a QSCN of Rectangle Algebra

the QCN $\mathcal{N}_z$ of Rectangle Algebra comprises the set of constraints $\{\Leftarrow (\mathsf{Moon}, \mathsf{Sun})\}$ (the disk of the Moon overlaps the disk of the Sun from left to right). Assuming that two observers in different hemispheres can actually see the same eclipse event, we may want to be able to capture the phenomenon both as seen from the perspective of an observer in the Northern hemisphere (the Moon moves from right to left), but also as seen in the Southern hemisphere (the Moon moves from left to right). The reader can verify that both scenarios are encoded by the following possible solutions of $\mathcal{N}$: $(\mathcal{N}_x, \mathcal{N}_y, \mathcal{N}_z)$ and $(\mathcal{N}_z, \mathcal{N}_y, \mathcal{N}_x)$. In particular, solution $(\mathcal{N}_x, \mathcal{N}_y, \mathcal{N}_z)$ corresponds to a scenario $\mathcal{N}(\sigma)$ of $\mathcal{N}$ defined by the set of constraints $R(\mathcal{N}_x, \mathcal{N}_y) = \{<\}$, $R(\mathcal{N}_y, \mathcal{N}_z) = \{<\}$, and $R(\mathcal{N}_x, \mathcal{N}_z) = \{<\}$, where σ corresponds to a solution of the underlying QCN of Point Algebra such as $\sigma(\mathcal{N}_x) = 0$, $\sigma(\mathcal{N}_y) = 1$, and $\sigma(\mathcal{N}_z) = 2$. Thus, solution $(\mathcal{N}_x, \mathcal{N}_y, \mathcal{N}_z)$ of $\mathcal{N}$ is a solution of the dSOP for $\mathcal{S}(\mathcal{N}(\sigma)) = (\mathcal{N}_x, \mathcal{N}_y, \mathcal{N}_z)$ (where in this case we note that the input sequence is already ordered with respect to $M_{\mathsf{d}}(\mathcal{N}(\sigma))$). Regarding the transition digraph $M_{\mathsf{d}}(\mathcal{N}(\sigma))$ of $\mathcal{S}(\mathcal{N}(\sigma))$, it contains the arcs $(\mathcal{N}_x, \mathcal{N}_y)$ and $(\mathcal{N}_y, \mathcal{N}_z)$, as defined in Definition 58. The same line of reasoning holds for solution $(\mathcal{N}_z, \mathcal{N}_y, \mathcal{N}_x)$.

We proceed with obtaining a complexity result for the satisfiability problem of a QSCN $\mathcal{N}$, as defined in Definition 59, for the case where a considered qualitative constraint language satisfies property P_3.

Theorem 42 *The satisfiability problem of a* QSCN $\mathcal{N}$*, where the qualitative constraint language used for the spatial* QCN*s satisfies property* P_3*, is* NP*-complete.*

Proof. NP-hardness follows from the fact that the SOP, which is NP-complete due to Theorem 38 (at page 185), can be reduced to the satisfiability problem for a QSCN in polynomial time. In particular, let $(\mathcal{N}_1, \mathcal{N}_2, \ldots, \mathcal{N}_k)$ be some QSS $\mathcal{S}$, then we can construct a QSCN $\mathcal{N} = (W, R)$ as follows. The set of variables W will be the set $\{\mathcal{N}_1, \mathcal{N}_2, \ldots, \mathcal{N}_k\}$, and R will be the mapping that associates the singleton relation $R(v, v') = \{=\}$ to each pair (v, v') of $W \times W$, i.e., the underlying QCN of Point Algebra of $\mathcal{N}$ will be the atomic QCN of $|W|$ variables that is completely defined by singleton relation $\{=\}$. Assuming that we use a matrix to represent a given QCN, this construction can be made in $O(|W|^2)$ time, where $|W| = k$. The aforementioned underlying QCN of Point Algebra is obviously satisfiable, with a trivial solution being the mapping σ, where $\sigma(v) = 0$ $\forall v \in W$. Therefore, $\mathcal{N}(\sigma)$ is exactly the underlying atomic QCN of Point Algebra, and is also a scenario of $\mathcal{N}$. As such, $\mathcal{N}(\sigma)$ will yield a QSS $\mathcal{S}(\mathcal{N}(\sigma)) = (\mathcal{N}_1, \mathcal{N}_2, \ldots, \mathcal{N}_k)$, and a transition digraph $M_{\mathsf{d}}(\mathcal{N}(\sigma)) = (W, A)$ of $\mathcal{S}(\mathcal{N}(\sigma))$, with $A = \{(\mathcal{N}_i, \mathcal{N}_j)$ and $(\mathcal{N}_j, \mathcal{N}_i) \mid \{\mathcal{N}_i, \mathcal{N}_j\} \in E\}$, where $M = (W, E)$ is the transition graph of $\mathcal{S}$, as defined in Definition 58. Namely, for every edge in M we introduce both directions of this edge, i.e., both arcs, in $M_{\mathsf{d}}(\mathcal{N}(\sigma))$.

Since $\mathcal{S} = \mathcal{S}(\mathcal{N}(\sigma))$, and for every edge $\{u, v\} \in E$ of M we have both arcs $(u, v), (v, u) \in A$ of $M_{\mathsf{d}}(\mathcal{N}(\sigma))$, it is clear that any input for the SOP serves as an input for the dSOP, and a solution of the dSOP is also a solution of the SOP and vice versa. Finally, as a solution of a QSCN $\mathcal{N}$ is a solution of the dSOP for $\mathcal{S}(\mathcal{N}(\sigma))$ with respect to $M_{\mathsf{d}}(\mathcal{N}(\sigma))$, where σ is a solution of the underlying QCN of Point Algebra of $\mathcal{N}$ and $\mathcal{N}(\sigma)$ its corresponding scenario, we ultimately have obtained a polynomial-time reduction from the SOP to the satisfiability problem of a QSCN. Membership in NP follows from the fact that provided with a candidate ordered satisfiable atomic QSS $\mathcal{S}$, we need to check if $\mathcal{S}$ is a solution of the dSOP for $\mathcal{S}$ and also check if the QCN of Point Algebra that results by removing the forbidden relation $>$ from all relations $R(\mathcal{N}_i, \mathcal{N}_j)$ of the underlying QCN of Point Algebra of $\mathcal{N}$, where $\mathcal{N}_i$ and $\mathcal{N}_j$ constitute a pair of consecutive QCNs $(\mathcal{N}_i, \mathcal{N}_j)$ in $\mathcal{S}$, is satisfiable. (If $R(\mathcal{N}_i, \mathcal{N}_j) = \{>\}$ in a scenario $\mathcal{N}(\sigma)$ of $\mathcal{N}$, we will obtain the arc $(\mathcal{N}_j, \mathcal{N}_i)$ in the corresponding transition digraph $M_{\mathsf{d}}(\mathcal{N}(\sigma))$, which invalidates $\mathcal{S}$ as a solution.) We can perform both checks in polynomial time as the dSOP is in NP due to Theorem 40, and $\diamond$-consistency decides the satisfiability of any QCN of Point Algebra (i.e., a scenario $\mathcal{N}(\sigma)$ of $\mathcal{N}$ along with solution σ can be extracted in polynomial time). Thus, the satisfiability problem of a QSCN $\mathcal{N}$, where the qualitative constraint language used for the spatial QCNs satisfies property P_3, is NP-complete. ⊣

Due to Theorem 42 and Proposition 55 (at page 182), we can immediately obtain the following result:

Corollary 41 *The satisfiability problem for a* QSCN $\mathcal{N}$ *where the qualitative constraint language used for the spatial* QCN*s is* RCC*-8, Interval Algebra, Block Algebra, or Rectangle Algebra is* NP*-complete.*

6.6 Conclusion and Future Work

In this chapter, we presented our contributions with respect to formalisms that combine spatial and temporal reasoning in an interrelated manner, and demonstrated how they advance and enrich the field of qualitative spatio-temporal reasoning.

In particular, we studied the qualitative spatio-temporal logic that results by combining the propositional temporal logic (PTL) with a qualitative spatial constraint language, namely, the $\mathcal{L}_{\mathbf{1}}$ logic that was presented in Section 4.2, and investigated the implication of the constraint properties of *compactness* and *patchwork* in qualitative spatio-temporal reasoning (cf. Section 3.6). We used these properties to strengthen results regarding the complexity of the satisfiability problem in $\mathcal{L}_{\mathbf{1}}$, by replacing the stricter global consistency property used in literature and, consequently, generalizing to more qualitative spatial constraint languages. We also used these properties to prove the correctness of a first semantic tableau method that given a $\mathcal{L}_{\mathbf{1}}$ formula ϕ systematically searches for a model for ϕ. This tableau method builds on Wolper's tableau method for PTL, while its basic principles and ideas can be applied to other tableau methods for PTL as well. Further, we identified fragments of the $\mathcal{L}_{\mathbf{1}}$ logic that capture significant aspects of spatio-temporal change. In particular, we addressed the issue of periodical, and smoothness and continuity constraints between spatial configurations, and obtained results on their computational properties. Regarding periodicity, we used the properties of compactness and patchwork to strengthen related results that exist in the literature and were presented in Section 4.5, by re-establishing conditions that allow for tractability and, again, generalizing to a larger class of qualitative spatial constraint languages. Moreover, and with respect to the discussion in Sections 4.3 and 4.4, we investigated the task of ordering a temporal sequence of qualitative spatial configurations

to meet certain transition constraints; in particular, this ordering is constrained by the use of conceptual neighbourhood graphs defined on qualitative spatial constraint languages. Specifically, we showed that the problem of ordering a sequence of qualitative spatial configurations to meet such transition constraints is NP-complete for the well-known languages of RCC-8, Interval Algebra, and Block Algebra. Based on this result, we also proposed a framework where the temporal aspect of a sequence of qualitative spatial configurations is constrained by a Point Algebra network, and again showed that the enhanced problem is in NP when considering the aforementioned languages.

