# Qualitative Spatial Configuration Queries

## Towards Next Generation Access Methods for gis

Paolo Fogliaroni

Dissertation
zur Erlangung des Grades eines Doktors der
Ingenieurwissenschaften

— Dr.-Ing. —

Vorgelegt im Fachbereich 3 (Mathematik & Informatik)
der Universität Bremen
26 Juli 2012

Datum des Promotionskolloquiums:

Gutachter: Prof. Christian Freksa, Ph.D. (Universität Bremen)

Dieter Pfoser, Ph.D. (Research Center "ATHENA", Institute for the Management of Information Systems)

# Abstract

For a long time survey, management, and provision of geographic information in Geographic Information Systems (GIS) have mainly had an authoritative nature. Today the trend is changing and such an authoritative geographic information source is now accompanied by a public and freely available one which is usually referred to as Volunteered Geographic Information (VGI). Actually, the term VGI does not refer only to the mere geographic information, but, more generally, to the whole process which assumes the engagement of volunteers to collect and maintain such information in freely accessible GIS.

The quick spread of VGI gives new relevance to a well-known challenge: developing new methods and techniques to ease down the interaction between users and GIS. Indeed, in spite of continuous improvements, GIS mainly provide interfaces tailored for experts, denying the casual user—usually a non-expert—the possibility to access VGI information. One main obstacle resides in the different ways GIS and humans deal with spatial information: GIS mainly encode spatial information in a *quantitative* format, whereas human beings typically prefer a *qualitative* and *relational* approach. For example, we use expressions like "the lake is to the *right-hand side* of the wood" or "is there a supermarket *close* to the university?" which qualitatively locate a spatial entity with respect to another.

Nowadays, such a gap in representation has to be plugged by the user, who has to learn about the system structure and to encode his requests in a form suitable to the system. Contrarily, enabling GIS to explicitly deal with qualitative spatial information allows for shifting the translation effort to the system side. Thus, to facilitate the interaction with human beings, GIS have to be enhanced with tools for efficiently handling qualitative spatial information.

The work presented in this thesis addresses the problem of enabling Qualitative Spatial Configuration Queries (QSCQs) in GIS. A QSCQ is a spatial database query which allows for an automatic mapping of spatial descriptions produced by humans: A user naturally expresses his request of spatial information by drawing a sketch map or producing a verbal description. The qualitative information conveyed by such descriptions is automatically extracted and encoded into a QSCQ.

The focus of this work is on two main challenges: First, the development of a framework that allows for managing in a spatial database the variety of spatial aspects that might be enclosed in a spatial description produced by a human. Second, the conception of Qualitative Spatial Access Methods (QSAMs): algorithms and data structures tailored for *efficiently* solving QSCQs. The main objective of a QSAM is that of countering the exponential explosion in terms of storage space—occurring when switching from a quantitative to a qualitative spatial representation—while keeping query response time acceptable.

# Zusammenfassung

Seit langem wird das Management und die Bereitstellung von Geoinformationen in Geographic Information Systems (GIS) von wenigen zentralen autoritativen Stellen bearbeitet. Im Informationszeitalter werden jedoch frei verfügbare öffentliche Informationsquellen, sogenannte Volunteered Geographic Information (VGI), immer wichtiger. Hierbei bezeichnet der Ausdruck VGI nicht nur die eigentliche Geoinformation, sondern den Gesamtprozess, in welchem durch das Mitwirken von Freiwilligen Geoinformationen in öffentlichen GIS gesammelt und gewartet werden.

Die schnelle Verbreitung von VGI resultiert in einer wohlbekannten Herausforderung: Das Entwickeln von Methoden und Techniken um die Interaktion zwischen Benutzern und GIS zu vereinfachen. Trotz kontinulierlicher Verbesserungen stellen GIS bis heute überwiegend Expertenschnittstellen bereit, sodass der Gelegenheitsnutzer kaum eine Möglichkeit hat, an Informationen aus VGI zu gelangen. Eine besondere Hürde besteht darin, dass Menschen und GIS grundsätzlich unterschiedlich mit räumlichen Informationen umgehen: GIS codieren räumliche Informationen zumeist auf *quantitative* Art und Weise, während Menschen normalerweise einen *qualitativen* und *relationalen* Ansatz bevorzugen. Beispielsweise benutzen wir Ausdrücke wie "Der See ist *rechts* des Waldes." oder "Gibt es einen Supermarkt *in der Nähe* der Universität?", sodass eine räumliche Entität in Bezug zu einer anderen gesetzt wird.

Heutzutage wird diese Lücke vom Benutzer gefüllt, der die Informationsstruktur des Systems erlernen muss und seine Anfragen in einer Form darstellen muss, die an das System angepasst ist. Um die Benutzerfreundlichkeit zu erhöhen und um Gelegenheitsnutzern Zugang zu GIS zu ermöglichen, muss also ein GIS, welches mit qualitativen räumlichen Informationen arbeitet, diese Übersetzungsarbeit leisten. Das System muss mit Werkzeugen zur effizienten Verarbeitung von qualitativen räumlichen Informationen ausgestattet werden.

Diese Dissertation adressiert das Problem der Behandlung von sogenannten Qualitative Spatial Configuration Queries (QSCQs) in GIS. Eine QSCQ ist eine Datenbankanfrage welche eine automatisierte Übersetzung von menschlichen räumlichen Beschreibungen erlaubt: Ein Benutzer wird seine Anfrage an eine räumliche Datenbank normalerweise in Form einer verbalen Beschreibung oder einer handgezeichneten

Skizze stellen. Die qualitative Information, welche in solchen Beschreibungen enthalten ist, wird automatisch extrahiert und in eine QSCQ übersetzt.

Der Fokus dieser Arbeit gilt vor Allem zwei Herausforderungen: Erstens der Entwicklung eines Rahmenwerks, welches es erlaubt, die Ausprägungen von räumlichen Aspekten in menschlichen Beschreibungen in einer Datenbank zu verwalten. Zweitens der Erstellung von sogenannten Qualitative Spatial Access Methods (QSAMs): Algorithmen und Datenstrukturen, welche darauf zugeschnitten sind, QSCQs *effizient* zu lösen. Stellt man quantitative Daten qualitativ dar, kommt es zu einer exponentiellen Explosion des benötigten Speicherplatzes. Das Hauptziel von QSAM ist es, diesen Mehraufwand zu verhindern oder zu begrenzen, während die Bearbeitungszeit einer Anfrage akzeptabel bleibt.

# Sommario

La raccolta, la gestione e la fornitura di informazioni geografiche nei Geographic Information Systems (GIS) ha avuto per un lungo periodo una natura prettamente autoritaria. Oggi la tendenza sta cambiando e questa sorgente autoritaria di informazioni geografiche è ora accompagnata da una pubblica e liberamente disponibile a cui ci si riferisce, tipicamente, col nome di Volunteered Geographic Information (VGI). In realtà, con il termine VGI non ci si riferisce solo alle mere informazioni geografiche, quanto piuttosto all'intero processo che assume l'impiego di volontari per il collezionamento ed il mantenimento di tali informazioni in GIS liberamente accessibili.

La rapida diffusione del VGI da nuova rilevanza ad una ben nota sfida: sviluppare nuovi metodi e tecniche che facilitino l'interazione tra utenti e GIS. Infatti, malgrado i continui miglioramenti, i GIS continuano a fornire interfacce principalmente disegnate per esperti, negando all'utente casuale—tipicamente un non esperto—la possibilità di accedere alle informazioni VGI. Uno degli ostacoli principali risiede nel diverso modo in cui GIS ed esseri umani gestiscono le informazioni spaziali: mentre i GIS codificano le informazioni spaziali principalmente in un formato *quantitativo*, gli esseri umani preferiscono ricorrere, tipicamente, ad un approccio *relazionale* e *qualitativo*. Per esempio, ricorriamo ad espressioni quali "il lago si trova *a destra* del bosco" oppure "il supermercato è *vicino* all'università?" che localizzano qualitativamente una entità spaziale rispetto ad un'altra.

Ad oggi, questo divario rappresentativo deve essere colmato dall'utente, il quale deve imparare a conoscere la struttura interna del sistema ed a codificare la sua richiesta in un formato che quest'ultimo possa interpretare. Viceversa, abilitare i GIS al trattamento esplicito delle informazioni spaziali qualitative permetterebbe di far ricadere lo sforzo di traduzione sul sistema piuttosto che sull'utente. Dunque, per facilitare l'interazione con gli esseri umani, i GIS devono essere potenziati con strumenti che permettano una gestione efficiente delle informazioni qualitative spaziali.

Il lavoro presentato in questa tesi affronta il problema di abilitare le Qualitative Spatial Configuration Queries (QSCQs) nei GIS. Una QSCQ

è un tipo di query ad un database spaziale che permette un mapping automatico delle descrizioni spaziali prodotte dagli esseri umani: un utente può esprimere la sua richiesta di informazioni spaziali disegnando delle sketch maps o producendo delle descrizioni verbali. Le informazioni qualitative convogliate da tali descrizioni vengono automaticamente estratte e codificate in una QSCQ.

Questo lavoro si focalizza principalmente sulla risoluzione di due problemi specifici. In primo luogo lo sviluppo di un framework che permetta di gestire in un database la varietà di aspetti spaziali che possono essere racchiusi in una descrizione prodotta da un essere umano. Successivamente l'attenzione è posta sull'ideazione di Qualitative Spatial Access Methods (QSAMs): algoritmi e strutture dati progettati per risolvere *efficientemente* una QSCQ. L'obiettivo principale di un QSAM è quello di contrastare l'esplosione esponenziale in termini di spazio di memorizzazione—a cui si è soggetti quando si passa da una rappresentazione spaziale quantitativa ad una qualitativa—mantenendo, contemporaneamente, un tempo di risposta delle query accettabile.

*a Roberto*

*"Ora insegna agli angeli come si vive*
*e fanne, di quel paradiso,*
*un vero Nirvana."*

# Acknowledgements

I gratefully acknowledge the Deutsche Forschungsgemeinschaft (DFG) which provided me with the financial support necessary to carry out my research under the grant IRTG GRK 1498 Semantic Integration of Geospatial Information.

Writing this dissertation took me considerably longer than I was expecting. I have to admit that I have definitely underestimated the amount of work required to write a scientific text of this size in a language that is not my mother tongue. Luckily, many have supported me during these months and helped me every time I was discouraged and tempted to admit a defeat. Well, it is time to thank all such people without whom this dissertation, probably, would have never come to an end.

My first thought and most grateful thanks go to my supervisor, Prof. Christian Freksa. Thank you, Christian, for having given me the opportunity to work in your wonderful research group and for allowing me enough freedom to develop and pursue my ideas. Your comments have been enlightening and they have often broadened my mind and gifted me with new perspectives on my work.

A special thank to the two post-docs who led me in the development of this work, Dr. Jan Oliver Wallgrün and Dr. Falko Schmid. Thank to Jan for the countless advices which helped me out in steering my research to the right direction, and thank to Falko who, with his stronger pragmatism, allowed me to come to a realization of my ideas in a reasonable amount of time.

I also owe a sincere thank to all my colleagues in IRTG, and especially to the Bremen sub-group which I have been daily in contact with. Thanks to Manfred Eppe, Torben Gerkensmeyer, Giorgio De Felice and Jae Hee Lee for all the informal discussions and for having been listening to my fancy ideas. In particular, thanks to Giorgio, a long-time friend, who supported me during the dark periods and helped me to find a way out of the countless pitfalls which I got stuck into.

I want to thank all my colleagues in the Cognitive Systems group for having provided me with a challenging and stimulant environment where to grow my ideas. A special thank goes to Dr. Mehul Bhatt and to Dr. Diedrich Wolter. Mehul, you were always available for advising and encouraging me: you taught me to trust more in my abilities.

Diedrich, your biting comments have been rumbling in my mind during the whole writing period: they helped me in being accurate in the definitions and picky in the descriptions.

Thanks also to all of them who accepted to revise and to proofread part of my thesis: Dr. Mehul Bhatt, Dr. Diedrich Wolter, Dr. Falko Schmid, Dr. Thomas Barkowsky, Dr. Lutz Frommberger and Dr. Ana-Maria Olteteanu. And thanks again to Manfred Eppe and Torben Gerkensmeyer for translating the abstract in German.

I cannot skip the administrative and technical staff: thanks to Ingrid, Gracia, Eva and Susanne for your help in deciphering and browsing through German bureaucracy as well as for your always happy goodmornings and your concerns about my health during the worst phases of writing. You helped me in starting each day with the right step. Alexander, I owe you more than half of this thesis: thank for saving me from disk failure and for having never left me without the necessary technical means.

This dissertation would have never seen the daylight without the constant support of my family which kept believing in me every single moment of my life. Thank to my mother, Maddalena, and to my brother, Andrea. This work is to reciprocate your endless support and understanding, especially during those dark periods when I become intractable.

A special thank goes to my girlfriend Michela, thank you for having been at my side during this enterprise, for having listened to me spelling out my thoughts, and for having read and commented on my writing style.

Thanks to my lifetime friends, who, similarly to my family, gave me the strength to not give up. Thanks to Andrea (Nano), Daniele (Sbrocceaux), Matteo (Naseaux), Giuseppe (Peppino), Jihad (Jax), Andrea (Gattaccio), Matteo (Ariel), Adriano (Donciò), Alessandro (Tobbino), Stefano (Senzacò), Samira (Mirasà) and Roberto (Lo Zio). Robby, this work is dedicated to your memory which will never die out of my mind. Thank you for being my friend, and having taught me the "secrets of life".

Yet, thanks to the friends with whom I spent large part of my free time here in Bremen, thanks to Sergio, Mitja, Lasse, Michael, Andach and Rami.

A final thank to all the people I met in these years, it was worth it to spending time together and sharing opinions on the most disparate topics. You are too many to fit in such a small space, but all your names reside in my heart.

# Contents

# List of Figures

# List of Tables

# List of Algorithms

# Notation and Symbols

| | |
|---|---|
| $2^S$ | Power set of a set $S$, |
| $\mathsf{a}$ | Arity of a generic qualitative relation or calculus, |
| $\mathcal{A}$ | Arc set of a hypergraph $\mathcal{H}$, |
| $A$ | Arity pool: set of arities of the calculi in $\boldsymbol{\mathcal{P}}$, |
| $\mathsf{b}$ | Cardinality of $\mathcal{B}$, |
| $\boldsymbol{\mathcal{B}}$ | Relation pool: set of base relations of the calculi in $\boldsymbol{\mathcal{P}}$, |
| $\mathsf{B}$ | Cardinality of $\boldsymbol{\mathcal{B}}$, |
| $\mathcal{B}$ | Set of base relations of a qualitative spatial calculus $\mathcal{M}$, |
| $\mathsf{c}$ | Number of spatial constraints in $\mathcal{Q}$, |
| $\mathcal{C}$ | Set of clusters associated with a spatial dataset $O$, |
| $\mathcal{D}$ | Domain of a qualitative relation, |
| $\mathcal{H}$ | A hierarchical spatial clustering index, |
| $\mathcal{H}$ | A hypergraph, |
| $\mathcal{I}$ | A spatial clustering index, |
| $\mathcal{IG}$ | An inference graph, |
| $\mathcal{IK}$ | An inference kernel, |
| $\mathcal{IP}$ | An inference path, |
| $\mathcal{M}$ | A qualitative spatial calculus, |
| $\mathsf{N}$ | Cardinality of $O$, |
| $\mathcal{N}$ | Node set of a hypergraph $\mathcal{H}$, |
| $O$ | Set of spatial objects / Spatial dataset, |

| | |
|---|---|
| $o$ | A spatial object, |
| $\boldsymbol{\mathcal{P}}$ | Pool of available qualitative spatial calculi, |
| $\mathsf{p}$ | Cardinality of $\boldsymbol{\mathcal{P}}$, |
| $\mathcal{Q}$ | A qualitative spatial configuration query, |
| R | Generic qualitative spatial relation, |
| $\mathcal{R}_{\mathcal{M}}(O)$ | Set of relations from $\mathcal{M}$ holding over a spatial set $O$, |
| RT | A relation table, |
| $t$ | A tile, |
| $T$ | Tileset associated with a spatial dataset $O$, |
| TT | A template table, |
| $\mathsf{v}$ | Number of spatial variables in $\mathcal{Q}$, |
| $X$ | Set of spatial variables in $\mathcal{Q}$, |
| $x$ | A spatial variable, |
| $Z$ | Generic acceptance zone qualitative spatial relation, |
| $\xi$ | A spatial constraint, |
| $\Xi$ | Set of spatial constraints in $\mathcal{Q}$, |
| $\Sigma(\mathcal{Q})$ | Solution of $\mathcal{Q}$, |
| $\tau$ | Sequence of tiles, |
| $\chi$ | Sequence of spatial variables, |
| $\omega$ | Sequence of spatial objects, |

# Acronyms

| | |
|---|---|
| 9-IM | 9-Intersection Model |
| AI | Artificial Intelligence |
| CDC | Cardinal Direction Calculus |
| CSP | Constraint Satisfaction Problem |
| DBMS | Database Management System |
| F-QSAM | Functional QSAM |
| FoR | frame of reference |
| GIS | Geographic Information System |
| GUI | Graphical User Interface |
| JEPD | jointly exhaustive and pairwise disjoint |
| JSP | Joint Satisfaction Problem |
| LUT | Lookup table |
| MBR | Minimum Bounding Rectangle |
| OGC | OpenGIS Consortium |
| PQBE | Pictorial Query-by-Example |
| QBE | Query-by-Example |
| QBS | Query-by-Sketch |
| QCN | Qualitative Constraint Network |
| QL | Query Language |
| QR | Qualitative Reasoning |
| QSAM | Qualitative Spatial Access Method |

| | |
|---|---|
| QSCQ | Qualitative Spatial Configuration Query |
| QSL-QSAM | Qualitative Storage Layer QSAM |
| QSR | Qualitative Spatial Representation and Reasoning |
| QSR-QSAM | QSR-based QSAM |
| RCC | Region Connection Calculus |
| RDBMS | Relational Database Management System |
| SAM | Spatial Access Method |
| SC-QSAM | Spatial Clustering QSAM |
| SQL | Structured Query Language |
| VGI | Volunteered Geographic Information |

Chapter 1

# Introduction

The perception of space serves a fundamental role in everyday life. For this reason we have always tried to *represent* and *store* spatial information to the best of our abilities. Beginning from geographic charts onwards, we investigated and developed continuously more advanced spatial representation and analysis instruments that help in easing down the process of making space-related decisions. For this purpose, today, we use Geographic Information Systems (GIS): complex computerized systems providing tools to store, manipulate, and analyze (geo)spatial data.

For a long time survey, management, and provision of geographic information in GIS have mainly had an authoritative nature and the access to such information has been restricted to expert users. Today, thanks to the drop in the cost of survey instruments (mainly GPS devices) and to the spread of information integration and sharing tools characteristic of the Web 2.0, the trend is changing. The "old" authoritative geographic information source is now accompanied by a "new", public and freely available one. Recently, Goodchild (2007) coined for it the term Volunteered Geographic Information (VGI): a particular form of user-generated content that assumes the active engagement of volunteers to collect and provide spatial data to be used in GIS.

One main goal of VGI is to make geographic information freely available and usable. Probably the best known form of VGI is represented by web-based projects like OpenStreetMap[1], which aims at collaboratively producing a free editable map of the world, and Wikimapia[2], which furnishes spatial data that users are allowed to tag with textual information. Such projects provide web interfaces to let the users interact with the spatial information stored in an underlying GIS.

[1] http://www.openstreetmap.org/
[2] http://wikimapia.org/

## 1.1 Motivation

Albeit Web 2.0's technologies allowed for bringing GIS within general public's reach, the expertise required to interact with them drastically reduce the number of potential spatial data contributors and consumers. Indeed, in spite of continuous improvements, GIS mainly provide interfaces tailored for experts, denying the casual user—usually a non-expert—the possibility to fully exploit their potentials. The challenge of developing new methods and techniques to interact with GIS in a more intuitive and natural way has been already tackled in the past (cf. Egenhofer, 1996, for example). Today, thanks to VGI, such a challenge gained new relevance (cf. Pfoser, 2011, for example).

GIS allow for performing a variety of spatial data manipulation and presentation for the most disparate tasks going from simple map visualization to path planning, to land usage examination, and to economical or criminal statistics. Nonetheless, most of these operations require a level of expertise that the casual user does not possess. Accordingly, general public can exploit GIS capabilities mainly by means of some web interfaces of the kind provided by OpenStreetMap or Wikimapia. According to Yao (1999), such interfaces restrict geographic data consumers to access spatial data in mainly two ways. (*i*) Location finding: detects a specific location and shows a map of the surrounding area; (*ii*) Routing: finds the shortest route between two locations. In both cases the user is required to specify the location(s) he is interested in either providing the specific name, the exact address or even the geographic coordinates.

Although such data access methods come in handy in many occasions, there is an important type of access that today is largely unaccounted for. It concerns cases in which we want to find a location on the basis of a set of spatial constraints that we specify. In such a case, most of the times we do not know the location a priori, hence we cannot specify an address or a name. From a certain perspective this kind of requests complements the previous ones in that the input consists somehow of the spatial description of an entity and the expected output is the entity's address, name, or geo-coordinates. Such a kind of requests can occur in everyday life demands.

**Example 1.1** (Apartment searching) **-** We are looking for an apartment to rent in the city of Bremen, Germany. We obviously do not know the address in advance, but we have a certain number of spatial constraints in our mind that the place has to satisfy. We want the Apartment to be (*i*) located *in between* the main station and the university, (*ii*) in a walking distance from—i.e. *close to*—a Supermarket and (*iii*) a Tram- or a Bus- Stop. Lastly, (*iv*) we may want a Park to be *visible* from the Apartment. Figure 1.1 depicts the part of the OpenStreetMap dataset of Bremen relevant for the request with icons showing entities of interest. The solution is shown in Figure 1.2 that highlights with different colors the three entity configurations fulfilling the requirements. Note that the bottom-right Apartment is part of two solutions (magenta and blue) as two different Parks are visible from it. Blurred icons indicate objects that do not fulfill the requirements, like, for instance, the Apartments on the left side of the map from which no Parks are visible.

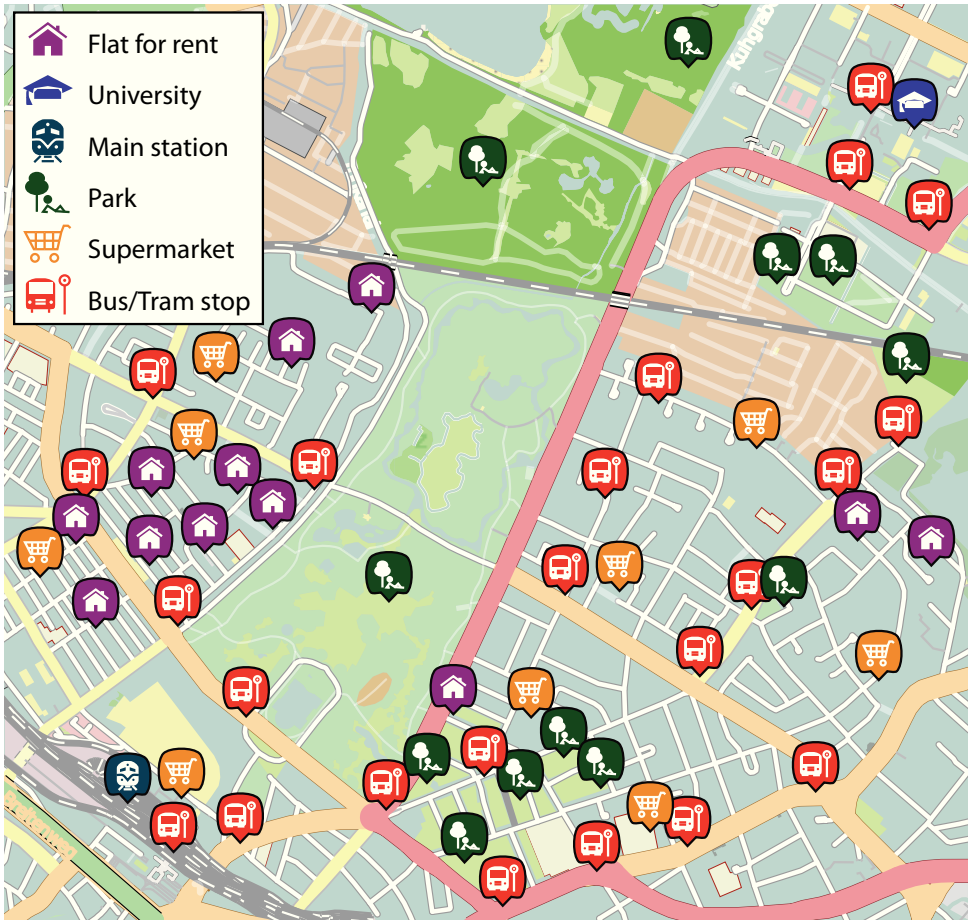



**Figure 1.1:** Partial visualization of the OpenStreetMap dataset of the city of Bremen (Germany) with icons showing some exemplary points of interest.

Basically, solving requests of this kind consists in identifying groups of entities arranged according to some spatial preferences which human beings naturally express in terms of qualitative spatial prepositions—e.g. *in between*, *close to*, and *visible from*. Since these prepositions embody spatial relations among spatial entities, we shall refer to such requests as *qualitative spatial relation queries*.

Qualitative spatial relation queries can play a fundamental role in many decision making scenarios concerned with spatial arrangement which may occur in situations ranging from house finding to urban planning and to strategic positioning of emergency camps in disaster management. Moreover, since they naturally encode spatial descriptions produced by humans, their enablement in GIS makes a breakthrough in the realization of new human–computer interaction techniques. For example, the development of web interfaces that allow for posing such queries can enlarge the public geographic service pool, increasing the amount of geographic data consumers.

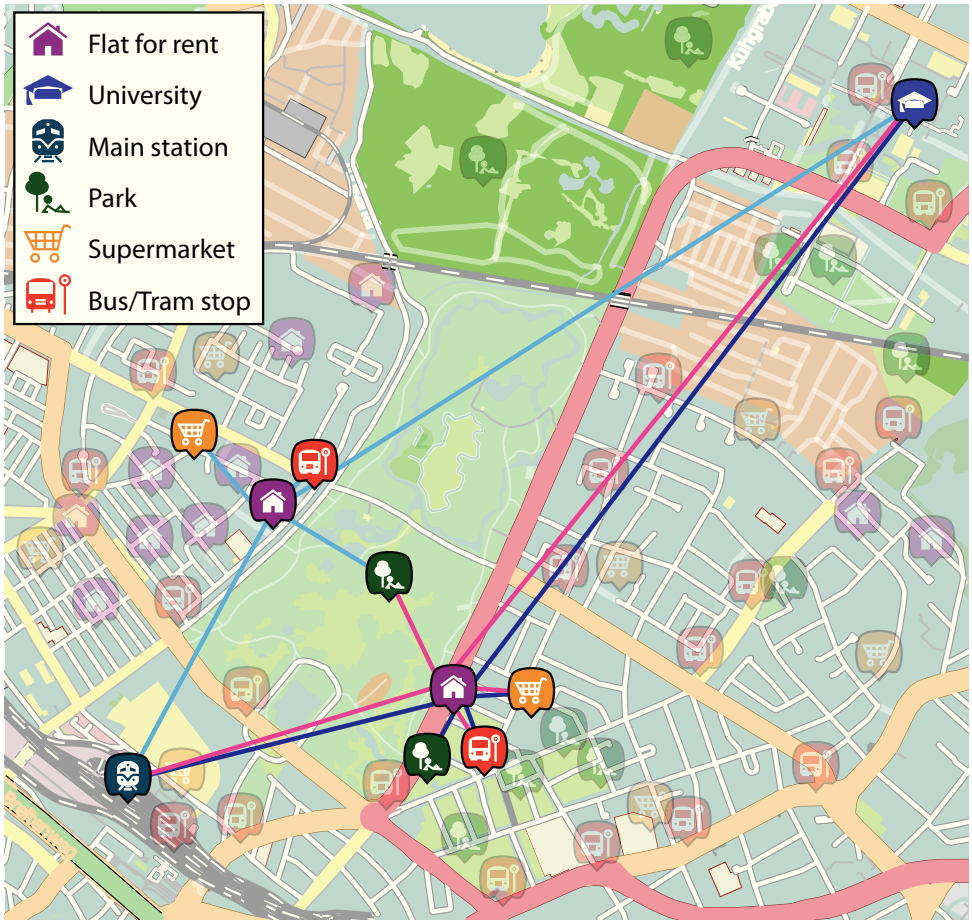



**Figure 1.2:** Configuration solutions for the apartment-search example. The bottom-right flat is part of two different solutions.

## 1.2 Qualitative Spatial Relation Queries

This work focuses on the development of methods and techniques to *efficiently* solve spatial queries for retrieving from a GIS *all* the sets of objects arranged in a specific configuration given in qualitative terms. The interpretation of spatial descriptions into a set of qualitative relations falls outside the scope of this work, however an analysis of their main characteristics is necessary to detect key requirements that have to be addressed.

### 1.2.1 Natural Spatial Descriptions

With the expression *natural spatial description* we shall intend, in the scope of this thesis, those spatial descriptions produced by a human being when communicating spatial knowledge to another human being. This kind of spatial descriptions are natural in the sense that they can be generated and interpreted effortlessly by

the individuals involved in the communication process without the need to resort to further descriptive means than the description itself.

Natural spatial descriptions come mainly in two forms: verbal or pictorial. In verbal forms a spatial entity is typically described using expressions that qualitatively locate it with respect to one or more other objects in the scene, e.g. "The cinema is *in between* the university and the main station, *close to* the park". Occasionally (e.g. when providing directions to a place) people likes to resort to pictorial descriptions, commonly known as sketches or sketch maps, as they provide a much more compact representation of spatial information. Sketch maps look similar to real maps in that they report spatial entities in a geometric format. However, metric information is usually not intended to be reported in sketches which, instead, correctly convey information about certain qualitative relations like relative position, ordering, and connectedness. Qualitative relations, thus, are reliable pieces of information that can be extracted from both verbal and pictorial spatial descriptions. As such, qualitative spatial relations are a key element that has to be considered in the development of solving techniques for querying via natural spatial descriptions.

A further important observation to consider when handling natural spatial descriptions is that they usually convey information regarding several aspects of space. For example, the sentence "The cinema is *in between* the university and the main station, *close to* the park" carries information about relative position and distance. We can safely assume that for every domain/task there are some essential spatial aspects to take into account, while the others are usually irrelevant. For instance, when giving directions to a place in a city, visibility can play an important role as the utilization of visual landmarks can allow for producing clearer instructions; similarly, relative directions might be preferable over cardinal ones. Contrarily, in a geographic description context in which one wants to explain where the city of Bremen is located within Germany, cardinal directions play a significant role, while speaking about visual landmarks is pointless.

Heterogeneous composition of spatial descriptions and context–dependent diversity of relevant spatial aspects are key issues to be considered to develop techniques that can realistically handle natural spatial descriptions. Accordingly, this work does not aim at treating selected aspects of space but, rather, pursues general and extensible solutions that allow for the integration of multiple qualitative spatial aspects.

### 1.2.2 Qualitative Spatial Configuration Queries

Usually a GIS contains information about millions of real-world spatial entities which are represented as geometric objects of different types, i.e. points, lines, polygons, or more complex combinations of these basic types. For example, at the time of writing, the OpenStreetMap dataset of the Niedersachsen region, in Germany, contains nearly 1.300.000 geometric objects.

Solving a qualitative spatial relation query is equivalent to searching for all the sets of objects in the GIS arranged as described in the query. In other words, the qualitative spatial relations expressed in the query have to be matched against those raised by the spatial dataset.

Matching is as much easier as the number of objects uniquely specified in the query increases: If one or more objects are specified by either their unique names, addresses, or geo-coordinates, the query difficulty scales down as the search can be "anchored" on such entities. Similarly, specifying the category of searched objects allows for reducing the number of entities to be searched and, thus, the difficulty of the query. Consequently, qualitative spatial relation queries can be classified according to the degree of specification they provide.

**Example 1.2** (Classes of qualitative spatial relation queries) **-** Answering the query "find all the objects *north of* ID", where ID identifies a certain object in the GIS, consists in checking which cardinal direction relation holds between any object $o$ in the GIS and ID and retrieving all the objects such that $o$ is *north of* ID.

A more demanding query might be "find all the Supermarkets *north of* a Park", where Supermarket and Park are categories of spatial entities which the geometric object in the GIS are tagged with. Answering this query requires to check all the object pairs Supermarket-Park in order to identify and retrieve those satisfying the expressed request.

This work focuses on the hardest class of qualitative spatial relation queries, namely that in which neither searched objects are fixed nor any kind of tag or categorization that allows for reducing the search space is given. We shall refer to such queries as Qualitative Spatial Configuration Queries (QSCQs).

**Efficiently Solving QSCQs: Managing the Space-Time Tradeoff** When switching from the geometric objects to the qualitative spatial relations existing over them, the amount of data to be considered is subjected to a combinatorial explosion: Let us assume, for simplicity, we are only interested in cardinal directions, then every pair of objects from a spatial dataset raises one qualitative relation. Hence, the Niedersachsen dataset gives rise to nearly $(1.3 \cdot 10^6)^2 = 1.69 \cdot 10^{12}$ cardinal direction relations. This number is doomed to augment in a realistic case when multiple spatial aspects have to be accounted for. Accordingly, one of the main challenges in solving a QSCQ is that of opportunely managing the access to such an enormous number of qualitative spatial relations.

Qualitative spatial relations are not explicitly stored in a GIS, thus, in order to perform the matching required to solve a QSCQ, they have to be computed from a geographic dataset. Given the typical size of a geographic dataset such an operation might require an amount of time unacceptable in a human-computer interaction scenario: For instance, in the apartment searching scenario reported in Example 1.1 a user would expect to receive an answer to his request within a few seconds whereas computing the qualitative relations might require hours, days or even months according to the size of the geographic dataset under consideration.

A typical approach to handle computation time issues is that of resorting to pre-computation: the qualitative spatial relations are computed and stored beforehand. This approach leads to a tradeoff situation between the query response time and storage space occupied by the qualitative relations, as reported in the graphics in Figure 1.3.

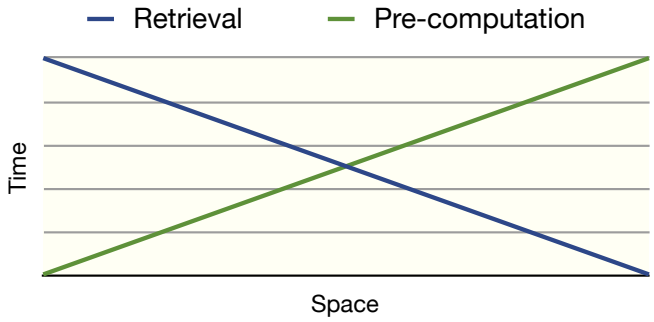



**Figure 1.3:** Retrieval and pre-computation of qualitative spatial relations from a spatial dataset: time as the number of stored relations, hence occupied space, varies.

The higher the number of pre-computed relations (i.e. the occupied storage space) the longer the time to pre-compute them and the shorter the query response time. A basic alternative, located at the extreme left of the graphics, corresponds to null pre-computation time and storage space at the cost of a retrieval time excessively high.

Its direct counterpart locates itself at the extreme right of the graphics and consists in pre-computing all the spatial relations occurring on the spatial dataset. This corresponds to maximal occupied storage space and minimal retrieval time. Moreover, the pre-computation time might be unfeasible in a real case scenario where multiple aspects of space have to be considered: by the time the pre-computation is completed the world is changed as well as its representation in the GIS, calling for a new pre-computation.

A third solution that aims at reaching an optimal space-time tradeoff (the point where the two curves meet) is based on the following observations. The number of qualitative relations generated from a spatial dataset can be drastically reduced by avoiding redundant ones. Indeed, many of the relations raised by a dataset are superfluous as they can be inferred from others.

**Example 1.3** (Superfluous Relations) **-** Let us consider the icons connected via cyan lines in Figure 1.2 and in particular the Park, the Apartment, and the Supermarket ones. The Apartment icon is northwest of the Park icon and the Supermarket is northwest of the Apartment. Investing little reasoning effort it is possible to infer that the Supermarket has necessarily to be northwest of the Park. Thus the latter relation is somehow redundant as it can be inferred from the other two.

A spatial dataset may give rise to many similar instances and it can be represented by as many relations as necessary to infer all the omitted ones. Therefore, qualitative relation reduction strategies can be employed to keep memory consumption within an acceptable range, while search time efficiency can be obtained by combining relation reconstruction strategies with indexing techniques.

Now, it is clear that to efficiently enable QSCQs it is not enough to device "superficial" solutions to put on top of existing systems—like interfaces or access functions. Rather, it is necessary to operate changes deep into the GIS storage layer, developing new algorithms and data structures that, combining spatial indexing techniques and qualitative relation reduction strategies, allow for maintaining spatial and temporal efficiency in the GIS. Such data structures have to be placed aside and designed to work synergistically with the existing ones.

## 1.3 Thesis and Contributions

This work aims at demonstrating the following thesis:

> The synergistic interplay of spatial access methods and reduction strategies is the key to enable and efficiently solve Qualitative Spatial Configuration Queries in Geographic Information Systems.

In particular, the main contributions are:

- A special type of spatial queries, based on the notion of qualitative relation, is defined that embraces and generalizes a variety of queries typically accounted for separately in the literature. Consequentially, this allows for the conception of generalized spatial indexing techniques.
- This work provides a theoretical and practical framework for the integration of an open number of qualitative spatial calculi and spatial access methods. The interplay of Qualitative Spatial Representation and Reasoning (QSR) and indexing techniques is identified as a suitable means to enable qualitative spatial queries in GIS and to retain space-time efficiency.
- Based on the aforementioned framework, a variety of reduction/reconstruction strategies for qualitative spatial relations are presented. They simultaneously exploit the properties of standard spatial indexing techniques and qualitative spatial calculi. An algorithmic realization is given for each of them.
- A novel approach that allows for identifying minimal sets of qualitative spatial relations needed to describe a spatial scene is developed. As such it provides a reduction strategy based on a general reduction theory that exploits properties of reasoning tables provided with qualitative calculi but does not depend on any particular algebraic property.

## 1.4 Outline of the Thesis

The remainder of this thesis is organized as follows: In Chapter 2 different approaches for the representation and management of spatial information are discussed and some fundamental background concepts are introduced.

The problem of enabling Qualitative Spatial Configuration Queries in Geographic Information Systems—and particularly in spatial databases—is detailed in Chapter 3. A set of necessary requirements is defined on top of which is designed a basic resolution method. Moreover, the issue of *dataset qualification* is raised and formalized for which two straightforward—but inefficient—solutions are discussed.

In Chapter 4 a theory of qualitative spatial relations indexing is developed. There, we define the concept of Qualitative Spatial Access Method (QSAM) which can embody the basic resolution methods presented in the previous chapter as well as more advanced indexing techniques. In particular, we introduce a family of QSAMs based on a tile&cluster strategy and a generalized approach based on QSR techniques. Such QSAMs have been conceived in the scope of this work; they are formalized and presented along with an algorithmic realization.

Chapter 5 is devoted to the development of a novel prototypical software framework. It provides an extension for an existing open-source Database Management System (DBMS) and a realization of the theoretical framework traced along the previous chapters.

Such a framework also provides a QSAM development and benchmark environment that has been used to evaluate the presented work. The results of such an evaluation are presented in Chapter 6. Finally, conclusions are drawn in Chapter 7 where also possible extensions and future work are outlined.

Chapter 2

# Representation and Management of Spatial Information

This chapter presents a review of the state–of–the–art on different techniques for the representation and management of spatial information in computer systems and Geographic Information Systems (GIS). Section 2.1 introduces some background concepts on graphs and hypergraphs, with a special focus on directed hypergraphs which are largely used in the next chapters. Section 2.2 narrows down to a specific field of knowledge representation called Qualitative Spatial Representation and Reasoning (QSR) that provides the formal ground and some core techniques which ideas of this thesis have been stemmed from. Finally, in Section 2.3 a review of the state-of-the-art on spatial databases is presented that draws special attention on spatial queries, spatial indexes, and spatial query languages.

## 2.1 Directed Hypergraphs

A graph is a combinatorial structure widely used in mathematics and computer science to represent sets of objects and binary connections over them. The represented objects are abstracted into entities called nodes, whereas interconnections are node pairs called edges. If the edges are *ordered* pairs one speaks of directed graphs and directed edges or, more simply, of digraphs and arcs. If multiple edges (resp. arcs) between the same node pair are admitted one speaks of multigraph (resp. multidigraph). Graphs lend themselves very well to be diagrammatically represented. As shown in Figure 2.1, nodes are represented by circles and edges (resp. arcs) by curves connecting the nodes.

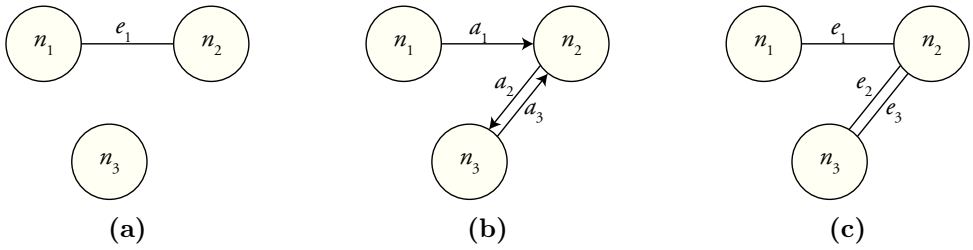



**Figure 2.1:** Three simple graphs. A graph is said undirected (a) if its edges are not oriented, otherwise it is called directed (b). If more instances of the same edge are allowed, one speaks of multigraph (c). The graph in (b) is not a multigraph because $a_1$ and $a_2$ have different orientation; thus, they are not instances of the same arc.

Hypergraphs are a generalization of graphs in which an edge can connect any number of nodes. They have been accurately studied in (Berge, 1976, 1989). In the scope of this work we are mainly concerned with directed hypergraphs, a generalization of directed graphs. More precisely, we are interested in a special type of directed hypergraphs which, in (Gallo *et al.*, 1993), are referred to as B-hypergraphs (or simply B-graphs).

**Definition 2.1** (B-graph) **-** A B-*graph* $\mathcal{H} = (\mathcal{N}, \mathcal{A})$ is an ordered pair, where $\mathcal{N}$ is a set of nodes and $\mathcal{A}$ a set of hyperarcs. A hyperarc $a \in \mathcal{A}$ is itself an ordered pair of the form $(\mathcal{T}, h)$. $\mathcal{T} \subset \mathcal{N}$ is a proper subset of the node set called the *arc tail* and denoted by $\mathcal{T}(a)$. $h \in \mathcal{N} \setminus \mathcal{T}$ is one element of the node set that is not contained in $\mathcal{T}(a)$. It is named the *arc head* and is denoted by $h(a)$.

**Example 2.1 -** Figure 2.2 depicts a simple B-graph $\mathcal{H} = (\mathcal{N}, \mathcal{A})$ consisting of three nodes $\mathcal{N} = \{n_1, n_2, n_3\}$ and one hyperarc $\mathcal{A} = \{a_1\}$ with $\mathcal{T}(a_1) = \{n_1, n_2\}$ and $h(a_1) = n_3$.