Regarding future work, we would like to consider domain interpretations that involve determined entities (constants) for qualitative constraint languages and study the implication of these interpretations in the context of qualitative spatio-temporal reasoning. In particular, in the entirety of this thesis, we have considered qualitative constraint languages where the domain is abstract and some structure is considered that allows to model any (syntactically) consistent QCN (that structure being referred to as a *canonical model* in the literature). It would therefore be interesting to explore how introducing constants in the domain of a qualitative constraint language could affect its satisfiability problem and, consequently, how this "extended" qualitative constraint language would behave when combined with a temporal logic or when considered to describe the spatial QCNs in a qualitative spatio-temporal sequence. With respect to the first part, i.e., the satisfiability problem in qualitative constraint languages extended with constants, there has already been some work with regard to the qualitative constraint languages of Point Algebra, Interval Algebra, and RCC-8 [Li *et al.*, 2013; Liu *et al.*, 2011]. In particular, it has been shown in [Li *et al.*, 2013; Liu *et al.*, 2011], that checking the satisfiability of a QCN of Point Algebra or Interval Algebra extended with constants, can be done in the same way as checking the satisfiability of any typical QCN of Point Algebra or Interval Algebra through a translation of the constraints that involve constants into qualitative relations. However, this does not hold in the case of RCC-8, as checking the satisfiability of even an atomic QCN of RCC-8 extended with polygonal landmarks becomes NP-complete. What is more, introducing constants in a qualitative constraint language breaks the patchwork and compactness properties that this language can exhibit, as these properties are established upon the use of some canonical model that considers an abstract domain. With respect to the second part, i.e., the behaviour of a qualitative constraint language extended with constants when combined with a temporal logic or when considered to describe the spatial QCNs in a qualitative spatio-temporal sequence, there has not been any published work so far to the best of our knowledge. Therefore, we would like to delve into this problem. Further, we would like to use our results with respect to the tableau method that we presented in Section 6.4 to implement a fast satisfiability checking tool for the $\mathcal{L}_1$ logic. To this end, we think it would be possible to extend Leviathan [Bertello *et al.*, 2016] with spatio-temporal reasoning capabilities.

Recently, we have defined a resolution method for the modal logic S5 [Salhi and Sioutis, 2015]. Specifically, we have proposed a conjunctive normal form (S5-CNF) that is mainly based on using labels referring to semantic worlds. In a sense, S5-CNF can be seen as a generalization of the conjunctive normal form in propositional logic by including the modal connective of necessity and labels in the clause structure. We have showed that every S5 formula can be transformed into an S5-CNF formula using a linear encoding, and introduced a simple resolution method for S5, composed of three deductive rules that can be seen as adaptations of Robinson's resolution rule to the possible-worlds semantics. Much like the modal logic S4 (discussed in Section 4.2), S5 is a model of *interior algebra*, a proper extension of Boolean algebra originally designed to capture the properties of the interior and closure operators of topology [Jónsson and Tarski, 1951; Jónsson and Tarski, 1952]. As such, it would be interesing to explore how S5 could fit in the

context of some topology-based spatio-temporal framework and investigate whether the work in [Salhi and Sioutis, 2015] could be of any use towards that effort.

Chapter 7

Conclusion and Future Work

We dealt with Qualitative Spatial and Temporal Reasoning, a major field of study in Artificial Intelligence and, particularly, in Knowledge Representation, which deals with the fundamental cognitive concepts of space and time in an abstract manner. The qualitative manner of dealing with space and time is in line with the qualitative abstractions of spatial and temporal aspects of the common-sense background knowledge on which the human perspective of physical reality is based. Typically, qualitative spatial and temporal reasoning restricts the rich mathematical theories that deal with spatial and temporal entities to simple qualitative constraint languages. The conciseness of the constraint languages used in the qualitative approach provides a good framework that further boosts research and applications in spatial and temporal reasoning, as it allows for rather inexpensive reasoning about entities located in space and time. For example, some of these calculi may be implemented for handling spatial Geographic Information Systems (GIS) queries efficiently and some may be used for navigating and communicating with a mobile robot [Hazarika, 2012; Bhatt *et al.*, 2011]. Typical applications of temporal calculi involve planning and scheduling [Allen and Koomen, 1983; Allen, 1991; Pelavin and Allen, 1987; Dorn, 1995], natural language processing [Song and Cohen, 1988], temporal databases [Snodgrass, 1987; Chen and Zaniolo, 1998], multimedia databases [Little and Ghafoor, 1993], molecular biology [Golumbic and Shamir, 1993] (e.g., arrangement of DNA segments/intervals along a linear chain involves particular temporal-like problems [Benzer, 1959]), and workflow [Lu *et al.*, 2006], while typical applications of spatial calculi involve intelligent vehicles [Lattner *et al.*, 2005], high level vision, natural language processing [Bhatt *et al.*, 2011], and of course GIS and mobile robot navigation as mentioned earlier.

In this context, we pushed the envelope in the field of qualitative spatial and temporal reasoning by making contributions with respect to several of its key aspects. In particular, given a knowledge base of qualitative spatial or temporal information, we defined novel local consistency conditions and related techniques to efficiently solve the fundamental reasoning problems that are associated with such knowledge bases. These reasoning problems consist of the *satisfiability problem*, which is the problem of deciding whether there exists a quantitative interpretation of all the entities of a knowledge base such that all of its qualitative relations are satisfied by that interpretation (such an interpretation being called a *solution*), the *minimal labeling problem*, which is the problem of determining all the atoms for each of the qualitative relations of a knowledge base that participate in at least one of its solutions, and the *redundancy problem*, which is the problem of obtaining all the qualitative relations of a knowledge base that do not contain at least one atom participating in a solution of the modified knowledge base that results by removing these qualitative relations. Further, we enriched the field of spatio-temporal

formalisms that combine space and time in an interrelated manner by making contributions with respect to a qualitative spatio-temporal logic that results by combining the propositional temporal logic (PTL) with a qualitative spatial constraint language, and by investigating the task of ordering a temporal sequence of qualitative spatial configurations to meet certain transition constraints. Regarding the spatio-temporal logic, we also presented a first semantic tableau method that given a formula ϕ of that logic systematically searches for a model for ϕ.

Concerning future work, and in relation to the work we presented in this thesis, several possible directions are described in the corresponding sections of Chapters 5 and 6, namely, Sections 5.8 and 6.6 respectively. These directions involve contributions in the context of the fundamental reasoning problems in the field of qualitative constraint-based spatial and temporal reasoning, but also in the context of formalisms that combine spatial and temporal reasoning in an interrelated manner. In what follows, we will focus on future directions that extend outside the scope of this thesis.

Of particular interest is the problem of obtaining a spatial or temporal configuration that maximizes the number of satisfied constraints in a knowledge base of qualitative spatial or temporal information. The motivation behind studying this problem lies in the fact that representing spatial or temporal information may inevitably lead to inconsistencies. As illustration, due to the ever-increasing enrichment of the Semantic Web with geospatial data [Egenhofer, 2002; Koubarakis *et al.*, 2012], it is often the case that the geometries of geographical objects are not captured correctly due to contradictory data of different sources. Thus, we can obtain inconsistent topological information when extracting topological relations from such geometries (e.g., two overlapping regions may be stated to be identical to a third region, which is impossible as they would also have to be identical to each other if that was the case). With respect to temporal information, timetabling is an example of a scheduling problem where inconsistencies can naturally arise due to the lack of resources for certain tasks [Petrovic and Burke, 2004]. In particular, timetabling deals with finding suitable temporal intervals for a number of tasks that require limited resources. In the context of a university, an inconsistency can appear when two professors choose to teach the same class of students at overlapping temporal intervals. The inconsistency must then be repaired by taking into account the available temporal intervals and the preferences of the professors, and minimizing changes in the timetable so as to distort its structure as little as possible. Solving this problem is clearly at least as difficult as solving the satisfiability problem. To solve this optimization problem, we have already been actively involved in proposing in [Condotta *et al.*, 2015] a branch and bound algorithm based on the techniques for checking the satisfiability of a knowledge base of qualitative spatial or temporal information that were presented in Chapter 5, viz., the use of a triangulation of the constraint graph of the considered knowledge base to reduce the number of constraints to be treated, the use of a tractable subclass of relations to reduce the width of the search tree, and the use of partial $\diamond$-consistency to efficiently propagate constraints and prune non-feasible base relations during search. In a later work, we viewed this problem as a partial maximum satisfiability problem (PMAX-SAT) and proposed two related families of encodings [Condotta *et al.*, 2016]. That approach can be seen as similar to that concerning the satisfiability problem for which SAT encodings have been proposed to solve it [Pham *et al.*, 2008]. Each PMAX-SAT encoding is based on a forbidden covering with regard to the composition table of the considered qualitative calculus. Intuitively, a forbidden covering is a compact set of triples that express all the non-feasible configurations for three spatial or temporal entities. Interestingly, in a way, the support SAT encoding and the forbidden SAT encoding proposed in [Pham *et al.*, 2008] correspond to two particular coverings of our proposed forbidden coverings respectively. The two proposed families of PMAX-SAT encodings differ from one another in the use of auxiliary propositional variables that allow factorizing

the number of obtained clauses. It should be noted that the encodings also use the triangulation techniques presented in Chapter 5 to reduce the number of constraints to be translated. Both of the aforementioned works open up several research directions. In particular, future work consists of using other methods, like methods of local search, and comparing the behavior of our algorithms with these different methods, while the proposed techniques could be used to create algorithms for solving merging problems of spatial or temporal knowledge bases such as the one handled in [Condotta *et al.*, 2010]. With respect to SAT encodings in particular, future work consists of conducting experiments with several PMAX-SAT solvers to compare their behavior against the instances obtained through our proposed PMAX-SAT encodings. Another perspective consists of using forbidden coverings in the context of SAT encodings for the satisfiability problem of knowledge bases of qualitative spatial or temporal information.

Moving outside the field of qualitative spatial and temporal reasoning, and close to the field of graph theory, we would like to explore compact graph representations, which can be of a benefit to any field that deals with graphs, such fields typically ranging from data mining and social network analysis to constraint programming and machine learning. In [Liakos *et al.*, 2014b] we have improved the state of the art methods for the compression of web and other similar graphs by introducing an elegant technique that further exploits the clustering properties observed in these graphs. That work leaves much to be considered for future contributions. In particular, as the effectiveness of the implementation presented in the aforementioned work relies heavily on efficiently compressing the diagonal of the adjacency matrix that represents a given graph, choosing an appropriate set for representing values of that diagonal may become more effective by using some specialized heuristic. It is therefore important to explore such heuristics for an even better compression of a given graph. Further, we would like to investigate other reordering algorithms that could have a positive impact on the approach, but also understand the impact that the approach can have on graph related tasks such as querying or traversing the graph.