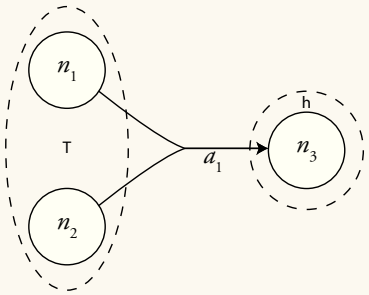



**Figure 2.2:** Simple B-graph.

Thanks to the syntactic structure of their arcs, B-graphs naturally lend themselves to model implication dependencies and have been applied to solve a variety of problems in computer science (cf. Ausiello *et al.*, 2001, for a detailed review). In

(Ausiello *et al.*, 1983), for instance, B-graphs are used to represent functional dependencies[1] in relational databases (cf. Elmasri & Navathe, 2008, for a thorough introduction to databases).

In the next two subsections we will review a series of useful hypergraph concepts that are necessary for a full comprehension of this work. Definitions are taken and adapted from (Ausiello *et al.*, 1983, 1986, 2001; Gallo *et al.*, 1993); for a complete overview refer to these sources.

### 2.1.1 Basic Concepts

Before proceeding with more advanced definitions, let us introduce the following basic concepts. Note that the definitions below also hold for graphs as special instances of hypergraphs.

**Definition 2.2** (Size) **-** The *size* of a hypergraph $\mathcal{H} = (\mathcal{N}, \mathcal{A})$ is defined as $|\mathcal{H}| = |\mathcal{N}| + |\mathcal{A}|$.

**Definition 2.3** (In- and Out-degree) **-** Let $\mathcal{H} = (\mathcal{N}, \mathcal{A})$ be a hypergraph and $n$ one of its nodes. We call *in-degree* (resp. *out-degree*) of $n$ the number of hyperarcs having $n$ as their head (resp. in their tail). We refer to all such hyperarcs as *incoming hyperarcs* (resp. *outgoing hyperarcs*).

**Definition 2.4** (Sub- and Super- hypergraph) **-** Let $\mathcal{H} = (\mathcal{N}, \mathcal{A})$ and $\mathcal{H}' = (\mathcal{N}', \mathcal{A}')$ be two hypergraphs, such that $\mathcal{N}' \subseteq \mathcal{N}$ and $\mathcal{A}' \subseteq \mathcal{A}$. Then, we say that $\mathcal{H}'$ is a *sub-hypergraph of* $\mathcal{H}$ or, alternatively, that $\mathcal{H}$ is a *super-hypergraph of* $\mathcal{H}'$ and denote it by $\mathcal{H}' \subseteq \mathcal{H}$. A sub-hypergraph (resp. super-hypergraph) is said *proper* if the inclusion is strict.

**Definition 2.5** (Arc-induced Subhypergraph) **-** Let $\mathcal{H} = (\mathcal{N}, \mathcal{A})$ be a hypergraph and $\mathcal{A}' \subseteq \mathcal{A}$ a subset of its hyperarcs. Let also $\mathcal{N}' = \cup_{a \in \mathcal{A}'} \mathcal{T}(a) \cup h(a)$ be the union of nodes in the tails and heads of the hyperarcs in $\mathcal{A}'$. The hypergraph $\mathcal{H}' = (\mathcal{N}', \mathcal{A}')$ is called the *subhypergraph of* $\mathcal{H}$ *induced by* $\mathcal{A}'$.

Similarly, it is possible to define a node-induced subhypergraph by considering the hyperarcs ingoing and outgoing a given subset of the hypergraph nodes.

### 2.1.2 Connectedness

In this section we review connectedness in directed hypergraphs which serves a significant role in the scope of this work. Given the complex structure of hypergraphs, we first give an overview of the most relevant concepts for graphs in order to allow for an intuitive understanding. Later we generalize such concepts to hypergraphs and give formal definitions.

Intuitively, we say that an undirected graph is connected if in its diagrammatic representation it is possible to go from each of its nodes to any other moving along

[1] A functional dependency (FD) is a constraint between two sets of attributes in a relation from a database.

its edges. For example, the graph in Figure 2.1(c) is connected whereas the one in Figure 2.1(a) is not. In other words, a graph is connected if there is a path connecting any node pair. More precisely, a path from $n_1$ to $n_l$ is a sequence $n_1, e_1, n_2, e_2, \ldots, e_{l-1}, n_l$, where $n_1, \ldots, n_l$ are nodes of the graph at hand and $e_i = (n_i, n_{i+1})$ is the edge connecting $n_i$ to $n_{i+1}$. Given two nodes $n_1$ and $n_2$ we say that they are connected, or alternatively mutually reachable, if there exists a path going from $n_1$ to $n_2$. Similarly, we say that a graph is connected if any of its nodes is reachable from the others.

The concept of connectedness extends to digraphs. However, due to the fact that edges are oriented (ordered node pairs), the existence of a path from a source node $n_s$ to a target one $n_t$ only guarantees that $n_t$ is reachable from $n_s$ but not vice-versa. Accordingly for digraphs it is possible to specify two different types of connectedness: strong and weak. We say that a digraph is strongly connected if its nodes are reachable from each other through directed paths. We say that it is weakly connected if the associated multigraph, obtained ignoring the arc orientations, is connected. For instance, the digraph in Figure 2.1(b) is weakly connected because its associated multigraph, reported in Figure 2.1(c), is connected.

These concepts can be generalized for directed hypergraphs.

**Definition 2.6** (Hyperpath) **-** Given a hypergraph $\mathcal{H} = (\mathcal{N}, \mathcal{A})$, a hyperpath from a non-empty set of source nodes $S \subset \mathcal{N}$, to a target node $t \in \mathcal{N}$ is a subhypergraph $\Pi_{St} = (\mathcal{N}_\Pi, \mathcal{A}_\Pi)$ of $\mathcal{H}$ such that the following holds:
If $t \in S$, $\mathcal{A}_\Pi = \emptyset$.
Otherwise the $l \geq 1$ hyperarcs of $\Pi_{St}$ can be ordered in a sequence $(a_1, \ldots, a_l)$ such that:

- $\forall a_i \in \mathcal{A}_\Pi$, $\mathcal{T}(a_i) \subseteq S \cup \{h(a_1), \ldots, h(a_{i-1})\}$;
- $t = h(a_l)$;
- No proper subhypergraph of $\Pi_{St}$ is a hyperpath from $S$ to $t$ in $\mathcal{H}$.

The above definition is taken from (Ausiello *et al.*, 2001) and coincides with the definition of B-path given in (Gallo *et al.*, 1993).

**Definition 2.7** (Hypercycle) **-** A hyperpath $\Pi_{St}$ is said a hypercycle if it consists of at least one hyperarc and $t \in S$.

Similarly to digraphs, also in directed hypergraphs it is possible to define a reachability relation between pairs of nodes:

**Definition 2.8** (Reachability) **-** Given a hypergraph $\mathcal{H} = (\mathcal{N}, \mathcal{A})$ and two of its nodes $s, t \in \mathcal{N}$, we say that $t$ is reachable from $s$ in $\mathcal{H}$, and denote it by $s \rightsquigarrow_{\mathcal{H}} t$, if there exists a hyperpath $\Pi_{St}$ from $S = \{s, \cdots\}$ to $t$.

Strong connectedness in hypergraphs directly stems from strong connectedness in digraphs: a hypergraph is said to be strongly connected if for each pair

$(s, t)$ of its nodes, there exists a hyperpath $\Pi_{St}$ and a hyperpath $\Pi_{Ts}$, that is, its nodes are mutually reachable from each other.

In fact, the reachability relation can be used to identify equivalence classes in the node set of non-strongly connected hypergraphs.

**Definition 2.9** (Strongly Connected Component) **-** Given a hypergraph $\mathcal{H} = (\mathcal{N}, \mathcal{A})$, a *strongly connected component* (SCC) is a subhypergraph $C = (\mathcal{N}_C, \mathcal{A}_C)$ of $\mathcal{H}$ such that the following conditions hold simultaneously:

- the nodes of $C$ are pairwise mutually reachable: $s \rightsquigarrow_C t$ and $t \rightsquigarrow_C s$ $\forall s, t \in \mathcal{N}_C$;
- $C$ is maximal: there exists no SCC $C'$ that is a super-hypergraph of $C$.

Also for directed hypergraphs, there exists the concept of weak connectedness. A directed hypergraph is said weakly connected if its associated multi-digraph, obtained replacing each hyperarc with a series of arcs going from each node in the tail to the head, is weakly connected.

**Definition 2.10** (Weakly Connected Component) **-** Given a hypergraph $\mathcal{H} = (\mathcal{N}, \mathcal{A})$, a *weakly connected component* (WCC) is a subhypergraph $C = (\mathcal{N}_C, \mathcal{A}_C)$ of $\mathcal{H}$ such that the following conditions hold simultaneously:

- $C$ is weakly connected;
- $C$ is maximal.

Above concepts are illustrated in the following:

**Example 2.2** (Hyperpaths, Components and Condensation) **-** Let us begin by considering the B-graph in Figure 2.2. There exists a hyperpath $\Pi_{S,n_3}$ going from the node subset $S = \{n_1, n_2\}$ to $n_3$, but neither a hyperpath $\Pi_{\{n_1,\cdots\},n_3}$ nor a hyperpath $\Pi_{\{n_2,\cdots\},n_3}$ exist. Therefore the hypergraph is not strongly connected and its strongly connected components correspond to the single nodes. Figure 2.3(a) shows the multi-digraph associated to it. It is easily verifiable that such a digraph is weakly connected since, by ignoring the arc directions, we have a connected graph. Accordingly the hypergraph in Figure 2.2 is weakly connected.

Now, let us move to Figure 2.3(b). The depicted hypergraph is a B-graph and is also not strongly connected since there is no hyperpath $\Pi_{\{n_2,\cdots\},n_3}$. Conversely, the hypergraph in Figure 2.3(c) is strongly connected: the addition of arc $a_4$ creates the missing hyperpath.

Finally let us take a look at the B-graph in Figure 2.3(d). It is weakly connected, but not strongly. Moreover it is easy to identify three strongly connected components: one consists of the nodes $\{n_1, n_3\}$, the other two correspond to the remaining nodes. The strongly connected components can be used to generate a so-called *condensation*. A condensed (hyper)graph is obtained by substituting each SCC with one node and removing redundant (hyper)arcs. The condensation of the hypergraph in Figure 2.3(d) is depicted in Figure 2.3(e). Note that the arc $a_5$ has been removed since it duplicates $a_6$ in the condensation.

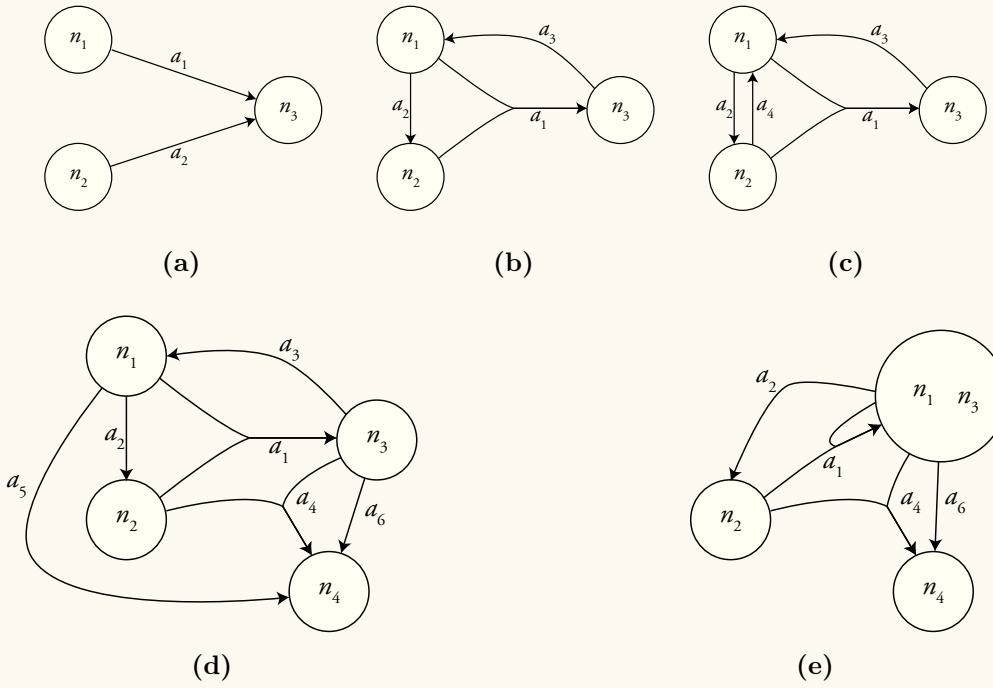



**Figure 2.3:** Hyperpaths, Components, and Condensation in hypergraphs.

## 2.2 Qualitative Spatial Representation and Reasoning

Qualitative Reasoning (QR) (cf. Forbus, 2008, for an introduction) is a subfield of Artificial Intelligence (AI) [1] that aims at abstracting from the continuous, infinitely precise, nature of the world into a discrete representation of it, providing symbolic reasoning techniques.

Forbus, Nielsen & Faltings (1991) point out that one of the main strengths of QR approaches is their minimality in representation. In other words, according to Freksa (1991b), qualitative representations allow for differentiating only as many concepts as necessary for the specific domain and task to achieve. Hence, QR turns particularly helpful when accurate measures (quantitative information) are missing or when resorting to precise calculations is unnecessary or even undesirable.

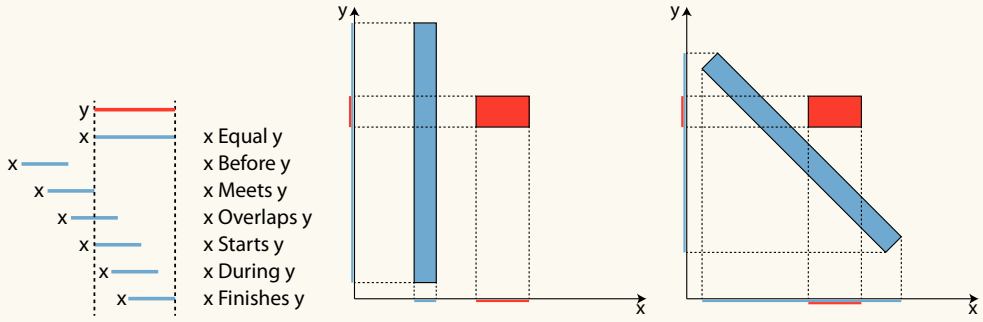
QR is usually concerned with scalar quantities, that is why it falls short when dealing with domains like the spatial one that are intrinsically multi-dimensional.

[1] For the interested reader, (Russell & Norvig, 2003) provides an excellent introduction to AI.

In such domains dimensions are related to each other in such a way that it is impossible to consider them separately. A rehash of mono-dimensional approaches that allows for their application to multi-dimensional domains turns usually into cumbersome adaptations and can potentially lead to wrong results.

**Example 2.3** (Applying mono-dimensional approaches to multi-dimensional domains) **-** A typical example in the literature is about the application of the interval algebra (Allen, 1983) to 2-D space. Allen's algebra was originally developed for reasoning on temporal intervals but it correctly handles any kind of convex intervals in 1-D space. It defines all the possible relations (thirteen) which can occur between two intervals as depicted in Figure 2.4(a).

One possible application to model relations occurring among convex regions in 2-D space consists in considering simultaneously x- and y-Cartesian projections to draw conclusions about the projected objects. Figure 2.4(b) depicts a successful scenario: projections on the x-axis are disconnected leading to the conclusion that the corresponding objects are also disconnected. Such an approach, however, works properly only when dealing with rectangles uniformly aligned with Cartesian axes, as done in (Guesgen, 1989) for instance. Figure 2.4(c), shows a failure scenario where the two rectangles do not overlap although both their projections do.




**(a)** The thirteen possible relations among two convex intervals (the last six are asymmetric and have inverse).

**(b)** Projection onto axes correctly reports on objects' relation.

**(c)** In this case application of Allen's calculus leads to a false result.

**Figure 2.4:** Example adapted from (Cohn & Renz, 2008). The Allen temporal calculus and its application to the spatial domain.

To successfully handle the complexity of multi-dimensional spaces a specialized subfield has emerged from general QR. It is called Qualitative Spatial Representation and Reasoning (QSR) (c.f. Cohn, 1997; Cohn & Hazarika, 2001; Cohn & Renz, 2008; Freksa, 1991b, for an overview) and it is mainly concerned with 2-D and the 3-D Euclidean spaces.

### 2.2.1 On Qualitative Spatial Descriptions

QSR is well suited for synthesizing human way of representing and reasoning about space. Indeed, human beings acquire, reason, and communicate about space everyday and, in particular when it comes to spatial information exchange, they mainly communicate spatial information by resorting to either a verbal (spoken or written) or a pictorial (sketches) description.

Tversky & Lee (1998) investigate route instruction generation and demonstrate that, typically, information reported in sketches is enough to convey the route, whereas verbal descriptions often miss some pieces of information—e.g. the starting or ending point—that are inferred from the context. However, they come to the conclusion that both formats schematize routes in a similar way, in the sense that they rely on the same constituent entities, e.g. landmarks and orientations. Accordingly, Tversky & Lee (1999) argue that verbal and graphical elements map onto one another.

The work of Klippel (2003) moves along the same line: He demonstrates the cognitive adequacy of wayfinding choremes[1] by means of an experiment in which subjects had to produce sketch maps out of verbal descriptions. Klippel's work is also an empirical demonstration that human beings mentally process space by chunking it and grouping large parts under a single concept. That is, humans do not deal with the infinite precision of metric space, but rather abstract from quantitative measures into qualitative categories (like left and right or inside and outside).

In the scope of this work we also assume that verbal and pictorial spatial descriptions rely upon a common structure and we use the general term *natural spatial description* to refer to the qualitative spatial information conveyed by either such description. QSR draws upon the definition of formal structures usually referred to as *qualitative spatial calculi*, which we assume to be valid candidates for modeling and reasoning on such information.

### 2.2.2 Qualitative Modeling: A World of Relations

Modeling is defined as the act of *"devising a representation, especially a mathematical one, of a phenomenon or system"* [2]. A qualitative (spatial) model is, indeed, a well-defined mathematical structure consisting of a potentially infinite set of symbols usually called *relations*. In fact such symbols aim at representing some sort of relationship intervening over a set of entities of interest.

**Definition 2.11** (a-ary Qualitative Relation) **-** Let $\mathcal{D}$ be a potentially infinite domain. An a-*ary qualitative relation* $\mathrm{R} \subseteq \mathcal{D} \times \mathcal{D} \times \cdots \times \mathcal{D} = \mathcal{D}^{\mathsf{a}}$ is a subset of the a-ary Cartesian power

[1] Wayfinding choremes are classifications of direction instructions into a set of directional prototypes.

[2] Definition from Oxford Dictionaries Online (http://www.oxforddictionaries.com/)

of the domain of interest. An element of an $\mathsf{a}$-ary relation is an ordered multiset of $\mathsf{a}$ domain elements called an $\mathsf{a}$-*tuple* .

The domain of interest in the scope of this work is the set of regions embedded in 2-D space. We adopt the typical interpretation of a region as a point-set since it is general enough to embrace lines and points as well—e.g. a point can be seen as a singleton point set. Then, an $\mathsf{a}$-ary spatial relation $\mathrm{R} \subseteq \{(o_1, \ldots, o_\mathsf{a}) \,|o_i \subseteq \mathbb{R}^2, i = 1, \ldots, \mathsf{a}\}$ is a subset of the infinite set consisting of any possible plane region $\mathsf{a}$-tuple.

The typical approach in QSR is to define a finite set of relations that is *jointly exhaustive and pairwise disjoint (*JEPD*)*.

**Definition 2.12** (JEPD set of qualitative relations) **-** A set $\mathcal{B}$ of $\mathsf{a}$-ary qualitative relations is called *jointly exhaustive and pairwise disjoint* if the following conditions hold simultaneously:

- $\mathcal{B}$ covers all the possible domain $\mathsf{a}$-tuples: $\bigcup_{\mathrm{R}_i \in \mathcal{B}} \mathrm{R}_i = \mathcal{D}^\mathsf{a}$;
- any domain $\mathsf{a}$-tuple is contained in one, and only one, relation from $\mathcal{B}$: $\mathrm{R}_i \cap \mathrm{R}_j = \emptyset$ $\forall\, \mathrm{R}_i, \mathrm{R}_j \in \mathcal{B}$ with $i \neq j$.

The relations in a JEPD set $\mathcal{B}$ are usually referred to as *base relations*, whereas those belonging to the powerset $2^\mathcal{B}$ of $\mathcal{B}$ are called *disjunctive relations*. The set of disjunctive relations is obtained by considering all the possible unions of base relations.

Given an $\mathsf{a}$-tuple of domain object $(o_1, \ldots, o_\mathsf{a}) \in \mathcal{D}^\mathsf{a}$ we say that *"the relation* R *holds over* $o_1, \ldots, o_\mathsf{a}$*"* or that *"*$o_1, \ldots, o_\mathsf{a}$ *are in the relation* R*"* and we write $\mathrm{R}(o_1, \ldots, o_\mathsf{a})$, to denote $(o_1, \ldots, o_\mathsf{a}) \in \mathrm{R}$. For binary relations it is also possible to use the more intuitive notation $o_1 \mathrm{R} o_2$ to say that $o_1$ is in relation R with $o_2$.

Finally it has to be noted that, although different relations can have different arities, a qualitative spatial model is typically defined over a set of relations of uniform arity $\mathsf{a}$, in which case one speaks of an $\mathsf{a}$-ary model.

The definition of relations suffices for representation purposes, whereas the conception of a set of operations over the defined symbols provides the model with reasoning capabilities.

### 2.2.3 Qualitative Reasoning: Relational Operations

According to Cohn & Renz (2008, pag. 572), spatial reasoning is concerned with "*[. . . ] deriving new knowledge from given information, checking consistency of given information, updating the given knowledge, or finding a minimal representation.*". In summary, reasoning consists in performing some sort of manipulation over given pieces of information to infer other (not necessarily new) pieces of information.

Classical set-theoretic operations are commonly used in QSR. The union operation $\cup$ is a binary operation that can be used to adduce uncertainty (produce

disjunctive relations). Contrarily, the intersection operation $\cap$ allows for refining disjunctive relations and can be used to reduce uncertainty. Intersection can also be used to check *consistency* over a set of given relations. Broadly speaking (consistency checking is treated in more detail in Section 2.2.5), a relation set is consistent if the relations of the set do not contradict each other, i.e. intersecting any pair of relations does not yield the empty set. Finally, the complement is a unary operation that can be used to express logical negation. E.g. to say that objects $o_1$ and $o_2$ are not in relation R we write $\neg \mathrm{R}(o_1, o_2)$ that is equivalent to the disjunction of all of the base relations but R.

Beyond standard set-theoretic operations, qualitative models are usually provided with two more kinds of operation that provide the main core for entailment: *permutation* and *composition*.

**Permutation** Permutation is a unary operation that, given the relation $\mathrm{R}_i$, holding over a domain element $\mathsf{a}$-tuple $(o_1, \ldots, o_{\mathsf{a}})$, yields the relation $\mathrm{R}_j$ holding over a permutation of the element tuple. Accordingly, the number of possible permutation operations depends on the arity $a$ of the model and is equal to $a! - 1$.

For binary relations it is possible to define only one permutation operation:

**Definition 2.13** (Converse of a binary relation) **-** Let $\mathcal{B}$ be a JEPD set of binary base relations defined over a domain $\mathcal{D}$ and $\mathrm{R} \in 2^{\mathcal{B}}$ a disjunctive relation. The unary *converse* operation ($^\smile$) is defined as:

$$\mathrm{R}^\smile = \{(o_2, o_1) \in \mathcal{D}^2 | (o_1, o_2) \in \mathrm{R}\}$$

For the ternary case it is possible to distinguish up to five different permutations. A possible nomenclature for them is introduced in (Freksa & Zimmermann, 1992) and later reused in (Wallgrün *et al.*, 2007):

**Definition 2.14** (Permutations of a ternary relation) **-** Let $\mathcal{B}$ be a JEPD set of ternary base relations defined over a domain $\mathcal{D}$ and $\mathrm{R} \in 2^{\mathcal{B}}$ a disjunctive relation. The five possible permutation operations are defined as follows:

| | |
|---|---|
| *Shortcut:* | $\mathrm{SC}(\mathrm{R}) = \{(o_1, o_3, o_2) \mid (o_1, o_2, o_3) \in \mathrm{R}\}$ |
| *Inverse:* | $\mathrm{INV}(\mathrm{R}) = \{(o_2, o_1, o_3) \mid (o_1, o_2, o_3) \in \mathrm{R}\}$ |
| *Homing:* | $\mathrm{HM}(\mathrm{R}) = \{(o_2, o_3, o_1) \mid (o_1, o_2, o_3) \in \mathrm{R}\}$ |
| *Shortcut inverse:* | $\mathrm{SCI}(\mathrm{R}) = \{(o_3, o_1, o_2) \mid (o_1, o_2, o_3) \in \mathrm{R}\}$ |
| *Homing inverse:* | $\mathrm{HMI}(\mathrm{R}) = \{(o_3, o_2, o_1) \mid (o_1, o_2, o_3) \in \mathrm{R}\}$ |

**Composition** Composition is a binary[1] operation that, given the relations holding over two overlapping $\mathsf{a}$-tuples of domain elements, yields the relation holding over an $\mathsf{a}$-tuple obtained from a concatenation of the given $\mathsf{a}$-tuples. For example, given the binary relations $\mathrm{R}_i(o_1, o_2)$ and $\mathrm{R}_j(o_2, o_3)$, a possible composition operation yields the relation $\mathrm{R}_k(o_1, o_3)$. For a generic $\mathsf{a}$-ary model it is possible to define up to $(\mathsf{a}!)^3$ composition operations according to the permutation of objects considered in the input and output relations.

For binary models it is commonly defined only one composition operation:

**Definition 2.15** (Composition of binary relations) **-** Let $\mathcal{B}$ be a JEPD set of binary base relations defined over a domain $\mathcal{D}$ and $\mathrm{R}_i, \mathrm{R}_j \in 2^{\mathcal{B}}$ two disjunctive relations. The *binary composition* operation is denoted by the symbol $\circ$ and is defined as follows:

$$\mathrm{R}_i \circ \mathrm{R}_j = \{(o_1, o_3) \in \mathcal{D}^2 | \exists o_2 \in \mathcal{D} : (o_1, o_2) \in \mathrm{R}_i \wedge (o_2, o_3) \in \mathrm{R}_j\}$$

If the set of disjunctive relations $2^{\mathcal{B}}$ defined for a qualitative spatial model is closed under intersection, union, complement, permutation, and composition then the model is commonly referred to as a qualitative calculus, to stress the fact that it also allows for symbolic reasoning beyond the mere representation. However, it is quite common that relations of spatial models are not closed at least under some of the aforementioned operations. In this case it is necessary to weaken the definition of the problematic operations in order to still be able to perform symbolic reasoning correctly. This is done by defining *weak* versions of the problematic operations: A weak operation yields the smallest relation in $2^{\mathcal{B}}$ containing the result of the original operations.

Commonly the operation under which spatial calculi are not closed is the composition. *Weak composition* (cf. Düntsch *et al.*, 2001; Renz & Ligozat, 2005) is defined as:

**Definition 2.16** (Weak composition of binary relations) **-** Let $\mathcal{B}$ be a JEPD set of binary base relations defined over a domain $\mathcal{D}$ and $\mathrm{R}_i, \mathrm{R}_j \in 2^{\mathcal{B}}$ two disjunctive relations. The *weak composition* operation is denoted by the symbol $\diamond$ and is defined as follows:

$$\mathrm{R}_i \diamond \mathrm{R}_j = \{\mathrm{R}_k \in \mathcal{B} | (\mathrm{R}_i \circ \mathrm{R}_j) \cap \mathrm{R}_k \neq \emptyset\}$$

Weak and strong composition can be defined for ternary calculi in a similar way. Moreover, the definitions of permutation and composition operations can also be extended to generic $\mathsf{a}$-ary models (cf. Condotta *et al.*, 2006).

If a calculus is closed under permutation and composition (or a weak form of them) the application of such operations over a relation in $2^{\mathcal{B}}$ yields another relation in $2^{\mathcal{B}}$. Moreover, as the base relation set $\mathcal{B}$ is finite it is possible, and

[1] It is possible to define also higher-arity composition operations. Ternary composition, for example is defined in (Condotta *et al.*, 2006).

common, to provide tables summarizing precomputed results for any base relation (possibly also for disjunctive ones). In the literature, tables summarizing permutation and composition operations are commonly referred to as *permutation tables* and *composition tables*, respectively. Since both types of tables allow for performing qualitative reasoning, in the scope of this work we will use the generic term *reasoning tables* to refer to either permutation or composition tables.

### 2.2.4 Classification of Qualitative Spatial Calculi

Research in QSR has led to the birth of a plethora of qualitative calculi. They vary widely with respect to features spanning from the modeled entities to the development techniques and underlying theory bases and to the modeled spatial aspects.

The research conducted so far has mainly focused on theoretical issues and the majority of developed qualitative calculi is usually focused on a single aspect of space—e.g. topology, distance, and direction. The main aim has been that of identifying singularities and algebraic properties holding on accurately bounded domains. However, according to (Egenhofer & Sharma, 1993; Renz & Nebel, 2007; Sharma, 1996), when it comes to real applications it is mostly necessary to focus on methods for dealing with multiple aspects simultaneously .

This section provides a survey and a classification of best known qualitative spatial calculi. The proposed sorting is similar in its nature to the work presented in (Freksa & Röhrig, 1993), and, rather than being based on algebraic properties of spatial calculi, aims at framing the spatial calculi within a categorizing coordinate system where axis dimensions represent useful characteristics for their utilization in a real-case scenario. The classification criteria are: qualitative relation type, frame of reference, arity, modeled space, and modeled spatial aspect.

#### 2.2.4.1 Classification of Spatial Relations

According to Clementini & Di Felice (1997) spatial relations can be classified in *topological*, *projective*, and *metric*. Topological relations express those geometric properties that stay unchanged under a topological transformation. Technically, a topological transformation is a bicontinuous mapping[1] from a topological space $X$ to a subset of a topological space $Y$. Informally, a topological space is called a "rubber sheet" and a topological transformation can be imagined as a continuous deformation of an object into another object obtained by a manipulation of such a rubber sheet. The manipulation can consist in shrinking, stretching, and twisting but not in cutting and glueing.

[1] A bicontinuous function is a continuous function admitting a continuous inverse.

Projective relations are those that remain invariant under projective transformations which are more restrictive than topological ones as they also preserve collinearity and cross-ratio[1].

Lastly it comes to metric relations when considering properties holding in Euclidean space, where quantitative concepts like angles and distances are defined.

Topological and projective relations are conceived upon topological and projective geometric spaces respectively, where such quantitative concepts are not defined at all, therefore they naturally have a qualitative connotation. On the other hand, metric relations are by their own nature quantitative. To turn them into qualitative relations one has to discretize real values to obtain a set of equivalence classes or, according to measure terminology introduced by Stevens (1946), to transform "ratio" measures into "ordinal" ones .

#### 2.2.4.2 Frames of Reference, Acceptance Areas, and Relation Arity

Practically, a spatial relation specifies a given interdependency between a *primary object* and one or more *reference objects*. The arity of a relation is the number of objects upon which the relation is defined and, although it primarily depends on the modeled spatial aspect, it can also be influenced, especially for projective relations, by the chosen frame of reference (FoR).

FoRs are investigated in many fields, e.g. philosophy, psychology, linguistics, and geography, and as many different terminologies have been developed. On the base of some linguistic motivations, Levinson (1996) presents a tripartite classification:

**Intrinsic:** the FoR depends, at least partly, on some object-specific property or feature. E.g. the front and the back of a person are determined by the asymmetry of the human body.

**Relative:** the FoR depends on a viewpoint. E.g. from one side of a river one sees the water stream going from left to right while, from the other side, one sees the water streaming from right to left.

**Absolute:** the FoR is based on fixed bearings. A typical example here is about geographic cardinal directions.

In the qualitative spatial domain a FoR can be intended as a partition scheme of the modeled space that depends on the reference objects. The zones resulting from the partition are commonly named *acceptance areas* (Clementini *et al.*, 1997) or *sectors* (Moratz *et al.*, 2005) and are a useful means to provide a more practical grasp on qualitative spatial relations.

The acceptance area $Z_{\mathrm{R}}$ of a given $\mathsf{a}$-ary qualitative relation R is the geometric interpretation of the relation definition. It is a region of the modeled space

[1]Cross-ratio is an important property in projective spaces. Given four collinear points $a, b, c, d$ the cross-ratio is the real number $\frac{ac \cdot bd}{bc \cdot ad}$

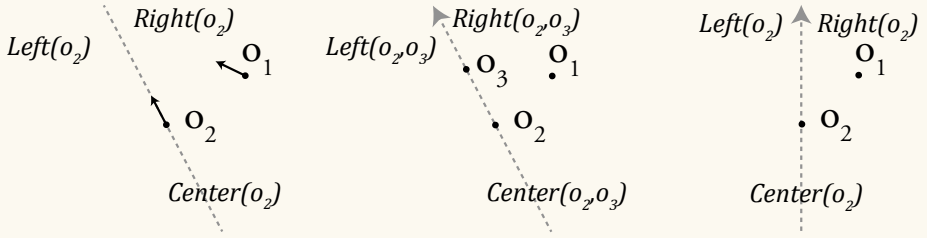
parametric with respect to the reference objects intervening in the relation such that a primary object $o_1$ is in relation R with the reference objects $(o_2, \ldots, o_a)$ if and only if $o_1$ completely falls within $Z_R(o_2, \ldots, o_a)$. Accordingly, the arity of a spatial calculus directly depends on the number of objects necessary to build its FoR.

**Example 2.4** (Direction relations) **-** Figure 2.5 shows three possible alternatives for a directional FoR. In each case the plane is split into the same three acceptance areas: *Left* (L), *Center* (C) and *Right* (R).

Figure 2.5(a) depicts an intrinsic FoR whose orientation depends on some property of the modeled objects which are therefore represented as oriented points. In such a case one reference object suffices to define the partition scheme and directional information can be encoded via binary relations: $R(o_1, o_2)$.

Figure 2.5(b) depicts a relative FoR. In this case the orientation is intended as a perspective from a reference point, cannot be inferred by any object feature and requires two points to be defined, namely $o_2$ and $o_3$. Accordingly the relations are ternary: $R(o_1, o_2, o_3)$.

Finally, Figure 2.5(c) reports an example of absolute FoR. The orientation is globally fixed and represented by a vertical directed line passing through the reference object at hand. Such a FoR gives rise to binary relations: $R(o_1, o_2)$.




**(a)** An intrinsic FoR. **(b)** A relative FoR. **(c)** An absolute FoR.

**Figure 2.5:** Modeling direction relations by different frame of references (FORs). The arity of qualitative relations also depends on the chosen FOR

### 2.2.4.3 Modeled Spaces and Spatial Primitives

Early qualitative spatial calculi have been developed at the beginning of the 1980's and they were mainly about relations among points or line segments in 2-D space. Such calculi provided valuable insights for the development of a solid qualitative spatial theory; however, modeling real world objects as points and lines is not always a satisfactory alternative.

To accommodate real world entities modeling requests, qualitative calculi dealing with regions in the plane have started to be devised. The changeover to such calculi immediately brought new issues into play; namely more complex partition schemes and the possibility that an object overlaps several acceptance areas.

**Example 2.5** (Qualitative directions for regions) **-** Figure 2.6 depicts a possible extension of the partition scheme in Figure 2.5(c) which allows for dealing with regions in the plane. This time two directed lines tangent to the reference object have to be drawn. The partition still yields three acceptance zones and, correspondingly, the three qualitative relations: *Left* (L), *Center* (C), and *Right* (R).

Note that the object $o_1$ overlaps two acceptance areas. According to the theoretical framework we traced so far this corresponds to saying that the relations $\text{C}(o_1, o_2)$ and $\text{R}(o_1, o_2)$ hold simultaneously. This is an infringement of the condition that the set of base relations has to be JEPD.

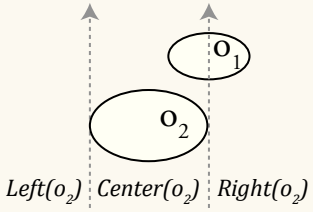



**Figure 2.6:** Frame of reference for extended objects. Single acceptance zones do not provide a JEPD relation set.

To overcome the problem of regions spanning multiple acceptance areas, every combination of the acceptance areas compatible with the kind of modeled primitives has to be included in the base relation set $\mathcal{B}$. So, for instance, if one is only interested in modeling simple regions[1] a JEPD relation set for the toy directional calculus in the previous example is $\mathcal{B} = \{\text{L}, \text{R}, \text{C}, \text{LC}, \text{CR}, \text{LCR}\}$. In the literature (Goyal & Egenhofer, 2000; Skiadopoulos & Koubarakis, 2004) the newly introduced relations, i.e. *LeftAndCenter* (LC), *CenterAndRight* (CR), and *LeftAndCenterAndRight* (LCR), are termed *multi-tile relations*, in contrast to the others that are referred to as *single-tile relations*. Note that the only combination excluded from the set is the multi-tile relation *LeftAndRight* (LR) as it can never occur that a primary object overlaps $Left$ and *Right* acceptance areas without also overlapping $Center$. However, such a relation has to be included if multi-regions[2] have to be modeled.

This work is mainly concerned with regions in 2-D space. However, for the sake of completeness, let us conclude with a few examples about 3-D qualitative modeling which recently has started to be investigated more actively: E.g. Billen & Zlatanova (2003) present a framework for the treatment of qualitative relations in 3-D cadastral applications whereas both Tassoni *et al.* (2011) and Bartie *et al.* (2011) deal with visibility relations among polyhedron in 3-D space.

[1] A simple region is a convex, connected and hole-free region.
[2] A multi-region is a region composed of several disconnected parts.

#### 2.2.4.4 Spatial Aspects

It seems a shared opinion (cf. Clementini & Di Felice, 2000; Freksa & Röhrig, 1993, among others) that the most fundamental aspects of space are topology, direction, and distance which reflect topological, projective, and metric properties, respectively. In fact, this is confirmed by some psychological findings of Piaget & Inhelder (1967) which suggest that children first apprehend topological concepts whereas projective and metric ones are recognized at later stages after having learnt of different view points and understood distances.

**Topology** Topology is probably the best studied spatial aspect in QSR. Topological calculi typically model connectivity among extended entities in 2-D or 3-D space and the most famous ones are the Region Connection Calculus (RCC) (Randell & Cohn, 1989; Randell *et al.*, 1992) and the 9-Intersection Model (9-IM) (Egenhofer, 1989, 1991).

RCC is based on formal logic and derives a set of fifteen relations from the primitive concept of connectedness among two entities. Eight of the initial fifteen relations have been discovered to form a JEPD set named RCC-8.

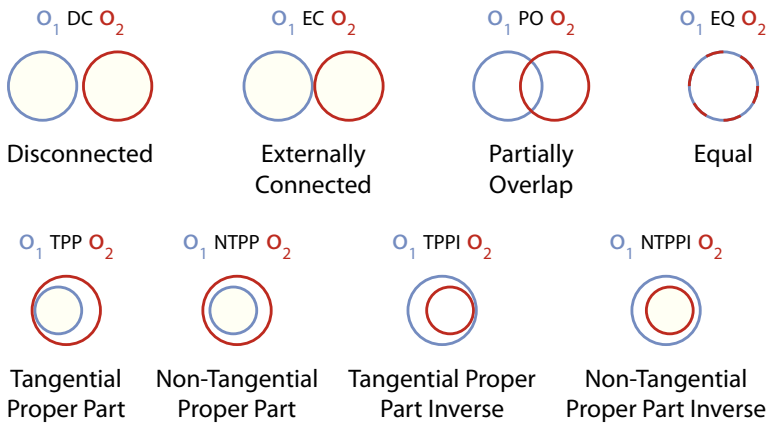



**Figure 2.7:** 8 jointly exhaustive and pairwise disjoint binary topological relations between extended objects: the Region Connection Calculus and the 9 Intersection Model.

Egenhofer's approach is based on combinatorics. A region is considered to be consisting of an interior and a boundary, whose intersection is the empty set and whose union yields a closed point set representing the region. Beyond these constituents, also the exterior of a region is considered and a relation between two regions is represented by means of a 3x3 Boolean matrix: Each matrix element stands for the intersection of one constituent element of the first region with another element of the second one. A matrix element is 1 if the corresponding constituents overlap, 0 otherwise. When considering simple regions, out of the $2^9$ matrix instances only eight can be spatially realized which can be directly mapped onto the RCC-8 ones.

Figure 2.7 reports the eight topological relations named according to RCC-8 nomenclature. Both calculi are provided with reasoning tables (cf. Cohn *et al.*, 1997; Egenhofer, 1991) for both binary composition and converse operations.

Due to the way the two models have been developed each has certain advantages. In particular, RCC-8 is better suited for logical and theoretical studies as it provides a formal logic theory, while 9-IM is mostly used in application domains because of its constructive definition. Note that in the remainder of this text we will refer to RCC-8 with the shorter notation RCC.

**Direction** The representation of direction information is also very well studied and a variety of models, typically further classified in relative and cardinal directions, is available.

Relative relations can employ either a relative or an intrinsic FoR and some of the the best known models are listed below. The single cross calculus (Freksa, 1992) is a binary model that assumes points in the plane as primitives. It employs an intrinsic FoR and a cross-shaped partition scheme centered in one reference point that subdivides the plane into eight sectors. The double cross calculus (Freksa, 1992; Freksa & Zimmermann, 1992) resorts to a similar partition scheme but it assumes a relative FoR and is therefore a ternary model. It splits the plane into fifteen sectors by means of the directed line passing through the two reference points and the two normals through these points. The Oriented Point Relation Algebra ($\mathcal{OPRA}$) (Moratz, 2006; Moratz *et al.*, 2005) and the $\mathcal{STAR}$ calculus (Renz & Mitra, 2004) are parametrized binary calculi dealing with points. The FoR is absolute but the main innovation here is that the number of fixed bearings varies according to the value assigned to a parameter. An interesting variation is provided by the Dipole Relation Algebra ($\mathcal{DRA}$) (Moratz *et al.*, 2000; Schlieder, 1995) that models directions between directed line segments (dipoles). A relation is composed by a quadruplet of symbols representing the direction (left or right) of every dipole endpoint with respect to the other dipole. A solution dealing with extended objects is provided by the 5-intersection model (Billen & Clementini, 2004a,b) that assumes a relative FoR and provides a JEPD set of ternary relations. The plane is split into five acceptance areas by means of the four tangents existing between a pair of regions. The five single-tile relations are called $LeftSide$, $RightSide$, $Between$, $Before$, and $After$.

Cardinal direction models are typically based on absolute FoRs like the cardinal direction calculus for points (Frank, 1991; Ligozat, 1998) distinguishing the four classical cardinal directions and four intermediate bearings and coming in two variants: with a cross-shaped and a cone-shaped partition scheme. Finally we present two models for extended objects. The Rectangle Algebra ($\mathcal{RA}$) (Balbiani *et al.*, 1998; Guesgen, 1989; Mukerjee & Joe, 1990) proposes a 2-D extension of Allen's interval calculus which deals with rectangles uniformly aligned with the Cartesian axes: The relations among two rectangles is identified by the pair of interval relations holding on the $x$- and $y$-projections of the rectangles under consideration. Lastly, the Cardinal Direction Calculus (CDC) (Goyal & Egenhofer,

in press, 1997) is capable of handling generic regions in the plane. The partition scheme is based on the four lines tangent to the edges of the Minimum Bounding Rectangle (MBR) of the reference object (cf. Figure 2.8). This model defines eight single-tile relations corresponding to the classical cardinal bearings NW, N, NE, E, W, SW, S, SE plus a ninth relation B corresponding to the reference object MBR. Skiadopoulos & Koubarakis (2004) present a deep investigation of the properties of the model and they also provide a method to automatically compute reasoning tables for the multi-tile relations starting from those for single-tile.