Bibliography

[Albert and Barabási, 2002] Réka Albert and Albert-László Barabási. Statistical mechanics of complex networks. *Rev. Mod. Phys.*, 74:47–97, 2002.

[Allen and Koomen, 1983] James F. Allen and Johannes A. G. M. Koomen. Planning Using a Temporal World Model. In *IJCAI*, 1983.

[Allen, 1981] James F. Allen. An Interval-Based Representation of Temporal Knowledge. In *IJCAI*, 1981.

[Allen, 1983] James F. Allen. Maintaining Knowledge about Temporal Intervals. *Commun. ACM*, 26:832–843, 1983.

[Allen, 1991] James F. Allen. Planning as Temporal Reasoning. In *KR*, 1991.

[Amaneddine and Condotta, 2012] Nouhad Amaneddine and Jean-François Condotta. From Path-Consistency to Global Consistency in Temporal Qualitative Constraint Networks. In *AIMSA*, pages 152–161, 2012.

[Amaneddine and Condotta, 2013] Nouhad Amaneddine and Jean-François Condotta. On the Minimal Labeling Problem of Temporal and Spatial Qualitative Constraints. In *FLAIRS*, 2013.

[Amaneddine *et al.*, 2013] Nouhad Amaneddine, Jean-François Condotta, and Michael Sioutis. Efficient Approach to Solve the Minimal Labeling Problem of Temporal and Spatial Qualitative Constraints. In *IJCAI*, 2013.

[Apt and Brand, 2006] Krzysztof R. Apt and Sebastian Brand. Infinite Qualitative Simulations by Means of Constraint Programming. In *CP*, 2006.

[Aristotle, 2004] Aristotle. *Posterior Analytics*. Kessinger Publishing, 2004.

[Baget and Tognetti, 2001] Jean-François Baget and Yannic S. Tognetti. Backtracking Through Biconnected Components of a Constraint Graph. In *IJCAI*, 2001.

[Balbiani and Condotta, 2002] Philippe Balbiani and Jean-François Condotta. Computational Complexity of Propositional Linear Temporal Logics Based on Qualitative Spatial or Temporal Reasoning. In *FroCoS*, 2002.

[Balbiani *et al.*, 1998] Philippe Balbiani, Jean-François Condotta, and Luis Fariñas del Cerro. A Model for Reasoning about Bidemsional Temporal Relations. In *KR*, 1998.

[Balbiani *et al.*, 1999] Philippe Balbiani, Jean-François Condotta, and Luis Fariñas del Cerro. A Tractable Subclass of the Block Algebra: Constraint Propagation and Preconvex Relations. In *EPIA*, 1999.

[Balbiani *et al.*, 2000] Philippe Balbiani, Jean-François Condotta, and Gérard Ligozat. Reasoning about Generalized Intervals: Horn Representability and Tractability. In *TIME*, 2000.

[Balbiani *et al.*, 2002] Philippe Balbiani, Jean-François Condotta, and Luis Fariñas del Cerro. Tractability Results in the Block Algebra. *J. Log. Comput.*, 12:885–909, 2002.

[Barabasi and Albert, 1999] A. L. Barabasi and R. Albert. Emergence of scaling in random networks. *Science (New York, N.Y.)*, 286:509–512, 1999.

[Barabasi and Bonabeau, 2003] Albert-Laszio Barabasi and Eric Bonabeau. Scale-Free Networks. *Scientific American*, pages 50–59, 2003.

[Bennett *et al.*, 2002] Brandon Bennett, Anthony G. Cohn, Frank Wolter, and Michael Zakharyaschev. Multi-dimensional modal logic as a framework for spatio-temporal reasoning. *Appl. Intell.*, 17:239–251, 2002.

[Bennett, 1996] Brandon Bennett. Modal Logics for Qualitative Spatial Reasoning. *Logic Journal of IGPL*, 4:23–45, 1996.

[Bennett, 1998] Brandon Bennett. Determining Consistency of Topological Relations. *Constraints*, 3:213–225, 1998.

[Bensalem *et al.*, 2007] Saddek Bensalem, Doron A. Peled, Hongyang Qu, Stavros Tripakis, and Lenore D. Zuck. Test Case Generation for Ultimately Periodic Paths. In *Haifa Verification Conference*, 2007.

[Benzer, 1959] Seymour Benzer. On the Topology of the Genetic Fine Structure. *Proceedings of the National Academy of Sciences of the United States of America*, 45:1607–1620, 1959.

[Berry *et al.*, 2002] Anne Berry, Jean R. S. Blair, and Pinar Heggernes. Maximum Cardinality Search for Computing Minimal Triangulations. In *WG*, 2002.

[Bertello *et al.*, 2016] Matteo Bertello, Nicola Gigante, Angelo Montanari, and Mark Reynolds. Leviathan: A New LTL Satisfiability Checking Tool Based on a One-Pass Tree-Shaped Tableau. In *IJCAI*, 2016.

[Bessière *et al.*, 1996] Christian Bessière, Amar Isli, and Gerard Ligozat. Global Consistency in Interval Algebra Networks: Tractable Subclasses. In *ECAI*, 1996.

[Beth, 1955] E.W. Beth. *Semantic Entailment and Formal Derivability.* Mededeelingen der Koninklijke Nederlandsche Akademie van Wetenschappen, Afd. Letterkunde. North-Holland, 1955.

[Bhatt *et al.*, 2011] Mehul Bhatt, Hans Guesgen, Stefan Wölfl, and Shyamanta Hazarika. Qualitative Spatial and Temporal Reasoning: Emerging Applications, Trends, and Directions. *Spatial Cognition & Computation*, 11:1–14, 2011.

[Bichot and Siarry, 2011] Charles-Edmond Bichot and Patrick Siarry. *Graph Partitioning.* ISTE-Wiley, 2011.

[Birkhoff, 1948] Garrett Birkhoff. *Lattice Theory*, volume 25 of *American Mathematical Society Colloquium Publications*. American Mathematical Society, 1948.

[Bliek and Sam-Haroud, 1999] Christian Bliek and Djamila Sam-Haroud. Path consistency on triangulated constraint graphs. In *IJCAI*, 1999.

[Bodirsky and Dalmau, 2006] Manuel Bodirsky and Víctor Dalmau. Datalog and Constraint Satisfaction with Infinite Templates. In *STACS*, 2006.

[Bodirsky and Dalmau, 2013] Manuel Bodirsky and Víctor Dalmau. Datalog and constraint satisfaction with infinite templates. *J. Comput. Syst. Sci.*, 79:79–100, 2013.

[Bodirsky and Wölfl, 2011] Manuel Bodirsky and Stefan Wölfl. RCC8 is polynomial on networks of bounded treewidth. In *IJCAI*, 2011.

[Bodlaender and Koster, 2010] Hans L. Bodlaender and Arie M. C. A. Koster. Treewidth computations I. Upper bounds. *Inf. Comput.*, 208:259–275, 2010.

[Bollobás, 2003] B. Bollobás. Mathematical results on scale-free random graphs. In *Handbook of Graphs and Networks*, pages 1–37. Wiley, 2003.

[Boussemart *et al.*, 2004] Frédéric Boussemart, Fred Hemery, Christophe Lecoutre, and Lakhdar Sais. Boosting Systematic Search by Weighting Constraints. In *ECAI*, 2004.

[Bouzy, 2001] Bruno Bouzy. Les concepts spatiaux dans la programmation du go. *Revue d'Intelligence Artificielle*, 15:143–172, 2001.

[Brand, 2004] Sebastian Brand. Relation Variables in Qualitative Spatial Reasoning. In *KI*, 2004.

[Broxvall, 2002] Mathias Broxvall. Constraint Satisfaction on Infinite Domains: Composing Domains and Decomposing Constraints. In *KR*, 2002.

[Burrieza and Ojeda-Aciego, 2005] Alfredo Burrieza and Manuel Ojeda-Aciego. A Multimodal Logic Approach to Order of Magnitude Qualitative Reasoning with Comparability and Negligibility Relations. *Fundam. Inform.*, 68:21–46, 2005.

[Burrieza *et al.*, 2009] Alfredo Burrieza, Emilio Muñoz-Velasco, and Manuel Ojeda-Aciego. Closeness and Distance Relations in Order of Magnitude Qualitative Reasoning via PDL. In *CAEPIA*, 2009.

[Burrieza *et al.*, 2011] Alfredo Burrieza, Emilio Muñoz-Velasco, and Manuel Ojeda-Aciego. A PDL Approach for Qualitative Velocity. *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems*, 19(1):11–26, 2011.

[Cano and Moral, 1994] Andrés Cano and Serafín Moral. Heuristic Algorithms for the Triangulation of Graphs. In *IPMU*, 1994.

[Chagrov and Zakharyaschev, 1997] Alexander V. Chagrov and Michael Zakharyaschev. *Modal Logic*, volume 35 of *Oxford logic guides*. Oxford University Press, 1997.

[Challita, 2012] Khalil Challita. A semi-dynamical approach for solving qualitative spatial constraint satisfaction problems. *Theor. Comput. Sci.*, 440-441:29–38, 2012.

[Chandra and Pujari, 2005] Priti Chandra and Arun K. Pujari. Minimality and Convexity Properties in Spatial CSPs. In *ICTAI*, 2005.

[Chen and Zaniolo, 1998] Cindy Xinmin Chen and Carlo Zaniolo. Universal Temporal Data Languages. In *DDLP*, 1998.

[Chmeiss and Condotta, 2011] Assef Chmeiss and Jean-Francois Condotta. Consistency of Triangulated Temporal Qualitative Constraint Networks. In *ICTAI*, 2011.

[Choromański *et al.*, 2013] Krzysztof Choromański, Michal Matuszak, and Jacek Miekisz. Scale-free graph with preferential attachment and evolving internal vertex structure. *J. Stat. Phys.*, 151:1175–1183, 2013.

[Clarke, 1981] Bowman L Clarke. A calculus of individuals based on "connection". *NDJFL*, 22:204–218, 1981.

[Clementini and Di Felice, 1995] Eliseo Clementini and Paolino Di Felice. A comparison of methods for representing topological relationships. *IJISA*, 3:149–178, 1995.

[Clementini *et al.*, 1993] Eliseo Clementini, Paolino Di Felice, and Peter van Oosterom. A Small Set of Formal Topological Relationships Suitable for End-User Interaction. In *SSD*, 1993.

[Clementini *et al.*, 1994] Eliseo Clementini, Jayant Sharma, and Max J. Egenhofer. Modelling topological spatial relations: Strategies for query processing. *Computers & Graphics*, 18:815–822, 1994.

[Cohn *et al.*, 1997] Anthony G. Cohn, Brandon Bennett, John Gooday, and Nicholas Mark Gotts. Qualitative Spatial Representation and Reasoning with the Region Connection Calculus. *GeoInformatica*, 1:275–316, 1997.

[Cohn, 1997] Anthony G. Cohn. Qualitative Spatial Representation and Reasoning Techniques. In *KI*, 1997.

[Condotta and D'Almeida, 2011] Jean-François Condotta and Dominique D'Almeida. Consistency of Qualitative Constraint Networks from Tree Decompositions. In *TIME*, 2011.

[Condotta and Lecoutre, 2010] Jean-François Condotta and Christophe Lecoutre. A Class of df-Consistencies for Qualitative Constraint Networks. In *KR*, 2010.

[Condotta *et al.*, 2005] Jean-François Condotta, Gérard Ligozat, and Stavros Tripakis. Ultimately Periodic Qualitative Constraint Networks for Spatial and Temporal Reasoning. In *ICTAI*, 2005.

[Condotta *et al.*, 2006a] Jean François Condotta, Gérard Ligozat, and Mahmood Saade. An empirical study of algorithms for qualitative temporal or spatial networks. In *ECAI Workshop on Spatial and Temporal Reasoning*, 2006.

[Condotta *et al.*, 2006b] Jean-Francois Condotta, Dominique Dalmeida, Christophe Lecoutre, and Lahkdar Sais. From Qualitative to Discrete Constraint Networks. In *KI Workshop on Qualitative Constraint Calculi*, 2006.

[Condotta *et al.*, 2006c] Jean-François Condotta, Gérard Ligozat, Mahmoud Saade, and Stavros Tripakis. Ultimately Periodic Simple Temporal Problems (UPSTPs). In *TIME*, 2006.

[Condotta *et al.*, 2007] Jean-François Condotta, Gérard Ligozat, and Mahmoud Saade. Eligible and Frozen Constraints for Solving Temporal Qualitative Constraint Networks. In *CP*, 2007.

[Condotta *et al.*, 2008] Jean-François Condotta, Souhila Kaci, and Nicolas Schwind. A Framework for Merging Qualitative Constraints Networks. In *FLAIRS*, 2008.

[Condotta *et al.*, 2009] Jean-François Condotta, Souhila Kaci, Pierre Marquis, and Nicolas Schwind. Merging Qualitative Constraint Networks in a Piecewise Fashion. In *ICTAI*, 2009.

[Condotta *et al.*, 2010] Jean-François Condotta, Souhila Kaci, Pierre Marquis, and Nicolas Schwind. A syntactical approach to qualitative constraint networks merging. In *Proceedings of LPAR-17*, pages 233–247, 2010.

[Condotta *et al.*, 2015] Jean-François Condotta, Ali Mensi, Issam Nouaouri, Michael Sioutis, and Lamjed Ben Said. A Practical Approach for Maximizing Satisfiability in Qualitative Spatial and Temporal Constraint Networks. In *ICTAI*, 2015.

[Condotta *et al.*, 2016] Jean-François Condotta, Issam Nouaouri, and Michael Sioutis. A SAT Approach for Maximizing Satisfiability in Qualitative Spatial and Temporal Constraint Networks. In *KR*, 2016.

[Condotta, 2004] Jean-François Condotta. A General Qualitative Framework for Temporal and Spatial Reasoning. *Constraints*, 9:99–121, 2004.

[Creignou *et al.*, 2001] N. Creignou, S. Khanna, and M. Sudan. *Complexity Classifications of Boolean Constraint Satisfaction Problems*. Monographs on Discrete Mathematics and Applications. Society for Industrial and Applied Mathematics, 2001.

[Cui *et al.*, 1992] Zhan Cui, Anthony G. Cohn, and David A. Randell. Qualitative Simulation Based on a Logical Formalism of Space and Time. In *AAAI*, 1992.

[Danzer *et al.*, 1963] Ludwig Danzer, Branko Grünbaum, and Victor Klee. *Helly's Theorem and Its Relatives*, volume 7 of *Proceedings of symposia in pure mathematics: Convexity*. American Mathematical Society, 1963.

[de Leng and Heintz, 2016] Daniel de Leng and Fredrik Heintz. Qualitative Spatio-Temporal Stream Reasoning with Unobservable Intertemporal Spatial Relations Using Landmarks. In *AAAI*, 2016.

[Debruyne and Bessière, 1997] Romuald Debruyne and Christian Bessière. Some Practicable Filtering Techniques for the Constraint Satisfaction Problem. In *IJCAI*, 1997.

[Dechter and Pearl, 1989] Rina Dechter and Judea Pearl. Tree Clustering for Constraint Networks. *AIJ*, 38:353–366, 1989.

[Dechter *et al.*, 1991] Rina Dechter, Itay Meiri, and Judea Pearl. Temporal Constraint Networks. *AIJ*, 49:61–95, 1991.

[Dechter, 2003] Rina Dechter. *Constraint processing*. Elsevier Morgan Kaufmann, 2003.

[Del Genio *et al.*, 2011] Charo I. Del Genio, Thilo Gross, and Kevin E. Bassler. All Scale-Free Networks Are Sparse. *Phys. Rev. Lett.*, 107:178701, 2011.

[Demri and D'Souza, 2002] Stéphane Demri and Deepak D'Souza. An Automata-Theoretic Approach to Constraint LTL. In *FSTTCS*, 2002.

[Demri and D'Souza, 2007] Stéphane Demri and Deepak D'Souza. An automata-theoretic approach to constraint LTL. *Inf. Comput.*, 205:380–415, 2007.

[Deville *et al.*, 1999] Yves Deville, Olivier Barette, and Pascal van Hentenryck. Constraint Satisfaction over Connected Row Convex Constraints. *AIJ*, 109:243–271, 1999.

[Diestel, 2012] Reinhard Diestel. *Graph Theory, 4th Edition*, volume 173 of *Graduate texts in mathematics*. Springer, 2012.

[Dorn, 1995] Jürgen Dorn. Dependable Reactive Event-Oriented Planning. *Data Knowl. Eng.*, 16:27–49, 1995.

[Dorogovtsev *et al.*, 2002] S. N. Dorogovtsev, A. V. Goltsev, and J. F. F. Mendes. Pseudofractal scale-free web. *Physical Review E*, 65:066122+, 2002.

[Duckham *et al.*, 2014] Matt Duckham, Sanjiang Li, Weiming Liu, and Zhiguo Long. On Redundant Topological Constraints. In *KR*, 2014.

[Dushnik and Miller, 1941] Ben Dushnik and E. W. Miller. Partially Ordered Sets. *American Journal of Mathematics*, 63:pp. 600–610, 1941.

[Dylla *et al.*, 2013] Frank Dylla, Till Mossakowski, Thomas Schneider, and Diedrich Wolter. Algebraic Properties of Qualitative Spatio-temporal Calculi. In *COSIT*, 2013.

[Egenhofer and Herring, 1991] Max J. Egenhofer and John Herring. *Categorizing Binary Topological Relationships Between Regions, Lines, and Points in Geographic Databases*. Department of Surveying Engineering, University of Maine, 1991.

[Egenhofer and Sharma, 1993] Max J. Egenhofer and Jayant Sharma. Assessing the consistency of complete and incomplete topological information. *Geographical Systems*, 1:47–68, 1993.

[Egenhofer *et al.*, 1994] Max J. Egenhofer, Eliseo Clementini, and Paolino Di Felice. Topological Relations Between Regions with Holes. *IJGIS*, 8:129–142, 1994.

[Egenhofer, 2002] Max J. Egenhofer. Toward the semantic geospatial web. In *ACM-GIS*, 2002.

[Egenhofer, 2010] Max J. Egenhofer. The Family of Conceptual Neighborhood Graphs for Region-Region Relations. In *GIScience*, 2010.

[Frank, 1991] Andrew U. Frank. Qualitative Spatial Reasoning with Cardinal Directions. In *ÖGAI*, 1991.

[Frank, 1992] Andrew U. Frank. Qualitative spatial reasoning about distances and directions in geographic space. *JVLC*, 3:343–371, 1992.

[Freksa, 1991] C. Freksa. Conceptual neighborhood and its role in temporal and spatial reasoning. *Decision Support Systems and Qualitative Reasoning*, pages 181–187, 1991.

[Freksa, 1992] Christian Freksa. Using orientation information for qualitative spatial reasoning. In *GIS - From Space to Territory: Theories and Methods of Spatio-Temporal Reasoning*, 1992.

[Fulkerson and Gross, 1965] D. R. Fulkerson and O. A. Gross. Incidence matrices and interval graphs. *Pacific J. Math.*, 15:835–855, 1965.

[Gabelaia *et al.*, 2003] David Gabelaia, Roman Kontchakov, Agi Kurucz, Frank Wolter, and Michael Zakharyaschev. On the Computational Complexity of Spatio-Temporal Logics. In *FLAIRS*, 2003.

[Gabelaia *et al.*, 2005] David Gabelaia, Roman Kontchakov, Ágnes Kurucz, Frank Wolter, and Michael Zakharyaschev. Combining Spatial and Temporal Logics: Expressiveness vs. Complexity. *JAIR*, pages 167–243, 2005.

[Gaintzarain *et al.*, 2008] Joxe Gaintzarain, Montserrat Hermo, Paqui Lucio, and Marisa Navarro. Systematic Semantic Tableaux for PLTL. *Electr. Notes Theor. Comput. Sci.*, 206:59–73, 2008.

[Galton, 1997] Antony Galton. Continuous Change in Spatial Region. In *COSIT*, 1997.

[Galton, 2000] Antony Galton. Continuous Motion in Discrete Space. In *KR*, 2000.