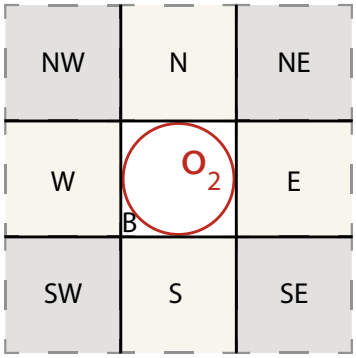



**Figure 2.8:** Frame of reference of the Cardinal Direction Calculus: it splits the space into 9 acceptance zones corresponding to as many single-tile base relations corresponding to the 8 canonical cardinal bearings plus an extra relation indicating the Minimum Bounding Rectangle of the reference object.

**Distance** Generally, models for the qualitative representation of distances aim at defining equivalence classes for metrically represented distances. Boundary values for equivalence classes can be chosen in many ways, reflecting the fact that distance is quite a tricky aspect to model since, usually, its conceptualization is inherently related to different factors. For example, an object can be perceived as close or distant according to the transportation vehicle at hand. Sometimes it is more convenient to describe distance in terms of the time necessary to cover the space among two points. Again, distance is usually associated with size and scale of referred entities: the city of Bremen can be considered as close to Berlin at a worldwide scale but far away within Germany. For such reasons usually qualitatively represented distance is not modeled as a stand-alone spatial aspect but within a larger qualitative framework.

Hernandez, Clementini & Di Felice (1995), for example, provide a partition scheme for qualitative distances based on a series of concentric circumferences centered at a reference object, however when it comes to the generation of reasoning tables, they also have to consider orientation in order to have somewhat usable reasoning tables. This work is further extended in (Clementini *et al.*, 1997) where orientation is soundly incorporated in the FoR: a star-shaped partition scheme centered at the reference object is superimposed on the concentric circumferences.

Similarly, Moratz & Ragni (2008) develop a ternary calculus capable of modeling position of a primary object with respect to a reference object observed by an agent. The work has cognitive robotic motivation, the considered primitives are points and the space partition scheme extends that of the single-cross calculus (Freksa, 1992): two lines through the observed reference object and a circumference centered on it are traced on top of the original partition scheme. It has also cognitive robotic motivations the work from Moratz & Wallgrün (2003) that deals with propagation of distance and orientation intervals within topological maps based on generalized Voronoi graphs. The aim is to generate hypotheses about cycles in the navigation environment that can be subsequently checked with quantitative approaches. Lastly, a different approach is undertaken in (Yao & Thill, 2006) where a modal system for qualitatively represented distances is introduced. The system distinguishes distances not only according to metric considerations but also taking into account the used transportation means. The authors exploit the system for distance-based retrieval of points of interest from a geographic database.

**Other Qualitatively Represented Spatial Aspects** Other kinds of qualitative spatial relations can possibly be expressed as a combination of topological, directional, and metric primitives. However, since referring back to such primitives leads to cumbersome manipulations of symbols, it is common habit to develop dedicated models over sets of symbols that directly address relations one is willing to reason about. Some examples are about visibility (Fogliaroni *et al.*, 2009; Santos *et al.*, 2009; Tarquini *et al.*, 2007) shape (Clementini & Di Felice, 1997; Cohn, 1995; Falomir *et al.*, 2010) and size (Bittner & Donnelly, 2007; Raiman, 1991; Zimmermann, 1995).

### 2.2.5 Qualitative Constraint Networks

One of most spread methods to perform qualitative reasoning resorts to the utilization of so-called Qualitative Constraint Networks (QCNs) (cf. Dechter, 1992, 2003; Montanari, 1974, for a detailed discussion).

**Definition 2.17** (Qualitative Constraint Network) **-** A *qualitative constraint network* is a triple $N = (X, \mathcal{D}, \Xi)$ where $X = \{x_1, \ldots, x_n\}$ is a set of variables ranging over the domain $\mathcal{D}$ and $\Xi = \{\xi_1, \ldots, \xi_n\}$ is a set of constraints over the variables. A constraint $\xi_i$ is a pair $\xi_i = (S_i, \mathrm{R}_i)$ where $S_i \subseteq X$ is a subset of the variable set and $\mathrm{R}_i$ is a qualitative relation defined over $S_i = \{x_1, \ldots, x_m\}$ which is called the scope of the constraint.

QCNs are suitable mathematical instruments for qualitatively representing a spatial scene. They can be seen as directed (hyper)graphs (cf. Section 2.1) where the nodes are the spatial objects occurring in the scene and the arcs stand for the relations intervening among them.

**Example 2.6** (Qualitative Spatial Representation) **-** Let us consider the spatial scene depicted in Figure 2.9(a) and the toy calculus introduced in Example 2.5 defining the following base relations: $\mathcal{B} = \{\text{L}, \text{R}, \text{C}, \text{LC}, \text{CR}, \text{LCR}\}$. Figure 2.9(b) reports the corresponding QCN representation.


**(a)** A geometric description. **(b)** A qualitative description.

**Figure 2.9:** A Qualitative Constraint Network (b) can be used to represent the spatial scene whose geometric representation is given in (a). The symbols L, R, C stand for the base qualitative relations *Left*, *Right*, *Center*, respectively. LCR stands for the base multi-tile relation *LeftAndCenterAndRight*.

The arcs in the (hyper)graph representation of a QCN can be labeled with disjunctive relations, meaning that the relation among two objects is uncertain or, equivalently, *underspecified*. A QCN is said *complete* if the relations among every $\mathsf{a}$-tuple of objects is one from the set of base relations $\mathcal{B}$. For instance, the network in Figure 2.9(b) is complete: note the presence of loops[1]. If the relation among two objects is unknown—it can be any from the set $2^{\mathcal{B}}$ of disjunctive relations—the corresponding arc in the graph representation is unreported.

**Definition 2.18** (Solution of a QCN) **-** Given a QCN $N = (X, \mathcal{D}, \Xi)$ a solution $sol(N)$ of $N$ is an assignment from the domain $\mathcal{D}$ of all its variables $X$ such that all the constraints in $\Xi$ are satisfied.

One major problem is that of deciding whether a solution exists for a given QCN, that is, to decide whether the constraints in $\Xi$ do not contradict one another. This is a particular case of so-called Constraint Satisfaction Problems (CSPs). If the domain $\mathcal{D}$ is infinite, which is for the spatial case, the consistency of a network can be checked via the *algebraic closure* algorithm which was first introduced in (Montanari, 1974) for binary calculi and later refined in (Mackworth, 1977). Basically, the algorithm exploits composition and converse operations to *propagate the constraints* and to verify that they do not conflict with each other, i.e. the relation yielded by an operation is a superset of the relation reported

[1]A loop is an arc originating and ending in the same node and has not to be confused with a cycle (cf. Section 2.1.2).

in the network. The *algebraic closure* algorithm can work with both strong and weak operations (cf. Section 2.2.3) and executes in $\mathcal{O}(n^3)$ worst case time where $n$ is the number of variables in the network. In (Dylla & Moratz, 2004) the algorithm is further extended for ternary calculi, in which case the complexity rises up to $\mathcal{O}(n^4)$.

### 2.2.6 Integration of Qualitative Models

It has been already recognized in the past (Egenhofer & Sharma, 1993; Renz & Nebel, 2007; Sharma, 1996) that, despite the investment of more than thirty years of research efforts in the field of QSR and the development of a vast and heterogeneous set of theoretical frameworks (cf. Section 2.2.4 for a classification), the application of the qualitative paradigm to tackle realistic problems remains relatively small.

One main reason is that human beings are generally capable of mixing several aspects when communicating and reasoning about spatial knowledge. Therefore, the exploitation of qualitative reasoning to realistic applications calls for a more general theory that can embrace the big variety of spatial features.

The *poverty conjecture*, initially enunciated for qualitative kinematics (Forbus *et al.*, 1987) and later slightly modified (Forbus *et al.*, 1991) to embrace spatial properties in general, leaves few hopes for the development of such a general theory. The conjecture claims that "*There is no purely qualitative, general-purpose representation of spatial properties*" and is further elaborated by adducing some elucidations: (*i*) Qualitative spatial representations are useful in many cases and they are fundamental to enable commonsense reasoning; however they must be related to quantitative information. (*ii*) Qualitative models have to be designed in a task-specific manner.

An alternative to enable practical application of QSR draws upon the integration of different qualitative models. This can be done in mainly two ways: (*i*) calculi fusion and (*ii*) inter–calculus reasoning.

The first approach consists in the combination of different calculi at both, a representational and reasoning level to obtain a new calculus. That is, several aspects of space are considered simultaneously to represent more complex relations and dedicated reasoning algorithms accounting for such complex aspects are developed. The literature is full of examples for combining calculi. Some of them are about the integration of topology and direction (Hernandez, 1994; Papadias, 1994; Papadias & Sellis, 1994b; Papadias *et al.*, 1995), topology and size (Gerevini & Renz, 1998, 2002), direction and distance (Clementini *et al.*, 1997), or more complex combinations (Clementini & Di Felice, 1997; Sharma, 1996).

The second approach mainly grounds in the algebraic properties of qualitative calculi. It consists in developing advanced reasoning techniques that allow for exploiting interdependencies between calculi. An example is provided by the Joint Satisfaction Problem (JSP): the problem of deciding whether the *joint network* obtained by merging two consistent QCNs is itself consistent. Some works in this

direction are (Gerevini & Renz, 1998, 2002) that investigate the case of RCC-8 with a newly defined calculus for size, (Li, 2007; Li & Cohn, 2009) considering topology, direction, and size, and (Liu *et al.*, 2009) focusing on complexity issues when joining RCC-8 with two cardinal direction calculi: $\mathcal{RA}$ and CDC.

### 2.2.7 On the Cognitive Adequacy of Qualitative Models

Several studies have been carried on to assess the cognitive adequacy of qualitative spatial models. Knauff *et al.* (1997) argue that "*whether or not a formal approach to qualitative spatial relations is a cognitively adequate model of human spatial knowledge is in fact an empirical question and can be answered only on the basis of empirical results*". Therefore they conducted experiments to assess the cognitive adequacy of RCC and 9-IM (cf. section 2.2.4.4). The results outlined that such models correctly reflect human way of dealing with topological relations. No equally sound results have been provided for other aspect of space like, for example, cardinal directions. However, much research supports the assumption that people usually reason about directions differentiating among a small set of qualitative directions (from four up to eight according to the level of accuracy required), therefore legitimating the underlying assumption that qualitative directional models can suitably encode this kind of spatial information. Similarly in the context of this work we generalize such an assumption and assume that qualitative models, if adequately designed, can provide a suitable means to represent human spatial knowledge; proving cognitive adequacy of qualitative models is not the aim of this work though.

## 2.3 Spatial Databases

A Geographic Information System (GIS) is a system of hardware and software elements that allows for surveying, storing, analyzing, manipulating, retrieving, sharing, mapping[1], and, more generally, presenting geographic data. At the core of a GIS is the storage layer, that is, a support to persistently represent data in a computer. Storing can be done by means of standard files—e.g. shapefiles (ESRI, 1998). Nonetheless, the usage of a database in this role can notably improve data retrieval and analysis performance. Databases allow for storing many dimensions of data, but they are not designed to manage multidimensional data as a whole. Therefore, given its multidimensional nature, spatial information cannot be easily handled by plain databases without a so-called "spatial extension".

Spatial databases are databases extended with data types and procedures for inserting, deleting, manipulating, and retrieving spatial information. Technically, a GIS needs to deal with geographically referenced spatial information, therefore one usually makes a further differentiation between spatial databases

[1]Mapping in this context is intended as the action of producing a map.

and geospatial databases (or simply geodatabases). Geodatabases are primarily spatial databases that offer support for geo-referencing spatial objects. However, although the findings presented in this text have been prompted by motivations of geographic nature, they completely disregard the geographic component and are general enough to work with spatial information in general. Therefore, in the reminder of this text we focus on the more general spatial databases.

### 2.3.1 Spatial Data Modeling and Representation

In the geographic domain one is interested in modeling real and fictitious entities. Real entities comprise physical objects, natural or artificial, that actually exist in the world, e.g. a lake or a building, whereas fictitious entities are those entities that are not physically distinguishable from surrounding ones, e.g. administrative districts or countries. Rather, boundaries of fictitious entities are defined artificially by humans, e.g. the borderline between two countries is an imagery line decided according to some political issues. An ontological perspective on this issue is taken by Smith (1995) who distinguishes between entities with *bona fide* and *fiat* boundaries, respectively.

Modeling spatial information can be done mainly following two approaches that in the literature (Longley *et al.*, 2005; Worboys & Duckham, 2004) are typically referred to as field-based and object-based. The field-based approach looks at the properties that have to be modeled as continuous fields and discretizes them via the superimposition of a—typically regular—geometric structure, e.g. a grid. With every cell of the structure a value is associated that summarizes features—e.g. the average—of the modeled property ranging over the points contained in the cell. This kind of representation is commonly referred to as *raster*.

The object-based approach is diametrically opposite to the field-based approach since the main focus is on the spatial entities which are modeled as first order objects. The typical resulting representation is referred to as *vector–based* since boundaries of spatial entities of interest are geometrically described by means of directed line segments called vectors. A vector is defined by the specification of its end points in terms of geometric coordinates.

Rasters and vectors are generated from different sources, are different at the representational level, they better lend themselves to different modeling purposes, and have to be managed differently.

Rasters present a good solution for modeling values varying continuously over space—e.g. digital elevation modeling—they are very similar to bitmap images and, indeed, are typically generated from satellite pictures. Some of the most important types of data management operations are: identification of spatial entities from satellite pictures (raw pixel sets), clustering and interpolation of recorded values, and statistical operations over given areas.

Vector representations are usually obtained from digitalization of charts, from vectorization of raster data, or from the direct collection of real–world entity coordinates—e.g. through GPS receivers. In such representations objects are

defined explicitly and the main operations regard identification of relationship among objects and the application of computational geometry algorithms—e.g. intersection, union, difference, and so forth.

Given the diversity of tools and techniques needed to handle the two data formats, a finer classification of databases dealing with rasters and vectors can be done. According to Güting (1994), databases providing instruments to treat raster data are better classified as image databases, whereas the term spatial database is reserved for those dealing with vector data. Although some of the techniques presented in this work can be adapted to work with rasters, they are mainly designed for vector data; therefore in the reminder of the text a vector representation is implicitly assumed and the term “spatial database" is used in the sense of Güting.

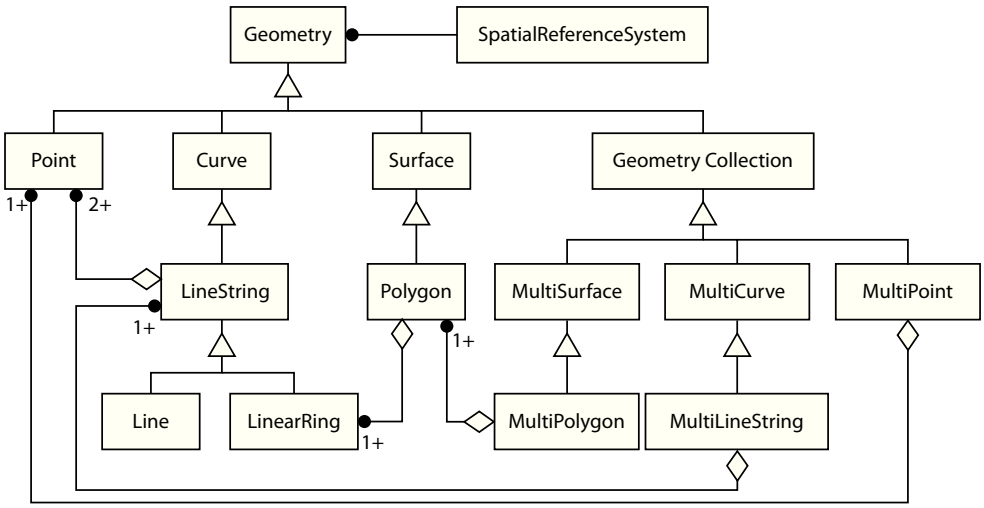



**Figure 2.10:** The geometric data type hierarchy as defined in (OpenGIS Consortium, 1998).

Although semantically different, real and fictitious spatial entities are usually represented by means of the same geometric primitives at the modeling level. The OpenGIS Consortium (OGC) (OpenGIS Consortium, 1998) released a hierarchical specification of such primitives as depicted in Figure 2.10. For the findings of this thesis, the semantic distinction of spatial entities is not relevant, therefore a spatial feature, be it real or fictitious, will be simply addressed with the terms geometry (or any more specific identifier in the hierarchy), object or entity; the adjective spatial will be preposed where necessary for the sake of clarity.

### 2.3.2 Spatial Data Management: Spatial Operators

Management of spatial data in spatial databases is provided by so-called spatial operators. Generally, a spatial operator is a function defined over a set of spatial objects that performs some kind of geometric operation over the input data and returns a result. Operators can apply to any spatial primitive or can be dedicated

to specific ones. Table 2.1 reports generic spatial operators defined in (OpenGIS Consortium, 1998) applicable over the whole geometric hierarchy in Figure 2.10.

Most simple type of spatial operators implement set-theoretic operations—e.g. intersection or union—but more complex ones can also be defined to assess spatial object properties and spatial relationships. An important typology of spatial operators is concerned with spatial relationship determination, referred to as topological operators in Table 2.1.

| **Type** | **Name** | **Description** |
|---|---|---|
| Basic operators | SpatialReference | Returns the spatial reference system of the geometry |
| | Envelope | Returns the minimum bounding rectangle of the geometry |
| | Export | Converts the geometry into a different representation |
| | IsEmpty | Returns TRUE if the geometry is the empty set |
| | IsSimple | Returns TRUE if the geometry i simple |
| | Boundary | Returns the boundary of the geometry |
| Topological operators | Equal | Tests if the geometries are spatially equal |
| | Disjoint | Tests if the geometries are disjoint |
| | Intersect | Tests if the geometries intersect |
| | Touch | Tests if the geometries touch each other |
| | Cross | Tests if the geometries cross each other |
| | Within | Tests if the given geometry is within another given geometry |
| | Contains | Tests if the given geometry contains another given geometry |
| | Overlap | Tests if the given geometry overlaps another given geometry |
| | Relate | Returns TRUE if the spatial relationship specified in the given 9-Intersection matrix holds |
| Basic analysis operators | Distance | Returns the shortest distance between any two points of two given geometries |
| | Buffer | Returns a geometry that represents all points whose distance from the given geometry is less than or equal to the specified distance |
| | ConvexHull | Returns the convex hull of the given geometry |
| | Intersection | Returns the intersection of two given geometries |
| | Union | Returns the union of two given geometries |
| | Difference | Returns the difference of two given geometries |
| | SymDifference | Returns the symmetric difference of two given geometries |

**Table 2.1:** Spatial operators defined in (OpenGIS Consortium, 1998) over the *geometry* type (cf. Figure 2.10).

Clementini & Di Felice (2000) point out the fact that the set of spatial operators defined by the OGC is not complete and should be extended with other fundamental operators according to the tripartite classification of spatial relations into topological, projective, and metric discussed in Section 2.2.4.1.

Practically, spatial operators are implemented as functions within a Database Management System (DBMS) and can therefore be called through the provided Query Language (QL) to manipulate, analyze, and retrieve spatial information from the database. In a query statement, spatial operators can possibly be alternated with standard operators—e.g. sum, subtraction, or Boolean operators.

Spatial data, thus, can be manipulated in several ways by means of different spatial operators. The focus of this work, however, is on spatial data searches and retrievals that will be referred to as spatial queries.

### 2.3.3 Spatial Queries and Indexes

A spatial query is a request to a spatial database that addresses, at least partly, spatial data in its statement, execution, or output. More practically, a spatial query consists in manipulation, analysis, or retrieval of spatial data that is done injecting spatial operators within QL statements.

| **Query** | **Search** |
|---|---|
| Exact match | All geometries identical to a given one |
| Point | All geometries overlapping a given point |
| Window | All geometries overlapping a given rectangular area |
| Intersection | All geometries overlapping a given one |
| Enclosure | All geometries enclosing a given one |
| Containment | All geometries enclosed by a given one |
| Adjacency | All geometries adjacent a given one |
| Nearest-neighbor | All geometries having minimal distance from a given one |

**Table 2.2:** Most common spatial queries as reported in (Gaede & Günther, 1998).

The most common spatial queries, according to Gaede & Günther (1998), are reported in Table 2.2. Such queries come usually in handy for diverse purposes and in diverse scenarios. For this reason they are usually addressed separately in the literature and dedicated solving strategies are developed for each of them that allow for optimizing their executions. However, for the scope of this work it is useful to generally regard them all as *spatial predicate queries* in which the predicate expresses a spatial relation that has to be satisfied. In the first row of Table 2.2 we find the *exact match query* that expresses the RCC relation *equal* (EQ) (cf. Figure 2.7). Next three query kinds are variants of the relation *partially overlap* (PO): it has to be noted that the *point query* can be seen as an extreme case of the *window query* (with the window collapsing into a point)

which, in turn, is a special case of the more general *intersection query*. Therefore, the first four listed query types, together with the *enclosure*, *containment*, and *adjacency* queries, can be categorized as topological predicate queries or, simply, as topological queries. *Nearest-neighbor queries*, are distance predicate queries. Lastly Gaede & Günther (1998) report a further type of query referred to as *spatial join*. Such kind of queries are also spatial predicate queries but the emphasis here is posed on the join operation to be performed between two different datasets. Spatial queries make use of spatial operators (cf. Table 2.1) that have already been recognized not to be exhaustive (cf. Section 2.3.2). Similarly the spatial queries reported in Table 2.2 do not cover all of the possible requests one may be interested in—e.g. directional queries are missing.