[Galton, 2001] Antony Galton. Dominance Diagrams: A Tool for Qualitative Reasoning About Continuous Systems. *Fundam. Inform.*, 46:55–70, 2001.

[Gantner *et al.*, 2008] Zeno Gantner, Matthias Westphal, and Stefan Wölfl. GQR-A Fast Reasoner for Binary Qualitative Constraint Calculi. In *AAAI Workshop on Spatial and Temporal Reasoning*, 2008.

[Garey and Johnson, 1979] Michael R. Garey and David S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness.* W. H. Freeman & Co., 1979.

[Garey *et al.*, 1976] M. R. Garey, David S. Johnson, and Larry J. Stockmeyer. Some Simplified NP-Complete Graph Problems. *Theor. Comput. Sci.*, 1:237–267, 1976.

[Gerevini and Nebel, 2002] Alfonso Gerevini and Bernhard Nebel. Qualitative Spatio-Temporal Reasoning with RCC-8 and Allen's Interval Calculus: Computational Complexity. In *ECAI*, pages 312–316, 2002.

[Gerevini and Renz, 2002] Alfonso Gerevini and Jochen Renz. Combining topological and size information for spatial reasoning. *AIJ*, 137:1–42, 2002.

[Gerevini and Saetti, 2007] Alfonso Gerevini and Alessandro Saetti. Efficient Computation of Minimal Point Algebra Constraints by Metagraph Closure. In *CP*, 2007.

[Gerevini and Saetti, 2011] Alfonso Gerevini and Alessandro Saetti. Computing the minimal relations in point-based qualitative temporal reasoning through metagraph closure. *AIJ*, 175:556–585, 2011.

[Gerevini and Schubert, 1995] Alfonso Gerevini and Lenhart K. Schubert. On Computing the Minimal Labels in Time Point Algebra Networks. *Computational Intelligence*, 11:443–448, 1995.

[Gerevini, 2005] Alfonso Gerevini. Incremental qualitative temporal reasoning: Algorithms for the Point Algebra and the ORD-Horn class. *AIJ*, 166:37–80, 2005.

[Girle, 2000] R. Girle. *Modal Logics and Philosophy.* McGill-Queen's University Press, 2000.

[Golinska-Pilarek and Muñoz-Velasco, 2012] Joanna Golinska-Pilarek and Emilio Muñoz-Velasco. Reasoning with Qualitative Velocity: Towards a Hybrid Approach. In *HAIS*, 2012.

[Golumbic and Shamir, 1993] Martin Charles Golumbic and Ron Shamir. Complexity and Algorithms for Reasoning about Time: A Graph-Theoretic Approach. *J. ACM*, 40:1108–1133, 1993.

[Golumbic, 2004] M. C. Golumbic. *Algorithmic Graph Theory and Perfect Graphs.* Elsevier Science, 2nd edition, 2004.

[Gooday and Galton, 1997] John Gooday and Antony Galton. The Transition Calculus: a high-level formalism for reasoning about action and change. *J. Exp. Theor. Artif. Intell.*, 9:51–66, 1997.

[Goodwin *et al.*, 2008] John Goodwin, Catherine Dolbear, and Glen Hart. Geographical Linked Data: The Administrative Geography of Great Britain on the Semantic Web. *TGIS*, 12:19–30, 2008.

[Goranko and Passy, 1992] Valentin Goranko and Solomon Passy. Using the Universal Modality: Gains and Questions. *J. Log. Comput.*, 2:5–30, 1992.

[Gottlob *et al.*, 2000] Georg Gottlob, Nicola Leone, and Francesco Scarcello. A comparison of structural CSP decomposition methods. *AIJ*, 124:243–282, 2000.

[Gottlob, 2012] Georg Gottlob. On minimal constraint networks. *AIJ*, 191-192:42–60, 2012.

[Gotts, 1994] Nicholas Mark Gotts. How Far Can We 'C'? Defining a 'Doughnut' Using Connection Alone. In *KR*, 1994.

[Gotts, 1996] Nicholas Mark Gotts. Topology From A Single Primitive Relation: Defining Topological Properties and Relations In Terms Of Connection. Technical report, School of Computer Studies, University of Leeds, 1996.

[Grigni *et al.*, 1995] Michelangelo Grigni, Dimitris Papadias, and Christos H. Papadimitriou. Topological Inference. In *IJCAI*, 1995.

[Guesgen, 1989] Hans Werner Guesgen. Spatial Reasoning Based on Allen's Temporal Logic. Technical report, International Computer Science Institute, 1989.

[Halin, 1976] Rudolf Halin. S-functions for graphs. *Journal of Geometry*, 8:171–186, 1976.

[Harel *et al.*, 2000] David Harel, Jerzy Tiuryn, and Dexter Kozen. *Dynamic Logic.* MIT Press, Cambridge, MA, USA, 2000.

[Hazarika, 2012] S.M. Hazarika. *Qualitative Spatio-Temporal Representation and Reasoning: Trends and Future Directions.* Igi Global, 2012.

[Heggernes *et al.*, 2001] P. Heggernes, Stanley C. Eisenstat, Gary Kumfert, and Alex Pothen. The computational complexity of the minimum degree algorithm. Technical report, ICASE, NASA Langley Research Center, 2001.

[Hein *et al.*, 2006] Oliver Hein, Michael Schwind, and Wolfgang König. Scale-Free Networks - The Impact of Fat Tailed Degree Distribution on Diffusion and Communication Processes. *Wirtschaftsinformatik*, 47:21–28, 2006.

[Heintz and de Leng, 2014] Fredrik Heintz and Daniel de Leng. Spatio-Temporal Stream Reasoning with Incomplete Spatial Information. In *ECAI*, 2014.

[Hernández, 1994] Daniel Hernández. *Qualitative Representation of Spatial Knowledge*, volume 804 of *Lecture Notes in Computer Science*. Springer, 1994.

[Hodges, 1997] Wilfrid Hodges. *A Shorter Model Theory*. Cambridge University Press, 1997.

[Hogge, 1987] John C. Hogge. TPLAN: a temporal interval-based planner with novel extensions. Technical report, University of Illinois at Urbana-Champaign. Department of Computer Science UIUCDCS-R-87-1367., 1987.

[Huang *et al.*, 2013] Jinbo Huang, Jason Jingshi Li, and Jochen Renz. Decomposition and tractability in qualitative spatial and temporal reasoning. *AIJ*, 195:140–164, 2013.

[Huang, 2012] Jinbo Huang. Compactness and its implications for qualitative spatial and temporal reasoning. In *KR*, 2012.

[Huth and Ryan, 2004] Michael Huth and Mark Ryan. *Logic in Computer Science: Modelling and Reasoning About Systems*. Cambridge University Press, 2004.

[Isli and Cohn, 2000] Amar Isli and Anthony G. Cohn. A new approach to cyclic ordering of 2D orientations using ternary relation algebras. *AIJ*, 122:137–187, 2000.

[Jégou and Terrioux, 2003] Philippe Jégou and Cyril Terrioux. Hybrid backtracking bounded by tree-decomposition of constraint networks. *AIJ*, 146:43–75, 2003.

[Jégou and Terrioux, 2014a] Philippe Jégou and Cyril Terrioux. Bag-Connected Tree-Width: A New Parameter for Graph Decomposition. In *ISAIM*, 2014.

[Jégou and Terrioux, 2014b] Philippe Jégou and Cyril Terrioux. Tree-Decompositions with Connected Clusters for Solving Constraint Networks. In *CP*, 2014.

[Jégou *et al.*, 2005] Philippe Jégou, Samba Ndiaye, and Cyril Terrioux. Computing and Exploiting Tree-Decompositions for Solving Constraint Networks. In *CP*, 2005.

[Jégou *et al.*, 2006] Philippe Jégou, Samba Ndiaye, and Cyril Terrioux. An Extension of Complexity Bounds and Dynamic Heuristics for Tree-Decompositions of CSP. In *CP*, 2006.

[Jégou *et al.*, 2007] Philippe Jégou, Samba Ndiaye, and Cyril Terrioux. Dynamic Heuristics for Backtrack Search on Tree-Decomposition of CSPs. In *IJCAI*, 2007.

[Jónsson and Tarski, 1951] Bjarni Jónsson and Alfred Tarski. Boolean algebras with operators. I. *Amer. J. Math.*, 73:891–939, 1951.

[Jónsson and Tarski, 1952] Bjarni Jónsson and Alfred Tarski. Boolean algebras with operators. II. *Amer. J. Math.*, 74:127–162, 1952.

[Kann, 1995] Viggo Kann. Strong Lower Bounds on the Approximability of some NPO PB-Complete Maximization Problems. In *MFCS*, 1995.

[Katsirelos and Bacchus, 2005] George Katsirelos and Fahiem Bacchus. Generalized NoGoods in CSPs. In *AAAI*, 2005.

[Knuth, 1973] Donald E. Knuth. *The Art of Computer Programming, Volume I: Fundamental Algorithms, 2nd Edition.* Addison-Wesley, 1973.

[Kondrak and van Beek, 1995] Grzegorz Kondrak and Peter van Beek. A Theoretical Evaluation of Selected Backtracking Algorithms. In *IJCAI*, 1995.

[Kondrak and van Beek, 1997] Grzegorz Kondrak and Peter van Beek. A Theoretical Evaluation of Selected Backtracking Algorithms. *AIJ*, 89:365–387, 1997.

[Kontchakov *et al.*, 2007] Roman Kontchakov, Agi Kurucz, Frank Wolter, and Michael Zakharyaschev. Spatial Logic + Temporal Logic = ? In *Handbook of Spatial Logics*, pages 497–564. 2007.

[Koubarakis *et al.*, 2011] Manolis Koubarakis, Kostis Kyzirakos, Manos Karpathiotakis, Charalampos Nikolaou, Michael Sioutis, Stavros Vassos, Dimitrios Michail, Themistoklis Herekakis, Charalampos Kontoes, and Ioannis Papoutsis. Challenges for Qualitative Spatial Reasoning in Linked Geospatial Data. In *BASR@IJCAI*, 2011.

[Koubarakis *et al.*, 2012] Manolis Koubarakis, Manos Karpathiotakis, Kostis Kyzirakos, Charalampos Nikolaou, and Michael Sioutis. Data Models and Query Languages for Linked Geospatial Data. In *Reasoning Web*, 2012.

[Krentel, 1988] Mark W. Krentel. The Complexity of Optimization Problems. *J. Comput. Syst. Sci.*, 36:490–509, 1988.

[Krokhin *et al.*, 2001] Andrei A. Krokhin, Peter Jeavons, and Peter Jonsson. A Complete Classification of Complexity in Allens Algebra in the Presence of a Non-Trivial Basic Relation. In *IJCAI*, 2001.

[Krokhin *et al.*, 2003] Andrei A. Krokhin, Peter Jeavons, and Peter Jonsson. Reasoning about temporal relations: The tractable subalgebras of Allen's interval algebra. *J. ACM*, 50, 2003.

[Kuipers, 1985] Benjamin Kuipers. The Limits of Qualitative Simulation. In *IJCAI*, 1985.

[Kuipers, 1986] Benjamin Kuipers. Qualitative Simulation. *AIJ*, 29:289–338, 1986.

[Kuipers, 1993] Benjamin Kuipers. Qualitative Simulation: Then and Now. *AIJ*, 59(1-2), 1993.

[Kuipers, 1994] Benjamin Kuipers. *Qualitative reasoning - modeling and simulation with incomplete knowledge.* MIT Press, 1994.

[Ladkin and Maddux, 1994] Peter B. Ladkin and Roger D. Maddux. On binary constraint problems. *JACM*, 41:435–469, 1994.

[Ladkin and Reinefeld, 1992] Peter B. Ladkin and Alexander Reinefeld. Effective Solution of Qualitative Interval Constraint Problems. *AIJ*, 57:105–124, 1992.

[Ladkin and Reinefeld, 1997] Peter B. Ladkin and Alexander Reinefeld. Fast Algebraic Methods for Interval Constraint Problems. *AMAI*, 19:383–411, 1997.

[Ladkin, 1986] Peter B. Ladkin. Time Representation: A Taxonomy of Internal Relations. In *AAAI*, 1986.

[Ladner, 1977] Richard E. Ladner. The Computational Complexity of Provability in Systems of Modal Propositional Logic. *SIAM J. Comput.*, 6:467–480, 1977.

[Lattner *et al.*, 2005] A. D. Lattner, I. J. Timm, M. Lorenz, and O. Herzog. Knowledge-based risk assessment for intelligent vehicles. In *International Conference on Integration of Knowledge Intensive Multi-Agent Systems*, 2005.

[Lecoutre *et al.*, 2006] Christophe Lecoutre, Lakhdar Sais, Sébastien Tabary, and Vincent Vidal. Last Conflict Based Reasoning. In *ECAI*, 2006.

[Lecoutre *et al.*, 2007] Christophe Lecoutre, Lakhdar Sais, Sébastien Tabary, and Vincent Vidal. Recording and Minimizing Nogoods from Restarts. *JSAT*, 1:147–167, 2007.

[Lecoutre *et al.*, 2009] Christophe Lecoutre, Lakhdar Sais, Sébastien Tabary, and Vincent Vidal. Reasoning from last conflict(s) in constraint programming. *AIJ*, 173:1592–1614, 2009.

[Lee *et al.*, 2016] Jae Hee Lee, Sanjiang Li, Zhiguo Long, and Michael Sioutis. On Redundancy in Simple Temporal Networks. In *ECAI*, 2016.

[Lewis and Langford, 1932] C.I. Lewis and C.H. Langford. *Symbolic logic*. Century philosophy series. The Century co., 1932.

[Li and Wang, 2006] Sanjiang Li and Huaiqing Wang. RCC8 binary constraint network can be consistently extended. *AIJ*, 170:1–18, 2006.

[Li *et al.*, 2009] Jason Jingshi Li, Jinbo Huang, and Jochen Renz. A divide-and-conquer approach for solving interval algebra networks. In *IJCAI*, 2009.

[Li *et al.*, 2013] Sanjiang Li, Weiming Liu, and Sheng-sheng Wang. Qualitative constraint satisfaction problems: An extended framework with landmarks. *Artif. Intell.*, 201:32–58, 2013.

[Li *et al.*, 2015a] Sanjiang Li, Zhiguo Long, Weiming Liu, Matt Duckham, and Alan Both. On redundant topological constraints. *AIJ*, 225:51–76, 2015. In press.

[Li *et al.*, 2015b] Sanjiang Li, Zhiguo Long, Weiming Liu, Matt Duckham, and Alan Both. On redundant topological constraints. *AIJ*, 225:51–76, 2015.

[Li, 2006] Sanjiang Li. On Topological Consistency and Realization. *Constraints*, 11:31–51, 2006.

[Liakos *et al.*, 2014a] Panagiotis Liakos, Katia Papakonstantinopoulou, and Michael Sioutis. On the Effect of Locality in Compressing Social Networks. In *ECIR*, 2014.

[Liakos *et al.*, 2014b] Panagiotis Liakos, Katia Papakonstantinopoulou, and Michael Sioutis. Pushing the Envelope in Graph Compression. In *CIKM*, 2014.

[Ligozat and Renz, 2004] Gérard Ligozat and Jochen Renz. What Is a Qualitative Calculus? A General Framework. In *PRICAI*, 2004.

[Ligozat, 1991] Gerard Ligozat. On Generalized Interval Calculi. In *AAAI*, 1991.

[Ligozat, 1994] Gérard Ligozat. Tractable relations in temporal reasoning: pre-convex relations. In *ECAI Workshop on Spatial and Temporal Reasoning*, 1994.

[Ligozat, 1996] Gerard Ligozat. A New Proof of Tractability for ORD-Horn Relations. In *AAAI/IAAI*, 1996.

[Ligozat, 1998] Gerard Ligozat. Reasoning about cardinal directions. *JVLC*, 9:23–44, 1998.

[Ligozat, 2011] Gérard Ligozat. *Qualitative Spatial and Temporal Reasoning.* Iste Series. Wiley, 2011.

[Little and Ghafoor, 1993] Thomas D. C. Little and Arif Ghafoor. Interval-Based Conceptual Models for Time-Dependent Multimedia Data. *IEEE Trans. Knowl. Data Eng.*, 5:551–563, 1993.

[Liu and Li, 2012] Weiming Liu and Sanjiang Li. Solving Minimal Constraint Networks in Qualitative Spatial and Temporal Reasoning. In *CP*, 2012.

[Liu *et al.*, 2011] Weiming Liu, Sheng-sheng Wang, Sanjiang Li, and Dayou Liu. Solving Qualitative Constraints Involving Landmarks. In *CP*, 2011.

[Long and Li, 2015] Zhiguo Long and Sanjiang Li. On Distributive Subalgebras of Qualitative Spatial and Temporal Calculi. In *COSIT*, 2015.

[Long *et al.*, 2016] Zhiguo Long, Michael Sioutis, and Sanjiang Li. Efficient Path Consistency Algorithm for Large Qualitative Constraint Networks. In *IJCAI*, 2016.

[Lu *et al.*, 2006] Ruopeng Lu, Shazia Wasim Sadiq, Vineet Padmanabhan, and Guido Governatori. Using a temporal constraint network for business process execution. In *ADC*, 2006.

[Lutz and Milicic, 2007] C. Lutz and M. Milicic. A Tableau Algorithm for DLs with Concrete Domains and GCIs. *JAR*, 38:227–259, 2007.

[Mackworth and Freuder, 1985] Alan K. Mackworth and Eugene C. Freuder. The Complexity of Some Polynomial Network Consistency Algorithms for Constraint Satisfaction Problems. *AIJ*, 25:65–74, 1985.

[Mackworth, 1977] Alan Mackworth. Consistency in Networks of Relations. *AIJ*, 8:99–118, 1977.

[McKinsey and Tarski, 1948] J. C. C. McKinsey and Alfred Tarski. Some Theorems About the Sentential Calculi of Lewis and Heyting. *J. Symb. Log.*, 13:1–15, 1948.

[Mitra, 2002] Debasis Mitra. A Class of Star-Algebras for Point-Based Qualitative Reasoning in Two-Dimensional Space. In *FLAIRS*, 2002.

[Mitra, 2004] Debasis Mitra. Modeling and Reasoning with Star Calculus. In *ISAIM*, 2004.

[Montanari, 1974] Ugo Montanari. Networks of constraints: Fundamental properties and applications to picture processing. *Inf. Sci.*, 7:95–132, 1974.

[Moratz *et al.*, 2005] Reinhard Moratz, Frank Dylla, and Lutz Frommberger. A relative orientation algebra with adjustable granularity. In *IJCAI Workshop on Agents in Real-Time and Dynamic Environments*, 2005.

[Moratz, 2006] Reinhard Moratz. Representing Relative Direction as a Binary Relation of Oriented Points. In *ECAI*, 2006.

[Mossakowski and Moratz, 2012] Till Mossakowski and Reinhard Moratz. Qualitative reasoning about relative direction of oriented points. *AIJ*, 180-181:34–45, 2012.

[Muller, 1998] Philippe Muller. A Qualitative Theory of Motion Based on Spatio-Temporal Primitives. In *KR*, 1998.

[Muller, 2002] Philippe Muller. Topological Spatio-Temporal Reasoning and Representation. *Computational Intelligence*, 18:420–450, 2002.

[Munkres, 2000] J.R. Munkres. *Topology*. Prentice Hall, Incorporated, 2000.

[Muñoz-Velasco *et al.*, 2014] Emilio Muñoz-Velasco, Alfredo Burrieza, and Manuel Ojeda-Aciego. A logic framework for reasoning with movement based on fuzzy qualitative representation. *Fuzzy Sets and Systems*, 242:114–131, 2014.

[Navarrete *et al.*, 2013] Isabel Navarrete, Antonio Morales, Guido Sciavicco, and M. Antonia Cárdenas Viedma. Spatial reasoning with rectangular cardinal relations - The convex tractable subalgebra. *AMAI*, 67:31–70, 2013.

[Nebel and Bürckert, 1995] Bernhard Nebel and Hans-Jürgen Bürckert. Reasoning about Temporal Relations: A Maximal Tractable Subclass of Allen's Interval Algebra. *JACM*, 42:43–66, 1995.

[Nebel, 1995] Bernhard Nebel. Computational Properties of Qualitative Spatial Reasoning: First Results. In *KI*, 1995.