Usage of spatial operators makes the syntax of spatial query statements similar to that of non-spatial ones. However, issues related to spatial data manipulation do not exhaust themselves with simple definition of spatial primitives and operators. There are at least two more main issues that have to be accounted for: the development of query and visualization languages suitable to human users and the conception of methods to speed up spatial data access. The former is a big topic in itself that falls largely beyond the scope of this work but will be partly addressed in section 2.3.5. The latter instead is the central theme of the next section.

### 2.3.4 Spatial Access Methods

One main concern of DBMSs is to provide fast access to data stored in secondary memory. Only a small part of data present in a database can be loaded into main memory and, despite technological improvements guaranteeing increasingly better performance, accessing secondary memory remains the bottleneck in information systems. For this reason, historically, DBMSs are furnished with auxiliary data structures designed to reduce the number of secondary memory accesses. Such structures are referred to as *indexes* or *access methods* and keep data organized in such a way that it is not necessary to perform a complete scan of a dataset when executing a search. Instead the index file, typically much smaller than data file, can be completely loaded into main memory and used to find the secondary memory blocks containing the searched data.

**Example 2.7** (Phone book) **-** One of the best known examples of index is the phone book. A phone book contains thousands of entries and it would be impossible to use if they would not be organized in a certain manner. A typical scenario consists in looking for the phone number of a person whose name and surname are known. Given that there are more duplicated first names rather than surnames, entries in a phone book are alphabetically ordered by surname. This simple artifice makes manual search possible because it allows to quickly direct the search to the right book section.

Typically an index is designed to work properly only for certain data types and for a specific set of queries—e.g. the organization of entries in a phone book is not suitable for searching the owner of a known phone number. Spatial data types and the kind of queries one might be willing to perform on them are inherently different from scalar types and queries, therefore a variety of dedicated indexing techniques have been devised: they are referred to as *spatial indexes* or *Spatial Access Methods (*SAM*s)*.

At a general level, spatial indexing approaches can be divided into hash coding, hierarchical approaches, and space-filling curves. Given that no total order exists in the spatial domain, hash functions are realized by means of a partition of the working space. Hierarchical approaches resort to tree-like structures similar to B-trees (Comer, 1979). Lastly, space-filling curves are well determined curves with recurrent patterns that aim at traversing the working space, this way a total order can be specified according to the sequential intersection of the space-filling curves with spatial objects.

Spatial indexes have to be small to fit into main memory. For this reason indexing data structures usually maintain approximations of spatial objects rather than a representation of the objects themselves—e.g. a polygon can be approximated by its centroid or by its Minimum Bounding Rectangle (MBR). In contrast to actual objects, approximations usually need a fixed, known, and smaller amount of memory to be represented. Accordingly representing approximations allows for producing more compact data structures and more efficient algorithms for their management.

Indexes that encode approximations are generally subject to search processes consisting of two phases. (i) The filter phase consists in searching index entries to obtain a superset of the actual query results. Indeed, by considering approximations, the search produces a set of candidate results containing both actual results and false hits. False hits are later ruled out through a (ii) refinement step when actual geometries are retrieved from the stored data and are tested against the requested spatial predicates.

A short summary of spatial access methods relevant for this work is given below. For a complete review refer to (Gaede & Günther, 1998; Samet, 2006).

**Grid-based Approaches** Some of the easiest and most powerful spatial indexing techniques are based on space tessellation by means of uniform grids (Bentley & Friedman, 1979; Franklin, 1978, 1984, 1989).

The space which the objects are embedded into, is partitioned via the superimposition of a grid and a correspondence is created among an object and the grid cells it overlaps.

Grid-based approaches are wonderful candidates to treat any typology of spatial queries, topological, projective, and metric, since the regularity of the grid structure allows for quickly restricting the search space. To answer a window query, for example, it is possible to check which grid cells overlap the given

search window and later refine the search by only considering the spatial objects associated to such cells.

A more advanced technique, still based on grids, is the *grid file* (Nievergelt *et al.*, 1984) which employs a multi-level partition by means of non–uniform grids.

**The R-tree Family** One of the most spread typology of spatial indexes is the R-tree family. The R-tree data structure was originally proposed by Guttman (1984) to boost window queries on n-dimensional data. The main underlying idea is that of grouping objects by proximity and represent each group by its Minimum Bounding Rectangle (MBR). Since all the objects in a group lie within its MBR, a query that does not intersect the MBR cannot intersect any of the contained objects.

Similarly to a B-tree (Comer, 1979), an R-tree is a balanced search tree—i.e. all leaf nodes are at the same level—that maintains references to database objects in its leaves. Each node $n$ in the tree contains index entries of the kind:

$$\mathrm{E} = (\mathrm{R}, r)$$

Leaf nodes are located at level 0, $r$ is a reference to a database object $o$ and $\mathrm{R} = \mathrm{MBR}(o)$ is its minimum bounding rectangle. In non-leaf nodes living at level $l > 0$, $r$ is a reference to a node located at the level $l - 1$ and R is the minimum bounding rectangle enclosing all the MBRs in the referenced node. Each node—except the root—contains between $m$ and $M$ index entries.

An R-tree is constructed iteratively, adding one object at time. When inserting a new object $o$ the tree is traversed from the root downwards. At each traversed node $n$, the insertion procedure selects the index entry $\mathrm{E} = (\mathrm{R}, r)$ whose MBR needs least enlargement to include $o$ and iterates on the node referenced by $r$. When a leaf is reached the object is added to the corresponding node. If the insertion causes a node-overflow—i.e. the leaf contains $M+1$ entries—the affected node is split into two nodes, each containing at least $m$ entries.

The splitting procedure represents the most critical part in the structure management: The split should be done in such a way as to minimize the possibility that both nodes generated by the split have to be visited during a search. Guttman (1984) suggests that, since the decision of visiting a node $n$ depends on the area of the MBR covering $n$, the total area of the MBRs of the new-generated nodes has to be minimized. He proposes two algorithms for doing the split that execute in time quadratic and linear, respectively, with the number $M$ of maximum entries.

Beckmann *et al.* (1990) propose an optimized version of Guttman's data structure called R*-tree. They stress the fact that the R-tree is based on a heuristic based on the minimization of the area of node MBRs and point out that such an optimization criterion is not demonstrated to be effectively the best one. Hence, Beckmann *et al.* go through a re-engineering of the access method and detect

that also the overlapping area among MBRs and their perimeters affect the performance. Accordingly they present a new splitting algorithm that takes into account all such parameters.

More recently, Brakatsoulas *et al.* (2002) proposed a novel variant of the R-tree family called cR-tree. The authors point out that the splitting procedure is basically a clustering problem, therefore, they resort to the k-means algorithm[1] to re-distribute overflowing objects opportunely. They also investigate the possibility of splitting overflowing object sets into $k$ groups, rather than the canonical approach that always splits into two.

Finally, the $R^+$-tree, presented in (Sellis *et al.*, 1987), is another well known variant in which the overlapping of MBRs is prohibited, therefore an object can belong to multiple nodes.

### 2.3.5 Spatial Query Languages

The problem of addressing spatial data is critical in spatial databases. Extensive studies (Egenhofer, 1994; Frank, 1982) have been conducted to identify the key requirements that a spatial Query Language has to fulfill.

For example, a variety of spatial dialects for Structured Query Language (SQL)[2] has been proposed (cf. Ingram & Phillips, 1987; Roussopoulos & Leifker, 1985, among others) before coming to a standard (OpenGIS Consortium, 1998). However, addressing spatial data through the specification of long lists of highly accurate geometric coordinates is a cumbersome and unnatural task for the users of a spatial database. Similarly, making sense of spatial query results presented in a textual format might be very difficult.

Given its nature, spatial data better lends itself to be addressed and presented in a graphical way. For this reason several attempts have been undertaken to elaborate graphical query and visualization languages. Query languages of this type adopt a philosophy similar to that of the, non-spatial, Query-by-Example (QBE) language (Zloof, 1977). Main aim of QBE is to provide an easy-to-use relational database language that does not require the user to be an expert of databases, computer science, or mathematics. In relational databases data is presented in tabular format, therefore QBE assists the user in directly shaping, on screen, the expected result table. Similarly, spatial QBE languages allow the user to provide an exemplary spatial scene to be retrieved.

Early approaches (Chang & Fu, 1980; Joseph & Cardenas, 1988) originated from the field of pictorial (image) databases but were not backed up by a thorough investigation on suitability requisites like the eleven ones presented in (Egenhofer, 1994).

[1] K-means algorithm is one of the best known clustering algorithms. It is also known as Lloyd's algorithm Lloyd (1982).

[2] SQL is a programming language designed for reading, modifying, and managing data stored in a Relational Database Management System (RDBMS).

Such requirements are later concretized in the specification of a Query-by-Sketch (QBS) framework (Egenhofer, 1996) comprehending a touch-sensitive input device to draw a sketch of the scene to be searched. QBS allows for annotating drawn objects with textual tags to facilitate the search—e.g. object categories like a forest or a lake—but does not require the user to specify in advance the relationships he is going to draw, as requested in similar approaches (Calcinelli & Mainguenaud, 1994; Lee & Chin, 1995). Egenhofer's approach is based on a user-system interaction schema according to which the sketch is continuously parsed and, in case of an ambiguity in the interpretation, a message is displayed to prompt the user to resolve it. The sketch is translated into a topological constraint set and, consequently, in a spatial query to be sent to the underlying spatial database. In the case distortions introduced into the sketch make an exact match impossible, the query can be relaxed switching some topological constraints with their conceptual neighbors (cf. Freksa, 1991a).

The actual execution of the query at the database level is not detailed in (Egenhofer, 1996); however in (Clementini *et al.*, 1994) pre-processing of topological query by means of consistency checking algorithms (c.f. Section 2.2.5) is tackled. The main idea is that topologically inconsistent queries either have to be relaxed or do not have to be performed at all. Egenhofer and Clementini *et al.* argue that, although consistency checking is a costly operation, topological queries usually consist of a small number of relations. In such cases the overhead due to the pre-computation is largely preferable to an unfruitful database search.

A different approach to the development of a pictorial query language is undertaken by Papadias & Sellis (1994a, 1995). They propose a Pictorial Query-by-Example (PQBE) language requiring the specification of a symbolic skeleton image rather than a skeleton table. The input query image is symbolic in the sense that it consist of a grid filled with object IDs in such a way that directional relationships are correctly represented. Papadias and Sellis consider nine basic directional relations similar to those of CDC (Goyal & Egenhofer, in press, 1997) and also resort to qualitative reasoning in the form of relation composition when the input query is represented by multiple skeleton images with at least one common object. PQBE provides the possibility of specifying constant and variable objects in skeleton images, as well as the possibility of expressing intersection, union and negation operations. The authors also describe how to use PQBE for purposes different than retrieval: the language provides some special symbols that can be used in skeleton images to denote objects to be inserted, deleted, or updated. The fundamental operation on which retrieval, insertion, deletion, and update rely is obviously the matching of skeleton images against the dataset that is done similarly to Egenhofer's approach: symbolic input images are encoded in a constraint network and subsequently in a spatial query. Also in this case no details are given about the actual query execution strategy.

Chapter 3

# Qualitative Spatial Configuration Queries

In this chapter we introduce, discuss, analyze, and devise solutions to the Qualitative Spatial Configuration Query (QSCQ) problem.

## 3.1 Qualitative Spatial Relation Queries

In Section 2.3.3 spatial queries have been defined and the most common types have been discussed. It has been outlined that, despite their semantic differences, spatial queries share a ground commonality: they all encode spatial predicates that, typically, embody some sort of qualitative spatial relation. Accordingly, the totality of such spatial queries can be generally referred to as *qualitative spatial relation queries*.

Qualitative spatial relation queries can be classified according to the "degree of indetermination" of the spatial predicates involved, namely according to the elements of the spatial predicate that are left unspecified. Let us consider queries involving only one binary predicate of the form

$$(\textit{primary object}, \textit{spatial relation}, \textit{reference object})$$

Moreover, let us assume that the spatial predicate is taken from a set of $\mathsf{B}$ qualitative relations and that the query is defined over a spatial dataset of cardinality $\mathsf{N}$—i.e. containing $\mathsf{N}$ objects. Then, Table 3.1 shows a possible categorization and nomenclature.

Given the typical size $\mathsf{N}$ of a spatial dataset, we can safely assume that $\mathsf{B} \ll \mathsf{N}$. Hence, Table 3.1 also lists the different types of qualitative spatial queries ordered according to the number of checks to be performed to resolve any query. A *relation retrieval* query, for instance, has to be checked against (in the worst case) all the $\mathsf{B}$ available spatial predicates in order to find out the one holding among

| Query Name | Spatial Predicate: Primary Object | Spatial Predicate: Qualitative Relation | Spatial Predicate: Reference Object | Spatial Request | Checks # |
|---|---|---|---|---|---|
| Relation Checking | given | given | given | Does the given relation hold over the given objects? | $\mathcal{O}(1)$ |
| Relation Retrieval | given | ? | given | Which relation does hold over the given objects? | $\mathcal{O}(\mathsf{B})$ |
| Object Retrieval | ? | given | given | Which objects are in the given relation with the given object? | $\mathcal{O}(\mathsf{N})$ |
| | given | given | ? | | |
| Object-Relation Retrieval | given | ? | ? | Which relations do hold between the given object and the rest of the dataset? | $\mathcal{O}(\mathsf{BN})$ |
| | ? | ? | given | | |
| Configuration Retrieval | ? | given | ? | Which objects are arranged according to the given relation? | $\mathcal{O}(\mathsf{N}^2)$ |
| World Snapshot | ? | ? | ? | How are the objects in the dataset arranged? | $\mathcal{O}(\mathsf{BN}^2)$ |

N: dataset cardinality B: relation set cardinality

**Table 3.1:** Classification of qualitative spatial relation queries according to the indetermination level of the spatial predicate and corresponding number of checks necessary to resolve the query.

the specified object pair. At a basic level, without considering possible spatial aspect interdependencies, the number of checks for queries embodying multiple predicates can be easily obtained by multiplying the number of checks necessary to perform for any predicate involved.

The *world snapshot* is the most demanding qualitative spatial relation query type but it does not encode a realistic spatial request. Immediately below, in terms of complexity, it follows the *configuration retrieval* query. It can encapsulate the essence of natural spatial descriptions (cf. Section 1.2.1) and it represents the most demanding qualitative spatial relation query type a database can realistically be queried with.

This work is concerned with the analysis of configuration retrieval queries and with the conception and development of Spatial Access Methods (SAMs) to efficiently solve them. The study is not restricted to single-predicate configurations, rather the focus is on those queries constrained by a series of possibly different spatial predicates, each with a degree of indetermination identical to that of a single-constraint configuration retrieval query. In the reminder of the text, such queries will be referred to with the full-explanatory name Qualitative Spatial Configuration Query (QSCQ).

## 3.2 QSCQ Enablement

It was already pointed out (cf. Sections 2.2.1 and 2.2.7) that humans cope with spatial features mainly in a qualitative and relational manner. Now, it is argued that qualitative spatial queries in general, and QSCQs in particular, are a suitable instrument to encode natural spatial descriptions and, thus, to enable spatial querying by natural language or by sketch.

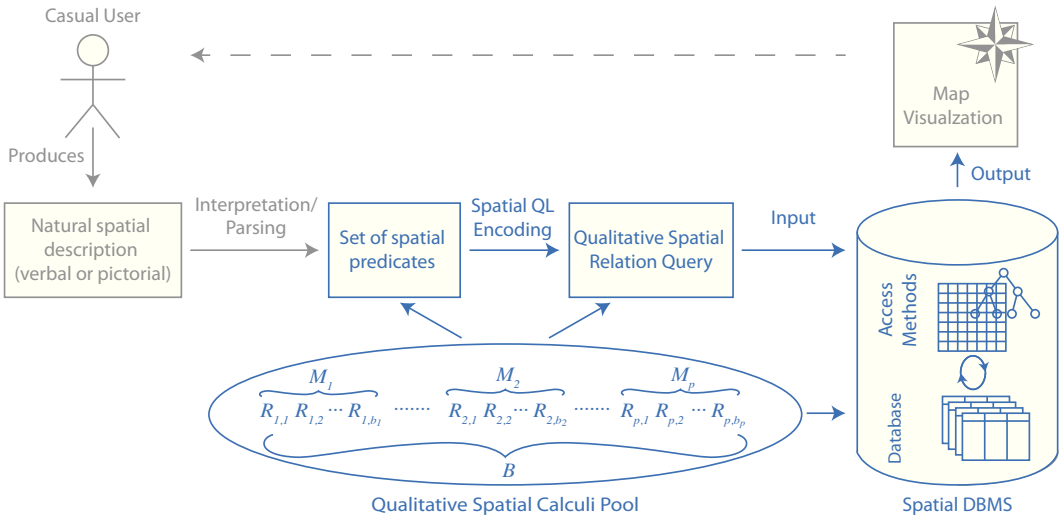



**Figure 3.1:** Natural spatial description queries: transformation flow for automatic interpretation into spatial query language. Gray items fall beyond the scope of this thesis and are assumed to be given.

Figure 3.1 depicts a structured overview of the process a natural spatial description has to undertake to be *automatically* interpreted into a qualitative spatial relation query which can be answered by a spatial database. The spatial description is (*i*) interpreted into a series of predicates taken from a finite set of spatial relations which are subsequently (*ii*) encoded into a qualitative spatial relation query. Lastly, the spatial database (*iii*) resolves the query statement, possibly taking advantage of some SAMs, and outputs the result which, optionally, can be further (*iv*) elaborated—e.g. graphically rendered—before being presented to the user.

The focus of this work is on spatial databases, therefore we will only address steps (*ii*) and (*iii*) in the process described above (blue items in Figure 3.1). In particular, we are interested in determining under which assumptions an automatic processing of spatial descriptions can be done and in conceiving efficient QSCQ solution strategies. In the reminder of this section we will focus on the first objective, whereas a discussion on the resolution approach is delayed to Section 3.4 after having formally defined a QSCQ and its solution.

### 3.2.1 Interpreting Natural Spatial Descriptions

Although, both natural language and image (sketch) interpretation processes are disregarded in this work, we expect their output to be $(i)$ a set of spatial variables constrained by $(ii)$ a set of spatial predicates.

**Example 3.1** (Encoding spatial predicates) **-** The request reported in Example 1.1 can straightforwardly be encoded into a set of spatial predicates in two steps. First, a variable has to be declared for any spatial entity reported in the description:

$a \leftarrow$ Apartment $\quad u \leftarrow$ University $\quad t \leftarrow$ Tram/Bus Stop
$m \leftarrow$ Main Station $\quad s \leftarrow$ Supermarket $\quad p \leftarrow$ Park

Second, every spatial proposition has to be encoded into a qualitative relation:

$$Between(a, m, u) \qquad Close(a, t)$$
$$Close(a, s) \qquad Visible(p, a)$$

### 3.2.2 Spatial Query Language Encoding

As of today, except for topological and metric distance predicates, semantics of spatial predicates is not formally defined in common spatial databases. Accordingly, spatial predicates have to be manually interpreted and Query Language (QL)-encoded by means of the spatial operators available within the underlying spatial Database Management System (DBMS).

**Example 3.2** (QSCQ today) **-** Let us resume again Example 1.1 and let us assume that the spatial dataset is stored in a Relational Database Management System (RDBMS). Furthermore, we assume Figure 3.2 to represent part of the logical schema of the database. Particularly, the table **Dwelling** contains houses and Apartments, the table **Station** maintains both train stations and Tram/Bus Stops while the table **Amenities** stores entities like Supermarkets, universities and Parks.

| **Dwelling** | |
|---|---|
| id | *serial integer* |
| type | *text* |
| name | *text* |
| address | *text* |
| for_rent | *boolean* |
| the_geom | *point* |

| **Station** | |
|---|---|
| id | *serial integer* |
| type | *text* |
| name | *text* |
| the_geom | *geometry* |

| **Amenities** | |
|---|---|
| id | *serial integer* |
| type | *text* |
| name | *text* |
| address | *text* |
| the_geom | *geometry* |

**Figure 3.2:** Logical scheme for the Bremen dataset depicted in Figure 1.1.

Then the request in Example 1.1 can possibly be encoded into a spatially-extended SQL language. The absence of appropriate spatial operators to express all of the required spatial constraints (cf. Section 2.3.2) forces the database user to geometrically interpret the spatial description and to elaborate a cumbersome query statement like the one shown in Listing 3.1. Since no formal semantics is provided for the requested predicates, several assumptions have been made. (*i*) The relation $Between(a, m, u)$ has been considered satisfied by all those Apartments $a$ located within an ellipse, with eccentricity equal to $\frac{2}{3}$, whose foci are $m$ and $u$ (cf. lines 15-19). (*ii*) Similarly, the qualitative distance relation *Close* has been interpreted as "less than 2 units away" (cf. lines 20 and 21), whatever a unit means. (*iii*) Lastly, the visibility constraint has been heavily approximated with the condition that no dwelling lies in between the searched Apartment and a Park (cf. lines 22-28).