[Nebel, 1996] Bernhard Nebel. Solving Hard Qualitative Temporal Reasoning Problems: Evaluating the Efficiency of Using the ORD-Horn Class. In *ECAI*, 1996.

[Nebel, 1997] Bernhard Nebel. Solving Hard Qualitative Temporal Reasoning Problems: Evaluating the Efficiency of Using the ORD-Horn Class. *Constraints*, 1:175–190, 1997.

[Nikolaou and Koubarakis, 2013] Charalampos Nikolaou and Manolis Koubarakis. Querying Incomplete Geospatial Information in RDF. In *SSTD*, 2013.

[Nikolaou and Koubarakis, 2014] Charalampos Nikolaou and Manolis Koubarakis. Fast Consistency Checking of Very Large Real-World RCC-8 Constraint Networks Using Graph Partitioning. In *AAAI*, 2014.

[Open Geospatial Consortium, 2012] Open Geospatial Consortium. OGC GeoSPARQL - A geographic query language for RDF data. OGC® Implementation Standard, 2012.

[Parter, 1961] Seymour Parter. The use of linear graphs in Gauss elimination. *SIAM review*, 3:119–130, 1961.

[Pelavin and Allen, 1987] Richard N. Pelavin and James F. Allen. A Model for Concurrent Actions Having Temporal Extent. In *AAAI*, 1987.

[Petrovic and Burke, 2004] Sanja Petrovic and Edmund K. Burke. University Timetabling. In *Handbook of Scheduling: Algorithms, Models, and Performance Analysis*. 2004.

[Pham *et al.*, 2008] Duc Nghia Pham, John Thornton, and Abdul Sattar. Modelling and solving temporal reasoning as propositional satisfiability. *Artif. Intell.*, 172:1752–1782, 2008.

[Planken *et al.*, 2010] Léon Planken, Mathijs de Weerdt, and Neil Yorke-Smith. Incrementally Solving STNs by Enforcing Partial Path Consistency. In *ICAPS*, 2010.

[Pnueli, 1977] Amir Pnueli. The Temporal Logic of Programs. In *FOCS*, 1977.

[Preparata and Shamos, 1985] Franco P. Preparata and Michael Ian Shamos. *Computational Geometry - An Introduction*. Springer, 1985.

[Ragni and Wölfl, 2005] Marco Ragni and Stefan Wölfl. Temporalizing Spatial Calculi: On Generalized Neighborhood Graphs. In *KI*, 2005.

[Ragni and Wölfl, 2006] Marco Ragni and Stefan Wölfl. Temporalizing Cardinal Directions: From Constraint Satisfaction to Planning. In *KR*, 2006.

[Randell *et al.*, 1992] David A. Randell, Zhan Cui, and Anthony Cohn. A Spatial Logic Based on Regions and Connection. In *KR*, 1992.

[Randell *et al.*, 2013] David A. Randell, Gabriel Landini, and Antony Galton. Discrete Mereotopology for Spatial Reasoning in Automated Histological Image Analysis. *TPAMI*, 35:568–581, 2013.

[Renz and Ligozat, 2005] Jochen Renz and Gérard Ligozat. Weak Composition for Qualitative Spatial and Temporal Reasoning. In *CP*, 2005.

[Renz and Mitra, 2004] Jochen Renz and Debasis Mitra. Qualitative Direction Calculi with Arbitrary Granularity. In *PRICAI*, 2004.

[Renz and Nebel, 1999] Jochen Renz and Bernhard Nebel. On the Complexity of Qualitative Spatial Reasoning: A Maximal Tractable Fragment of the Region Connection Calculus. *AIJ*, 108:69–123, 1999.

[Renz and Nebel, 2001] Jochen Renz and Bernhard Nebel. Efficient Methods for Qualitative Spatial Reasoning. *JAIR*, 15:289–318, 2001.

[Renz and Nebel, 2007] Jochen Renz and Bernhard Nebel. Qualitative Spatial Reasoning Using Constraint Calculi. In *Handbook of Spatial Logics*, pages 161–215. 2007.

[Renz, 1999] Jochen Renz. Maximal Tractable Fragments of the Region Connection Calculus: A Complete Analysis. In *IJCAI*, 1999.

[Renz, 2002a] Jochen Renz. A Canonical Model of the Region Connection Calculus. *JANCL*, 12:469–494, 2002.

[Renz, 2002b] Jochen Renz. *Qualitative Spatial Reasoning with Topological Information*. Springer-Verlag, 2002.

[Renz, 2007] Jochen Renz. Qualitative Spatial and Temporal Reasoning: Efficient Algorithms for Everyone. In *IJCAI*, 2007.

[Röhrig, 1993] Ralf Röhrig. *CYCORD: a theory of qualitative spatial reasoning*. Labor für Künstliche Intelligenz Hamburg: LKI-M. 1993.

[Röhrig, 1994] Ralf Röhrig. A theory for qualitative spatial reasoning based on order relations. In *AAAI*, 1994.

[Rose, 1972] D. J. Rose. A graph-theoretic study of the numerical solution of sparse positive definite systems of linear equations. In *Graph Theory and Computing*, pages 183–217. Academic Press, 1972.

[Rosenkrantz *et al.*, 1977] Daniel J. Rosenkrantz, Richard Edwin Stearns, and Philip M. Lewis II. An Analysis of Several Heuristics for the Traveling Salesman Problem. *SIAM J. Comput.*, 6:563–581, 1977.

[Russell and Norvig, 2010] Stuart J. Russell and Peter Norvig. *Artificial Intelligence - A Modern Approach (3. internat. ed.)*. Pearson Education, 2010.

[Saade, 2008] Mahmoud Saade. *Étude du raisonnement temporel basé sur la résolution de contraintes*. PhD thesis, Université d'Artois, 2008.

[Salhi and Sioutis, 2015] Yakoub Salhi and Michael Sioutis. A Resolution Method for Modal Logic S5. In *GCAI*, pages 252–262, 2015.

[Santos and Moreira, 2009] Maribel Yasmina Santos and Adriano Moreira. Conceptual neighborhood graphs for topological spatial relations. In *WCE*, 2009.

[Schlieder, 1993] Christoph Schlieder. Representing visible locations for qualitative navigation. In *Qualitative Reasoning and Decision Technologies*, 1993.

[Schlieder, 1995] Christoph Schlieder. Reasoning About Ordering. In *COSIT*, 1995.

[Schlieder, 1996] Christoph Schlieder. Qualitative Shape Representation. In *GISDATA Specialist Meeting on Geographical Objects with Undetermined Boundaries*, 1996.

[Schrijver, 1986] Alexander Schrijver. *Theory of Linear and Integer Programming*. John Wiley & Sons, Inc., 1986.

[Sioutis and Condotta, 2014a] Michael Sioutis and Jean-François Condotta. Incrementally Building Partially Path Consistent Qualitative Constraint Networks. In *AIMSA*, 2014.

[Sioutis and Condotta, 2014b] Michael Sioutis and Jean-Franćois Condotta. Tackling Large Qualitative Spatial Networks of Scale-Free-Like Structure. In *SETN*, 2014.

[Sioutis and Condotta, 2014c] Michael Sioutis and Jean-François Condotta. Vertex Incremental Path Consistency for Qualitative Constraint Networks. In *SETN*, 2014.

[Sioutis and Koubarakis, 2012] Michael Sioutis and Manolis Koubarakis. Consistency of Chordal RCC-8 Networks. In *ICTAI*, 2012.

[Sioutis *et al.*, 2014] Michael Sioutis, Jean-François Condotta, Yakoub Salhi, and Bertrand Mazure. A Qualitative Spatio-Temporal Framework Based on Point Algebra. In *AIMSA*, 2014.

[Sioutis *et al.*, 2015a] Michael Sioutis, Jean-François Condotta, Yakoub Salhi, and Bertrand Mazure. A Tableau Method for Generalized Qualitative Spatio-Temporal Reasoning. In *STeDy@IJCAI*, 2015.

[Sioutis *et al.*, 2015b] Michael Sioutis, Jean-François Condotta, Yakoub Salhi, and Bertrand Mazure. Generalized Qualitative Spatio-Temporal Reasoning: Complexity and Tableau Method. In *TABLEAUX*, 2015.

[Sioutis *et al.*, 2015c] Michael Sioutis, Jean-François Condotta, Yakoub Salhi, and Bertrand Mazure. The Implication of Patchwork and Compactness in Qualitative Spatio-Temporal Reasoning. In *STeDy@IJCAI*, 2015.

[Sioutis *et al.*, 2015d] Michael Sioutis, Jean-François Condotta, Yakoub Salhi, Bertrand Mazure, and David A. Randell. On Ordering Spatio-Temporal Sequences to meet Transition Constraints. In *STeDy@IJCAI*, 2015.

[Sioutis *et al.*, 2015e] Michael Sioutis, Jean-François Condotta, Yakoub Salhi, Bertrand Mazure, and David A. Randell. Ordering Spatio-Temporal Sequences to Meet Transition Constraints: Complexity and Framework. In *AIAI*, 2015.

[Sioutis *et al.*, 2015f] Michael Sioutis, Sanjiang Li, and Jean-François Condotta. Efficiently Characterizing Non-Redundant Constraints in Large Real World Qualitative Spatial Networks. In *IJCAI*, 2015.

[Sioutis *et al.*, 2015g] Michael Sioutis, Sanjiang Li, and Jean-François Condotta. On Redundancy in Linked Geospatial Data. In *LDQ@ESWC*, 2015.

[Sioutis *et al.*, 2015h] Michael Sioutis, Yakoub Salhi, and Jean-François Condotta. A Simple Decomposition Scheme For Large Real World Qualitative Constraint Networks. In *FLAIRS*, 2015.

[Sioutis *et al.*, 2015i] Michael Sioutis, Yakoub Salhi, and Jean-François Condotta. On the Use and Effect of Graph Decomposition in Qualitative Spatial and Temporal Reasoning. In *SAC*, 2015.

[Sioutis *et al.*, 2016a] Michael Sioutis, Jean-François Condotta, and Manolis Koubarakis. An Efficient Approach for Tackling Large Real World Qualitative Spatial Networks. *IJAIT*, 25:1–33, 2016.

[Sioutis *et al.*, 2016b] Michael Sioutis, Zhiguo Long, and Sanjiang Li. Efficiently Reasoning about Qualitative Constraints through Variable Elimination. In *SETN*, 2016.

[Sioutis *et al.*, 2016c] Michael Sioutis, Yakoub Salhi, and Jean-François Condotta. Studying the Use and Effect of Graph Decomposition in Qualitative Spatial and Temporal Reasoning. In *Knowledge Eng. Review*, 2016. In press.

[Sioutis, 2014] Michael Sioutis. Triangulation versus Graph Partitioning for Tackling Large Real World Qualitative Spatial Networks. In *ICTAI*, 2014.

[Sistla and Clarke, 1985] A. Prasad Sistla and Edmund M. Clarke. The Complexity of Propositional Linear Temporal Logics. *J. ACM*, 32:733–749, 1985.

[Skiadopoulos and Koubarakis, 2005] Spiros Skiadopoulos and Manolis Koubarakis. On the consistency of cardinal direction constraints. *AIJ*, 163:91–135, 2005.

[Snodgrass, 1987] Richard T. Snodgrass. The Temporal Query Language TQuel. *ACM Trans. Database Syst.*, 12:247–298, 1987.

[Song and Cohen, 1988] Fei Song and Robin Cohen. The Interpretation of Temporal Relations in Narrative. In *IJCAI*, 1988.

[Stergiou and Koubarakis, 1998] Kostas Stergiou and Manolis Koubarakis. Backtracking Algorithms for Disjunctions of Temporal Constraints. In *AAAI*, 1998.

[Stergiou and Koubarakis, 2000] Kostas Stergiou and Manolis Koubarakis. Backtracking algorithms for disjunctions of temporal constraints. *AIJ*, 120:81–117, 2000.

[Steyvers and Tenenbaum, 2005] Mark Steyvers and Joshua B. Tenenbaum. The Large-Scale Structure of Semantic Networks: Statistical Analyses and a Model of Semantic Growth. *Cognitive Science*, 29:41–78, 2005.

[Story and Worboys, 1995] P. A. Story and Michael F. Worboys. A Design Support Environment for Spatio-Temporal Database Applications. In *COSIT*, 1995.

[Tarjan and Yannakakis, 1984] Robert Endre Tarjan and Mihalis Yannakakis. Simple Linear-Time Algorithms to Test Chordality of Graphs, Test Acyclicity of Hypergraphs, and Selectively Reduce Acyclic Hypergraphs. *SIAM J. Comput.*, 13:566–579, 1984.

[Tarski, 1941] Alfred Tarski. On the calculus of relations. *The Journal of Symbolic Logic*, 6:73–89, 1941.

[Tsao-Chen, 1938] Tang Tsao-Chen. Algebraic postulates and a geometric interpretation for the Lewis calculus of strict implication. *Bull. Amer. Math. Soc.*, 44:737–744, 1938.

[van Beek and Cohen, 1990] Peter van Beek and Robin Cohen. Exact and approximate reasoning about temporal relations. *Computational Intelligence*, 6:132–144, 1990.

[van Beek and Dechter, 1995] Peter van Beek and Rina Dechter. On the Minimality and Decomposability of Row-Convex Constraint Networks. *JACM*, 42:543–561, 1995.

[van Beek and Manchak, 1996] Peter van Beek and Dennis W. Manchak. The design and experimental analysis of algorithms for temporal reasoning. *JAIR*, 4:1–18, 1996.

[Van Beek, 1990] Peter Van Beek. *Exact and approximate reasoning about qualitative temporal relations*. PhD thesis, University of Waterloo, 1990.

[van Beek, 1992] Peter van Beek. Reasoning About Qualitative Temporal Information. *AIJ*, 58:297–326, 1992.

[Vardi and Wolper, 1986] Moshe Y. Vardi and Pierre Wolper. An Automata-Theoretic Approach to Automatic Program Verification. In *LICS*, 1986.

[Vilain *et al.*, 1990] Marc Vilain, Henry Kautz, and Peter van Beek. Readings in qualitative reasoning about physical systems. chapter Constraint Propagation Algorithms for Temporal Reasoning: A Revised Report, pages 373–381. Morgan Kaufmann Publishers Inc., 1990.

[Vilain, 1982] Marc B. Vilain. A System for Reasoning About Time. In *AAAI*, 1982.

[Wallgrün, 2012] Jan Oliver Wallgrün. Exploiting qualitative spatial reasoning for topological adjustment of spatial data. In *SIGSPATIAL*, 2012.

[Walsh, 2001] Toby Walsh. Search on High Degree Graphs. In *IJCAI*, 2001.

[Westphal and Hué, 2012] Matthias Westphal and Julien Hué. Nogoods in Qualitative Constraint-Based Reasoning. In *KI*, 2012.

[Westphal and Wölfl, 2009] Matthias Westphal and Stefan Wölfl. Qualitative CSP, Finite CSP, and SAT: Comparing Methods for Qualitative Constraint-based Reasoning. In *IJCAI*, 2009.

[Westphal *et al.*, 2009] Matthias Westphal, Stefan Wölfl, and Zeno Gantner. GQR: A fast solver for binary qualitative constraint networks. In *AAAI Spring Symposium on Benchmarking of Qualitative Spatial and Temporal Reasoning Systems*, 2009.

[Westphal *et al.*, 2010] Matthias Westphal, Stefan Wölfl, and Jason Jingshi Li. Restarts and Nogood Recording in Qualitative Constraint-based Reasoning. In *ECAI*, 2010.

[Westphal *et al.*, 2013] Matthias Westphal, Julien Hué, Stefan Wölfl, and Bernhard Nebel. Transition Constraints: A Study on the Computational Complexity of Qualitative Change. In *IJCAI*, 2013.

[Westphal, 2014] Matthias Westphal. *Qualitative Constraint-based Reasoning: Methods and Applications*. PhD thesis, Albert-Ludwigs-Universität Freiburg, 2014.

[Wolper, 1985] Pierre Wolper. The tableau method for temporal logic: An overview. *Logique et Analyse*, 28:119–136, 1985.

[Wolter and Zakharyaschev, 2000a] Frank Wolter and Michael Zakharyaschev. Spatial Reasoning in RCC-8 with Boolean Region Terms. In *ECAI*, pages 244–250, 2000.

[Wolter and Zakharyaschev, 2000b] Frank Wolter and Michael Zakharyaschev. Spatio-temporal representation and reasoning based on RCC-8. In *KR*, 2000.

[Wolter and Zakharyaschev, 2003] Frank Wolter and Michael Zakharyaschev. Exploring artificial intelligence in the new millennium. chapter Qualitative Spatiotemporal Representation and Reasoning: A Computational Perspective. Morgan Kaufmann Publishers Inc., 2003.

[Yannakakis, 1981] M. Yannakakis. Computing the Minimum Fill-In is NP-Complete. *SIAM J. on Algebraic Discrete Methods*, 2:77–79, 1981.

[Zhang and Marisetti, 2009] Yuanlin Zhang and Satyanarayana Marisetti. Solving connected row convex constraints by variable elimination. *AIJ*, 173:1204–1219, 2009.

[Zhang and Poole, 1994] Nevin Zhang and David Poole. A simple approach to Bayesian network computations. In *AI*, 1994.

[Zhang, 2007] Yuanlin Zhang. Fast Algorithm for Connected Row Convex Constraints. In *IJCAI*, 2007.

Abstract

Qualitative Spatial and Temporal Reasoning is a major field of study in Artificial Intelligence and, particularly, in Knowledge Representation, which deals with the fundamental cognitive concepts of space and time in an abstract manner.

In our thesis, we focus on qualitative constraint-based spatial and temporal formalisms and make contributions to several aspects. In particular, given a knowledge base of qualitative spatial or temporal information, we define novel local consistency conditions and related techniques to efficiently solve the fundamental reasoning problems that are associated with such knowledge bases. These reasoning problems consist of the *satisfiability problem*, which is the problem of deciding whether there exists a quantitative interpretation of all the entities of a knowledge base such that all of its qualitative relations are satisfied by that interpretation, the *minimal labeling problem*, which is the problem of determining all the atoms for each of the qualitative relations of a knowledge base that participate in at least one of its solutions, and the *redundancy problem*, which is the problem of obtaining all the non-redundant qualitative relations of a knowledge base. Further, we enrich the field of spatio-temporal formalisms that combine space and time in an interrelated manner by making contributions with respect to a qualitative spatio-temporal logic that results by combining the propositional temporal logic (PTL) with a qualitative spatial constraint language, and by investigating the task of ordering a temporal sequence of qualitative spatial configurations to meet certain transition constraints.

Keywords: Spatial and temporal reasoning, qualitative constraints, satisfiability problem, minimal labeling problem, redundancy problem, spatio-temporal logic

Résumé

Le raisonnement spatial et temporel qualitatif est un domaine principal d'études de l'intelligence artificielle et, en particulier, du domaine de la représentation des connaissances, qui traite des concepts cognitifs fondamentaux de l'espace et du temps de manière abstraite.

Dans notre thèse, nous nous focalisons sur les formalismes du domaine du raisonnement spatial et temporel qualitatif représentant les informations par des contraintes et apportons des contributions sur plusieurs aspects. En particulier, étant donnée des bases de connaissances d'informations qualitatives sur l'espace ou le temps, nous définissons des nouvelles conditions de consistance locale et des techniques associées afin de résoudre efficacement les problèmes fondamentaux se posant. Nous traitons notamment du *problème de la satisfiabilité* qui est le problème de décider s'il existe une interprétation quantitative de toutes les entités satisfaisant l'ensemble des contraintes qualitatives. Nous considérons également le *problème de l'étiquetage minimal* qui consiste à déterminer pour toutes les contraintes qualitatives les relations de base participant à au moins une solution ainsi que le *problème de redondance* consistant à déterminer les contraintes qualitatives non redondantes. En outre, nous enrichissons le domaine des formalismes spatio-temporels par des contributions concernant une logique spatio-temporelle combinant la logique temporelle propositionnelle (PTL) avec un langage de contraintes qualitatives spatiales et une étude de la problématique consistant à gérer une séquence temporelle de configurations spatiales qualitatives devant satisfaire des contraintes de transition.

Mots-clés: raisonnement spatial et temporel, contraintes qualitatives, problème de satisfiabilité, problème de l'étiquetage minimal, problème de redondance, logique spatio-temporelle