**Listing 3.1: Possible SQL encoding of the spatial request in Example 1.1**

```
1  SELECT
2    a.the_geom, m.the_geom, u.the_geom,
3    s.the_geom, t.the_geom, p.the_geom
4  FROM
5    Dwelling AS d,
6    Dwelling AS a, Station AS m, Amenities AS u,
7    Amenities AS s, Station AS t, Amenities AS p
8  WHERE
9    a.type = "Apartment" AND a.for_rent = true
10   AND m.type = "Main Station"
11   AND u.type = "University"
12   AND s.type = "Supermarket"
13   AND t.type = "Tram/Bus Stop"
14   AND p.type = "Park"
15   AND (distance(a.the_geom,m.the_geom) +
16     distance(a.the_geom,u.the_geom))
17     <=
18     (distance(m.the_geom,u.the_geom) +
19     (distance(m.the_geom,u.the_geom)/2))
20   AND distance(a.the_geom,s.the_geom) < 2
21   AND distance(a.the_geom,t.the_geom) < 2
22   AND NOT intersect(
23         set_difference(
24           convex_hull(union(a.the_geom,p.the_geom)),
25           union(a.the_geom,p.the_geom)
26         ),
27         d.the_geom
28       )
```

Example 3.2 blatantly demonstrates that a simple spatial request like the one reported in Example 1.1 can lead to a relatively complicated QL statement. Yet, it has to be noted that the SQL encoding in Listing 3.1 provides an arbitrary interpretation of the original spatial predicates—i.e. other encodings are possible. Lastly, the visibility predicate has been heavily simplified for the sake of clarity whereas a more appropriate encoding would require a Park not to be occluded to the Apartment by any spatial object stored in the dataset.

In order to allow for automatic and unambiguous QL-encoding, formal semantics has to be provided for any spatial predicate to be encoded. Since qualitative spatial calculi provide formally defined qualitative relations, we propose the interpretation of spatial descriptions to result into a set of qualitative spatial relations rather than into generic spatial predicates. Accordingly, we assume there exists a pool of available qualitative spatial calculi from which relations can be picked and that such a pool is large enough to comprehend all the spatial relations needed to generate an exact interpretation of the spatial description.

**Definition 3.1** (Calculus, relation, and arity pool) **-** A *calculus pool* $\boldsymbol{\mathcal{P}} = \{\mathcal{M}_1, \ldots, \mathcal{M}_{\mathsf{p}}\}$ is a set of $\mathsf{p}$ qualitative spatial calculi such that each $\mathcal{M}_i \in \boldsymbol{\mathcal{P}}$ defines a set $\mathcal{B}_i = \{\mathrm{R}_{i,1}, \ldots, \mathrm{R}_{i,\mathsf{b}_i}\}$ of $\mathsf{b}_i$ basic relations of uniform arity $\mathsf{a}_i \geq 2$ ($\mathsf{a}_i \in \mathbb{N}$). Accordingly, the *relation pool* $\boldsymbol{\mathcal{B}} = \{\mathcal{B}_1, \ldots, \mathcal{B}_{\mathsf{p}}\}$ is the set of $\mathsf{B} = \sum_{i=1}^{\mathsf{p}} \mathsf{b}_i$ base spatial relations defined by the qualitative calculi in $\boldsymbol{\mathcal{P}}$. Lastly, the *arity pool* $A \subseteq \mathbb{N}$ is the set of distinct arities of the models in $\boldsymbol{\mathcal{P}}$.

We further assume that the calculus pool $\boldsymbol{\mathcal{P}}$ is also available within the spatial database as a set of boolean-valued functions: one for each $\mathsf{a}$-ary relation $\mathrm{R} \in \boldsymbol{\mathcal{B}}$. Each function is of the form:

$$\mathrm{R}(o_1, \ldots, o_{\mathsf{a}})$$

and returns true if R holds over the spatial object $\mathsf{a}$-tuple $(o_1, \ldots, o_{\mathsf{a}})$. Such functions enrich the set of spatial operators available in the database and allow for $(i)$ automatic QL-encoding and $(ii)$ much leaner query statements.

**Example 3.3** (QSCQ encoding) **-** Let us resume once again Example 1.1 and, again, let us set the assumption that the spatial dataset is stored in a RDBMS whose logical scheme is partly depicted in Figure 3.2. This time we set the further assumption that the spatial predicates generated in Example 3.1 correspond to spatial relations from $\boldsymbol{\mathcal{B}}$. Then, the original spatial description can be automatically SQL-encoded into the QSCQ reported in Listing 3.2. The relations $Between(a, m, u)$, $Close(a, t)$, $Close(a, s)$ and $Visible(p, a)$ are straightforwardly mapped onto a series of operators in the WHERE clause, as shown in Listing 3.2.

Listing 3.2: QSCQ encoding of the spatial request in Example 1.1

```
SELECT
  a.the_geom, m.the_geom, u.the_geom,
  s.the_geom, t.the_geom, p.the_geom
FROM
  Dwelling AS a, Station AS m, Amenities AS u,
  Amenities AS s, Station AS t, Amenities AS p
WHERE
  a.type = "Apartment" AND a.for_rent = true
  AND m.type = "Main Station"
  AND u.type = "University"
  AND s.type = "Supermarket"
  AND t.type = "Tram/Bus Stop"
  AND p.type = "Park"
  AND Between(a,m,u)
  AND Close(a,t)
  AND Close(a,s)
  AND Visible(p,a)
```

## 3.3 The QSCQ Problem

So far we described the challenge of enabling qualitative spatial relation queries in a rather conceptual way to illustrate all involved constituents more clearly. Yet, now that all the components involved in the game have been introduced and an analysis of QSCQ solving procedure is to be carried out, a formalization of the concepts of QSCQ and of its solution is necessary.

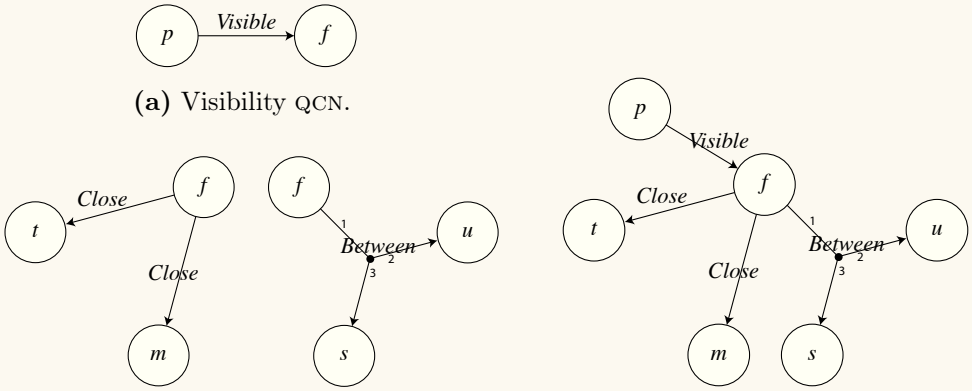
**Definition 3.2** (Qualitative Spatial Configuration Query) **-** Given a calculus pool $\boldsymbol{\mathcal{P}}$ and a spatial dataset $O$. A *Qualitative Spatial Configuration Query* defined over $(O, \boldsymbol{\mathcal{P}})$ is a pair $\mathcal{Q} = (X, \Xi)$ where $X$ is a set of $\mathsf{v}$ variables ranging over $O$ and $\Xi$ is a set of $\mathsf{c}$ spatial constraints. Each constraint $\xi \in \Xi$ is, in turn, a pair $\xi = (\mathcal{R}, \chi)$ where $\mathcal{R} \subseteq \boldsymbol{\mathcal{B}}$ is the set of relations of uniform arity $\mathsf{a}$ defined over the same variable $\mathsf{a}$-sequence $\chi \in X^{\mathsf{a}}$. The relationship between the variable set and the constraint set is expressed by the following rule:

$$X = \{\bigcup_{i=1}^{\mathsf{c}} \chi(\xi_i)\}$$

where $\chi(\xi_i)$ is the spatial variable $\mathsf{a}$-sequence constrained by $\xi_i$

According to such a definition a QSCQ can be represented as a *multi-calculus Qualitative Constraint Network (QCN)* (cf. Section 2.2.5). The variables are the nodes and the constraints the oriented hyperarcs connecting them.

**Example 3.4** (QCN representation) **-** Assuming that the three different spatial predicates in Example 3.1 are relations from different calculi, they can be represented as three distinct QCNs, as depicted in Figures 3.3(a)–(c). Alternatively the multi-calculus QCN shown in Figure 3.3(d) can be generated. Note that for constraints with arity higher than 2 the order in which the variables appear in the sequence is denoted with small numbers near the different branches of the hyperarc.




**(a)** Visibility QCN. **(b)** Distance QCN. **(c)** Position QCN. **(d)** Multi-calculus QCN.

**Figure 3.3:** Spatial predicates from Example 3.1 can be represented either as one multi-calculus Qualitative Constraint Network (QCN) (d) or as three distinct single-calculus QCNs (a), (b) and (c).

In the scope of this work, we only consider inconsistency- and ambiguity-free QSCQs, i.e. the corresponding QCNs can have arc labels composed of several relations provided that each such relation is a base relation from a different calculus and that they do not contradict each other.

A spatial dataset $O$ can also be represented as a multi-calculus QCN with the nodes representing the dataset objects and the arcs representing the relations from $\boldsymbol{\mathcal{B}}$ holding over object $\mathsf{a}$-tuples (for any calculus arity $\mathsf{a}$ in $A$). The latter are obtained operating what we call a $\boldsymbol{\mathcal{P}}$*-qualification* of the spatial dataset.

**Definition 3.3** (Qualified Dataset) **-** Given a spatial dataset $O$, a calculus pool $\boldsymbol{\mathcal{P}}$ containing $\mathsf{p}$ qualitative spatial calculi. The $\boldsymbol{\mathcal{P}}$*-qualified dataset* is the set $\mathcal{R}_{\boldsymbol{\mathcal{P}}}(O) = \{\mathcal{R}_{\mathcal{M}_i}(O) | i = 1, \ldots, \mathsf{p}\}$ whose $i$-th element $\mathcal{R}_{\mathcal{M}_i}(O)$ is in turn a set containing all the relations from the base relation set $\mathcal{B}_i$ of the calculus $\mathcal{M}_i$ induced from the spatial dataset $O$. Any such set is named $\mathcal{M}_i$-qualified dataset.

We stated earlier that solving a QSCQ consists in finding all the sets of objects from the spatial dataset that satisfy the constraint set expressed in the query. Equivalently, we can say that solving a QSCQ consists in finding all occurrences of its QCN representation as a subgraph of the QCN representation of the spatial dataset.

This is a well known problem in graph theory that goes under the name of *subgraph isomorphism* or *subgraph matching*: A graph $G = (V_G, E_G)$ is isomorphic to a subgraph of a graph $H = (V_H, E_H)$, denoted by $G \simeq H_G \subseteq H$, if there exists an injection $f : V_G \to V_H$ such that, for every pair of vertices $v_i, v_j \in V_G$, if $(v_i, v_j) \in E_G$ then $(f(v_i), f(v_j)) \in E_H$. The definition varies slightly for directed, labelled, hypergraphs, that is what multi-calculus QCNs are:

**Definition 3.4** (QSCQ solution) **-** Given a QSCQ $\mathcal{Q} = (X, \Xi)$ defined over $(O, \boldsymbol{\mathcal{P}})$. Given also the functions $r_{O,\mathsf{a}} : O^{\mathsf{a}} \to 2^{\boldsymbol{\mathcal{B}}}$ returning the set of $\mathsf{a}$-ary relations from $\boldsymbol{\mathcal{B}}$ that hold between $\mathsf{a}$-tuples of objects from $O$ and a similar set of functions $r_{X,\mathsf{a}} : X^{\mathsf{a}} \to 2^{\boldsymbol{\mathcal{B}}}$ returning the set of $\mathsf{a}$-ary relations enforced by the constraint set $\Xi$ over $\mathsf{a}$-tuples of variables from $X$. The solution $\Sigma(\mathcal{Q})$ of $\mathcal{Q}$ is the set of all one-to-one mappings $\sigma : X \to O$ such that, the following condition holds:

$$r_{X,\mathsf{a}}(x_{i_1}, \ldots, x_{i_\mathsf{a}}) \subseteq r_{O,\mathsf{a}}(\sigma(x_{i_1}), \ldots, \sigma(x_{i_\mathsf{a}})) \; \forall \; (x_{i_1}, \ldots, x_{i_\mathsf{a}}) \in X^{\mathsf{a}} \; \forall \; \mathsf{a} \in A \tag{3.1}$$

From now on, for the sake of readability, we will refer to the QCN interpretations of a QSCQ and of a spatial dataset with the terms *query graph* and *data graph*, respectively.

## 3.4 Solving QSCQs

The interpretation of QSCQs and spatial datasets as QCNs allows for devising query solving methods grounded on subgraph matching enumeration algorithms (cf. Conte *et al.*, 2004, for a thoroughly literature survey on the subject).

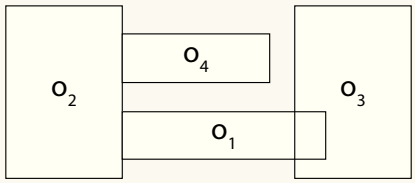
Practically, a subgraph matching consists in assigning each node $x_i$ of the query graph one node $o_j$ of the data graph in such a way that $(i)$ $o_j$ is not assigned to multiple $x_i$ and $(ii)$ the isomorphism condition is satisfied. For directed arc-labeled hypergraphs such a condition is the one in Equation 3.1.

If the query graph has $\mathsf{v}$ nodes and the data graph has $\mathsf{N}$ nodes, there are as many possible node assignments as the number of $\mathsf{v}$-permutations (without repetitions) of $\mathsf{N}$ elements: $\frac{\mathsf{N}!}{(\mathsf{N}-\mathsf{v})!}$. Possibly, all such assignments can be represented by means of a tree structure: The tree has as many levels as the number of nodes in the query graph and each tree node corresponds to an assignment $x_i \sim o_j$. Tree-nodes at $i$-th level of the tree assign the $i$-th query graph node each data graph node which has not been assigned in any ancestor tree-node. The root of the tree is located at level 0 and corresponds to the empty assignment $\nsim$. Accordingly, a path from the root to a leaf represents a complete assignment, but is not guaranteed to be a matching.

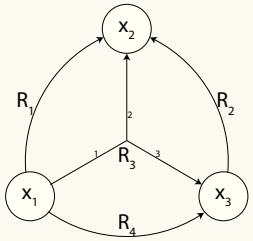
**Example 3.5 -** Let us assume to have a spatial dataset $O$ as in Figure 3.4(b) which we want to query by the QSCQ in Figure 3.4(a) represented by the query graph in Figure 3.4(c) Let us also assume to have a calculus pool $\boldsymbol{\mathcal{P}} = \{\mathcal{M}_1\, \mathcal{M}_2\}$ such that $\mathcal{B}_1 = \{\mathrm{R}_1, \mathrm{R}_2, \mathrm{R}_4, \cdots\}$ and $\mathcal{B}_2 = \{\mathrm{R}_3, \cdots\}$. The data graph $(O, \mathcal{R}_{\boldsymbol{\mathcal{P}}}(O))$, whose arc set is the $\boldsymbol{\mathcal{P}}$-qualification of $O$, is depicted in Figure 3.4(d) where, for the sake of readability, arcs unimportant for the scope of this example are reported in light grey. Enumerating the subgraph matchings, then, is equivalent to finding all the root-leaf paths in the assignment tree of Figure 3.4(e) which satisfy Equation 3.1.

$$\left( \left\{ \begin{array}{l} x_1 \\ x_2 \\ x_3 \end{array} \right\}, \left\{ \begin{array}{l} \xi_1 : (\mathrm{R}_1, (x_1, x_2)) \\ \xi_2 : (\mathrm{R}_2, (x_3, x_2)) \\ \xi_3 : (\mathrm{R}_3, (x_1, x_2, x_3)) \\ \xi_4 : (\mathrm{R}_4, (x_1, x_3)) \end{array} \right\} \right)$$

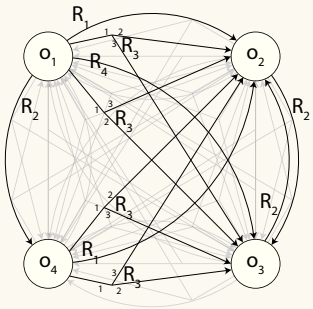
**(a)** QSCQ.



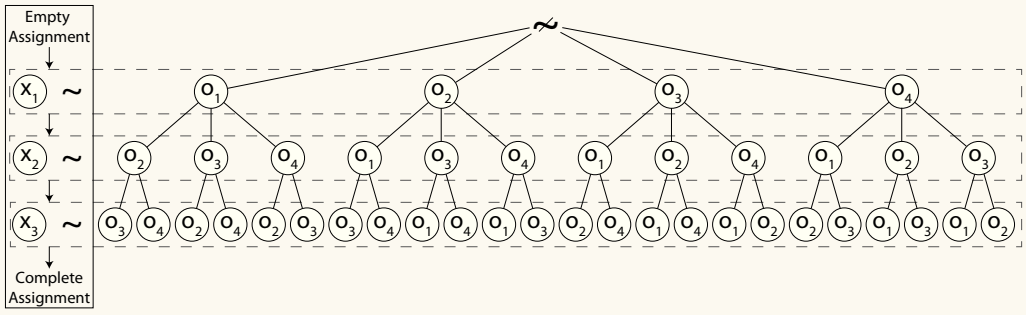

**(b)** Spatial Dataset.




**(c)** Query graph.




**(d)** Data graph.




**(e)** Tree of possible assignments.

**Figure 3.4:** Querying the spatial dataset in (b) by the Qualitative Spatial Configuration Query (QSCQ) in (a) is equivalent to find all subgraph matchings occurring between the query graph (c) and the data graph (d) associated to the spatial dataset. In turn, this is equivalent to find all the root-leaf paths within the possible assignment tree (e) that satisfy the subgraph matching condition in Equation 3.1.

One class of (sub)graph matching enumeration algorithms draws upon a tree search with backtracking to identify all the possible matchings. The first algorithm of this kind—and still one of the most used ones—is described in (Ullmann, 1976). Beyond the backtracking tree search, the algorithm employs a forward checking to early detect unfruitful assignment tree paths. Forward checking draws upon node adjacency: the assignment $x_i \sim o_j$ can be part of a complete matching only if $(i)$ $o_j$ has at least as many adjacent nodes as $x_i$ and $(ii)$ for each adjacent node $x_k$ of $x_i$ there exists at least one node adjacent to $o_j$ that can be assigned to $x_k$. Obviously, the smaller is $i$, the higher is the subtree rooted in $x_i$ and the more numerous are the paths that we can avoid to visit (cf. Figure 3.4(e) for a visual verification). Ullmann's algorithm was originally designed for undirected, unlabeled graphs but its basic strategy lends itself very well to be adapted for other graph types.

### 3.4.1 Basic Retrieval Strategy

For the kind of graphs we cope with, a forward checking based on node adjacency does not provide an efficient solution: The data graph will always be a complete graph with a number of nodes and arcs much greater than that of the query graph. Therefore, most of the times an assigned data graph node will have enough adjacent nodes to allow for further assignments.

In (Wallgrün *et al.*, 2010), a variant of Ullmann's algorithm is used to find the best possible matching between a pair of QCNs representing Volunteered Geographic Information (VGI) sources. The authors make our same point that, usually, volunteered spatial information comes in the form of descriptions or depictions and, accordingly, investigate the possibility of doing the matching at a qualitative constraint level. Although apparently very similar to the QSCQ problem, the problem tackled by Wallgrün *et al.* differs in two main points. $(i)$ They look for the best partial matchings rather than for exact matchings: they search for partial matchings with the highest number of constraints being satisfied. This means that they are faced with a maximum common subgraph problem instead of a subgraph isomorphism. $(ii)$ They also consider ambiguous QCNs, i.e. arc labels may contain disjunctive relations. Accordingly, they employ an $A^*$ algorithm to drive the tree search and consistency checking as main constituent of the forward checking function.

Our approach also stems from Ullmann's algorithm and exploits the matching condition in Equation 3.1 which enforces both a structural and a semantic matching. The structural part concerns the conformation of the arc set: If there exists a hyperarc among an $\mathsf{a}$-tuple of $x$ nodes, it has to exist a hyperarc of the same arity among the matched nodes of the data graph. Moreover (semantic matching), the label of the data graph hyperarc has to consist of a set of relations that contains those occurring in the query graph hyperarc. If this condition holds on a complete assignment, it also has to hold on any partial assignment constituting a complete one.

The retrieval strategy we developed tries to build the matchings incrementally extending the empty one. Basically, it is a breadth-first search driven by the arc structure of the query graph with a forward checking on the arc labels. The search proceeds by $e$ levels at once, where $e$ is the number of nodes of the analyzed arc that have not been assigned yet. While the search proceeds, the forward checking rules out all the partial assignments that do not match with the query-subgraph induced by the arcs analyzed so far.

Algorithm 3.1 reports the algorithmic realization. We assume that an input QSCQ $\mathcal{Q} = (X, \Xi)$, with $\Xi = \{\xi_1, \ldots, \xi_{\mathsf{c}}\}$, is defined over the usual pair $(O, \boldsymbol{\mathcal{P}})$. A (partial) matching is denoted by a pair $(\widetilde{\chi}, \widetilde{\omega})$ where $\widetilde{\chi}$ is a sequence without repetitions of variables from $X$ and $\widetilde{\omega}$ is a similar sequence of objects from $O$ such that the $j$-th element $\widetilde{\omega}[j]$ of $\widetilde{\omega}$ is assigned to the $j$-th variable $\widetilde{\chi}[j]$ in $\widetilde{\chi}$. All sequences $\widetilde{\omega}$ which satisfy the first $i$ constraints in $\Xi$ are maintained in the set $\Omega_i$.

**Algorithm 3.1** RETRIEVE: Solves a given QSCQ $\mathcal{Q}$ via subgraph matching

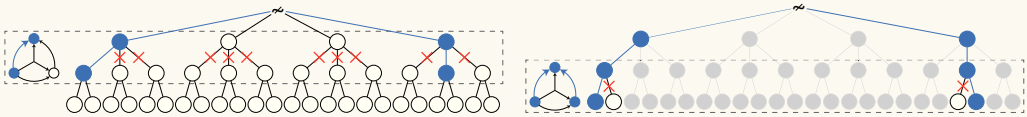
**Input:**
$\mathcal{Q} = (X, \Xi)$: QSCQ
**Output:**
$\Sigma(\mathcal{Q})$: solution of the given QSCQ

```
1:  function RETRIEVE(Q)
2:      Ω_0 ← {()};                                   ▷ initial partial matching
3:      χ̃ ← ();                                      ▷ initialize list of assigned variables
4:      for 1 ≤ i ≤ c do
5:          Ω_i ← ∅;                                  ▷ initialize extended partial matching
6:          for all ω̃ ∈ Ω_{i−1} do
7:              Ω_i ← Ω_i ∪ EXTENDPMATCHING(χ̃, ω̃, ξ_i);
8:          if Ω_i = ∅ then return (() , ∅);          ▷ no matching found!
9:          χ̃ ← χ̃ ⊎ χ(ξ_i);                          ▷ update list of assigned variables
10:     return (χ̃, Ω_c);
```

The algorithm starts (Lines 2-3) with the empty matching $\nsim$ represented by the set $\Omega_0$, containing an empty sequence of dataset objects, and an empty variable sequence $\widetilde{\chi}$. For any constraint $\xi_i \in \Xi$ and for any $\widetilde{\omega} \in \Omega_{i-1}$, the algorithm tries to extend the partial matching $(\widetilde{\chi}, \widetilde{\omega})$ compatibly with $\xi_i$ and stores the results in $\Omega_i$ (Line 7). If the tentative matching extension fails for every $(\widetilde{\chi}, \widetilde{\omega})$—i.e. yields $\Omega_i = \emptyset$—then no (partial) assignment satisfying the first $i-1$ constraints satisfies the first $i$ constraints. Accordingly, no complete matching exists and the algorithm terminates early. Otherwise, before proceeding to the next constraint, the algorithm extends the sequence of matched variables $\widetilde{\chi}$ by the variables of the current constraint that have not been matched before (Line 9). We assume this is done by the operator $\uplus$ that concatenates two sequences of symbols yielding a sequence without repetitions.

The matching expansion is assumed to be done by a function ExtendPMatching that takes as input a partial matching $(\widetilde{\chi}, \widetilde{\omega})$ and a constraint $\xi = (\mathcal{R}, \chi)$ of arity $\mathsf{a}$. Basically the function checks the set $\mathcal{R}_\omega \subseteq \boldsymbol{\mathcal{B}}$ of relations holding over any dataset object $\mathsf{a}$-tuple $\omega \in O^{\mathsf{a}}$ against the relation set $\mathcal{R}$ required by the constraint. Any $\omega$ such that $\mathcal{R}_\omega \supseteq \mathcal{R}$ is merged with the sequence $\widetilde{\omega}$ and returned to the main procedure. Note that the number of object $\mathsf{a}$-tuples to be considered depends on the arity $\mathsf{a}$ of the constraint as well as on the number $\mathsf{e}$ of constrained variables that have not been assigned yet; it is equal to $(\mathsf{N} - (\mathsf{a} - \mathsf{e}))^{\mathsf{e}}$.

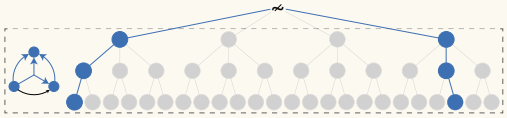
**Example 3.6 -** The application of Algorithm 3.1 to the Example 3.5 is depicted in Figure 3.5. Each subfigure shows how the iteration over a certain arc of the query graph (reported to the left) determines partial matchings (blue paths in the search tree). Assignments that do not satisfy the matching condition are eliminated (red crosses) from the search tree and reported in grey in the next subfigure. Note that, the first two constraints, reported in Figures 3.5(a)-(b), yields two complete assignments that also satisfy the third constraint, in Figure 3.5(c). Lastly, the analysis of the last constraint, in Figure 3.5(d), allows for the removal of one of the partial matchings and yields the final result: $(\widetilde{\chi}, \Omega_4) = ((x_1, x_2, x_3), \{(o_1, o_2, o_3)\})$.



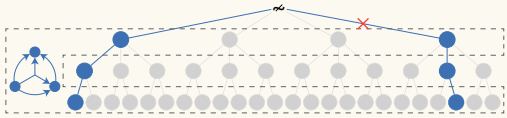
**(a)** $\xi_1 : (\mathrm{R}_1, (x_1, x_2))$.



**(b)** $\xi_2 : (\mathrm{R}_2, (x_3, x_2))$.



**(c)** $\xi_3 : (\mathrm{R}_3, (x_1, x_2, x_3))$.



**(d)** $\xi_4 : (\mathrm{R}_4, (x_1, x_3))$.

**Figure 3.5:** Matching the query graph in Figure 3.4(c) against the data graph in Figure 3.4(d) by Algorithm 3.1. The algorithm explores the search tree in Figure 3.4(e) driven by the arc structure of the query graph.

## 3.5 Dataset Qualification

In the previous section we argued that solving QSCQs is equivalent to enumerating the subgraph matchings between a query graph and the data graph. We silently assumed that both such graphs are given and we focused on the basic matching procedure.

However, it has to be considered that a real geographic dataset usually consists of millions of geometric objects (the nodes of the data graph) but does not come

with spatial relations (the arcs of the data graph) explicitly represented. Thus, in a realistic scenario, such relations have to be computed from the dataset objects. Given the typical size of real geographic datasets, this is a very costly operation that has to be carefully accounted in the retrieval process.

### 3.5.1 Runtime Qualification

The most straightforward approach for dataset qualification consists in qualifying the spatial dataset while proceeding in the matching process. For this purpose we use the operators available in the spatial DBMS to compute qualitative relations at runtime.

It is worth recalling that in Section 3.2.2 we made the assumption that the standard spatial operator set present in a DBMS is complemented by a set of Boolean-valued functions: one for any relation in the relation pool $\boldsymbol{\mathcal{B}}$.

**Algorithm 3.2** ComputeRelation$_{\mathcal{M}}$: Computes which of the relations defined in $\mathcal{M}$ holds over an $\mathsf{a}$-tuple of spatial objects

**Input:**
$\omega$: $\mathsf{a}$-tuple of spatial objects
**Output:**
R: the relation from $\mathcal{M}$ holding over the input $\mathsf{a}$-tuple

```
1: function ComputeRelation_M(ω)
2:     for all R ∈ B do
3:         if R(ω) then                          ▷ checks if ω ∈ R
4:             return R;
```

From now on, we further assume that for any $\mathsf{a}$-ary calculus $\mathcal{M} \in \boldsymbol{\mathcal{P}}$ in the calculus pool there exists a function ComputeRelation$_{\mathcal{M}}$ which takes as input an $\mathsf{a}$-tuple of dataset objects $\omega$ and returns the base relation $\mathrm{R} \in \mathcal{B}$ holding over them. A general implementation for any such function is reported in Algorithm 3.2 and consists in checking $\omega \in O^{\mathsf{a}}$ against any base relation defined by the calculus $\mathcal{M}$ until the relation holding on the object $\mathsf{a}$-tuple is found. The check is done via the boolean functions we recalled above. If each such function runs in constant time and $|\mathcal{B}| = \mathsf{b}$, relation computation executes in $\mathcal{O}(\mathsf{b})$ time.

Given that any calculus in the pool provides a set of jointly exhaustive and pairwise disjoint (JEPD) base relations, one, and only one, relation from $\mathcal{B}$ can hold over any dataset object $\mathsf{a}$-tuple. Accordingly, the cardinality of the $\mathcal{M}$-qualified dataset $\mathcal{R}_{\mathcal{M}}(O)$ equals the number of dataset object $\mathsf{a}$-tuples

$$|\mathcal{R}_{\mathcal{M}}(O)| = \mathsf{N}^{\mathsf{a}} \tag{3.2}$$

and $\mathcal{M}$-qualification runs in time proportional to the number of relation checks:

$$\mathcal{O}\left(\mathsf{b}\mathsf{N}^{\mathsf{a}}\right) \tag{3.3}$$

Given that, in the worst case, a QSCQ involves relations from any calculus in the pool $\boldsymbol{\mathcal{P}}$, runtime qualification affects the time of *each* QSCQ retrieval by a factor:

$$\mathcal{O}\left(\sum_{\mathcal{M}\in\boldsymbol{\mathcal{P}}} \mathsf{b}(\mathcal{M})\mathsf{N}^{\mathsf{a}(\mathcal{M})}\right) \tag{3.4}$$

where $\mathsf{b}(\mathcal{M})$ and $\mathsf{a}(\mathcal{M})$ stand for the number of base relations defined in $\mathcal{M}$ and the arity of the calculus, respectively. Such a temporal cost is definitely unfeasible when considering real geospatial datasets, even for a small calculus pool.

Clementini *et al.* (1994), showed that for the 9-Intersection Model (9-IM) (Egenhofer, 1989, 1991) the relation computation time can be drastically reduced by performing the checks according to the relation-occurrence-frequency order. However, such heuristics only works for models with highly unbalanced relation-occurence-frequencies, e.g. nearly 90% of pairs of objects from a real geospatial dataset are *Disconnected*.

A further drawback of this approach is that it does not allow for exploiting and integrating purely qualitative information into the system, that is, spatial information directly coming in qualitative terms rather than computed from a quantitative dataset—e.g. the kind of information that can be derived from natural spatial descriptions.

### 3.5.2 Pre-qualification

A typical approach used in computer science to cope with intensive computational tasks consists in resorting to Lookup Tables (LUTs): data structures designed to accommodate the results of a certain computation, which allow to replace the runtime calculation with a faster lookup operation.

Accordingly, a different approach to tackle the dataset qualification consists in extending the underlying spatial DBMS with a qualitative storage layer, namely enriching the database schema by some LUTs dedicated to accommodate qualitative spatial relations. We call such tables *relation tables*. This solution achieves the double effect of (*i*) reducing QSCQ retrieval time and (*ii*) enabling the DBMS to deal with a mix of qualitative and geometric information pieces.

Having a place to store qualitative relations allows for $\boldsymbol{\mathcal{P}}$-qualifying the spatial dataset only once. Moreover, the relation tables can be indexed with standard techniques—i.e. B-tree (Comer, 1979) or hashing (Fagin *et al.*, 1979; Litwin, 1980)—allowing for sub-linear access times. Assuming we employ an index which guarantees logarithmic access time, the computational overhead for QSCQ retrievals reported in Equation 3.4 becomes:

$$\mathcal{O}\left(\sum_{\mathcal{M}\in\boldsymbol{\mathcal{P}}} \log\left(\mathsf{N}^{\mathsf{a}(\mathcal{M})}\right)\right) \tag{3.5}$$

The downside of this approach lies in the amount of extra storage space required to store the qualified dataset; according to Equation 3.2 it is proportional to:

$$|\mathcal{R}_{\boldsymbol{\mathcal{P}}}(O)| = \sum_{\mathcal{M}\in\boldsymbol{\mathcal{P}}} |\mathcal{R}_{\mathcal{M}}(O)| = \sum_{\mathcal{M}\in\boldsymbol{\mathcal{P}}} \mathsf{N}^{\mathsf{a}(\mathcal{M})} \tag{3.6}$$

Moreover, similarly to the previous approach, the time required for the prequalification depends on the cardinality $\mathsf{N}$ of the spatial dataset and on the arity and the number of relations in $\boldsymbol{\mathcal{P}}$. Although the latter ones are very small with respect to the former, the dependency from the dataset cardinality represents an important issue when dealing with real geographic datasets.

**Example 3.7 -** Let us consider the Bremen OpenStreetMap[1] dataset—a small dataset for geographic standards—with $\mathsf{N} \approx 7 \times 10^4$ objects. Let us assume that the calculus pool $\boldsymbol{\mathcal{P}} = \{\text{CDC}\}$ only consists of the Cardinal Direction Calculus (CDC) (Goyal & Egenhofer, in press, 1997) and that the spatial dataset contains only simple regions. CDC is a binary calculus—i.e. $\mathsf{a}(\text{CDC}) = 2$—which for simple regions, only defines $\mathsf{b}(\text{CDC}) = 218$ JEPD base relations. Then, according to Equation 3.6, the qualified dataset $\mathcal{R}_{\boldsymbol{\mathcal{P}}}(Bremen)$ contains a number of relations in the range of $(7 \times 10^4)^2 = 4.9 \times 10^9$ and, according to Equation 3.3, the qualification process can require up to approximately $218 \cdot (7 \times 10^4)^2 = 1.06 \times 10^{12}$ relation checks.

A final drawback of having such a big amount of extra stored information regards the maintenance of consistency in the dataset. For example, the removal or the update of a spatial object has to be propagated along the whole qualified dataset which, given the size of the latter, may be a very costly operation.

## 3.6 Summary and Focus

In this chapter we defined a generalization for spatial predicate queries that allows for looking at them as a single query type. We classified them according to the level of indeterminacy of the encoded spatial predicates and identified the class hardest to solve: Qualitative Spatial Configuration Queries (QSCQs).

We argued that, if the semantics of spatial predicates is described formally and available in a DBMS, qualitative spatial queries represent a valuable means to automatically encode natural spatial descriptions. Therefore they are a key element for the design of more natural human-Geographic Information System (GIS) interfaces: qualitative spatial queries can be used for both, contributing information into a spatial database as well as "questioning" it. We raised the point that qualitative spatial calculi provide the necessary semantics and that verbal (and/or pictorial) spatial descriptions can be interpreted into a series of

[1] www.openstreetmap.org/

spatial predicates. Then we assumed to have available a pool of qualitative spatial calculi providing as many relations as necessary to capture the semantics of all the spatial predicates resulting from the interpretation of any natural spatial description. We concluded that, having such a calculus pool encoded as a series of operators in a spatial DBMS allows for a straightforward and automatic QL-encoding of natural spatial descriptions.

Drawing upon this assumption, we focused on QSCQ solving. We provided a formal definition and noted that a QSCQ can be represented as a multi-calculus QCN which we named query graph. Similarly a spatial dataset can also be seen as a huge QCN: the data graph. Accordingly, we showed that solving a QSCQ is equivalent to enumerate all the subgraph isomorphisms occurring between the two networks and we designed a simple graph matching procedure which exploits properties of this kind of graphs.

Matching algorithms have to rely on the arc set—i.e. qualitative relations—of both graphs and we pointed out that assuming the data graph to be given is a too strong hypothesis because of the following motivations. (*i*) Usually spatial datasets come in a geometric representation, i.e. qualitative relations occurring on the geometric set are not explicit. (*ii*) Given the typical cardinality of spatial datasets, the qualification process is a very costly operation which heavily affects the overall matching time and, therefore, cannot be disregarded in QSCQ-retrievals. Accordingly, we presented two basic approaches to tackle the dataset qualification problem. Runtime qualification is straightforwardly realizable but highly inefficient since it requires to compute relations occurring over the whole dataset at any QSCQ-retrieval. The pre-qualification allows for qualifying the dataset only once and even for indexing the resulting qualified dataset. Accordingly this constitutes the best choice to maintain reasonable retrieval times but, on the other hand, requires an enormous quantity of extra storage space.

In agreement with this summary, the remainder of this work is focused on two main investigations: (*i*) How dataset qualification can be done to obtain a good tradeoff between storage space and QSCQ-retrieval time. (*ii*) How the assumption that the calculus pool contains enough qualitative calculi can be realized.

Efficient execution of spatial queries is achieved in spatial DBMS through the exploitation of so-called Spatial Access Methods (SAMs) (cf. Section 2.3.4). In Chapter 4, we design a series of SAMs tailored for solving QSCQs; we term them Qualitative Spatial Access Methods (QSAMs). They draw upon a combination of LUTs and Qualitative Spatial Representation and Reasoning (QSR) techniques to maintain a good trade-off between the space occupied by the dataset qualification and QSCQ retrieval time.

The variety of expressions occurring in natural spatial descriptions does not allow for deciding in advance which spatial relations will be necessary for the encoding: Neither one can predict what spatial aspects—i.e. topological vs. directional—will be reported, nor which type of spatial calculi will be more suitable for the encoding—i.e. cardinal vs. relative directions. Consequently, the assumption that the calculus pool has to provide enough qualitative relations

to encode any spatial description, is better realized allowing for an extendable pool. This shift of perspective calls for a practical approach: In Chapter 5 a novel prototypical software framework is developed that provides a solution for a readily integration of qualitative spatial calculi as well as a development and benchmarking tool for QSAMs.

Chapter 4

# Qualitative Spatial Access Methods

In Section 3.4 we designed a basic strategy for solving a Qualitative Spatial Configuration Query (QSCQ) that is basically a subgraph isomorphism procedure. The graphs to be matched are Qualitative Constraint Networks (QCNs) representing the given QSCQ and the spatial dataset being queried.

In a user-computer interaction scenario, a QSCQ results from the interpretation of a spatial description produced by a human being. In a realistic case, it is licit to assume that such a description does not convey more than a few tens of qualitative spatial constraints. In this work, we do not cope with the interpretation process and assume that a spatial description comes already interpreted into a QSCQ.

In contrast, spatial datasets typically consist of millions of geometric objects, that is, the arcs of the QCN representing the dataset are not explicitly available. Thus a *dataset qualification* (cf. Definition 3.3) has to be performed in order to be able to resolve a QSCQ. Given the typical size of a spatial dataset, a full dataset qualification is a very costly operation that have to be carefully taken into account in the QSCQ solving strategy.

In Sections 3.5.1 and 3.5.2 two basic strategies for the dataset qualification have been analyzed: runtime qualification and pre-qualification. The first one does not require any extra storage space to be used but is highly inefficient for what concerns the retrieval time. The second one calls for a database schema extension—named *qualitative storage layer*—where the qualified dataset can be stored explicitly. This allows for fast retrievals but has two main drawbacks: (*i*) It requires an enormous quantity of extra storage space. (*ii*) Given that the temporal cost of a full pre-qualification is extremely high, such a solution can hardly manage real world dynamism. That is, by the time the pre-computation is done, the changes that have occurred, for instance, in a city and reported in the corresponding spatial dataset are not reflected in the qualified dataset, calling for a new qualification.

In this chapter we propose an alternative to improve the above-mentioned approaches. The main underlying idea is to extend the spatial database with a qualitative storage layer but only pre-compute a targeted subset of the whole qualified dataset. The missing relations have to be rebuildable at retrieval time, ideally without resorting to a full dataset qualification. For doing this we suggest to resort to a relation reduction/reconstruction paradigm to be applied in the qualification and retrieval phase.

Reduction and reconstruction operations can be directly encoded within Spatial Access Methods (SAMs) specifically designed to optimize QSCQ executions:

**Definition 4.1** (Qualitative Spatial Access Method) **-** Given a spatial dataset $O$ and a calculus pool $\boldsymbol{\mathcal{P}}$, a *Qualitative Spatial Access Method (*QSAM*)* over $(O, \boldsymbol{\mathcal{P}})$ is a triple $(q, \mathcal{D}, r)$ such that $q : (O, \boldsymbol{\mathcal{P}}) \to \mathcal{D}$ is a *dataset qualifier function* which, given the spatial dataset, populates the set of data structures $\mathcal{D}$ underlying the QSAM. $r : (\mathcal{D}, \boldsymbol{\mathcal{P}}) \to \Sigma(\mathcal{Q})$ is a *retriever function* that exploits the information stored in $\mathcal{D}$ to solve a given QSCQ.

A QSAM is characterized by a reduction and a reconstruction strategy that determine the main features of the qualifier and retriever functions, as well as the constitution of the data structure set.

In the remainder of this chapter four different typologies of QSAMs are presented together with a discussion on the provided space-time tradeoff. In Sections 4.1 and 4.2 two QSAMs are developed that embody the basic strategies of runtime qualification and pre-qualification, respectively. Section 4.3 is dedicated to the design of a QSAM family drawing upon a *spatial clustering reduction strategy*, whereas in Section 4.4 is devised a solution whose main idea is to use standard Qualitative Spatial Representation and Reasoning (QSR) techniques to rebuild missing relations. Finally, Section 4.5 concludes the chapter with a summary.

## 4.1 Functional QSAM

According to Definition 4.1, the runtime qualification strategy presented in Section 3.5.1 is a QSAM of the form

$$(-, -, \textsc{Retrieve-Functional})$$

where the qualifier function $q$ is void as well as the set $\mathcal{D}$ of data structures. The algorithmic realization of the retriever function, Retrieve-Functional, is identical to the one in Algorithm 3.1 with the function ExtendPMatching realized as in Algorithm 4.1.

The function fetches all the necessary information about the constraints, i.e. the set of enforced relations $\mathcal{R}$, the variable sequence $\chi$, the arity $\mathsf{a}$, and the set of calculi $\boldsymbol{\mathcal{P}}_\xi$ which the relations in $\mathcal{R}$ belong to. Then, it iterates over all the dataset object $\mathsf{a}$-tuples compatible with the input—cf. Example 4.1. If the relations $\mathcal{R}_\omega$ from $\boldsymbol{\mathcal{P}}_\xi$ holding over the object $\mathsf{a}$-tuple $\omega$ at hand equal those in

$\mathcal{R}$, $\omega$ is merged with the sequence $\widetilde{\omega}$ of objects already matched and is added to the set of extended matchings $\Omega_\xi$.

**Example 4.1 -** Let us assume that the variables $\widetilde{\chi} = (x_1, x_2)$ are already assigned the objects $\widetilde{\omega} = (o_5, o_7)$ and that the constraint $\xi$ is defined over the variables $\chi = (x_2, x_3)$. Then, only object pairs $(o_i, o_j)$ with $o_i = o_7$ are *compatible* with the input.

Note that the number of object $\mathsf{a}$-tuples to be considered depends on the arity $\mathsf{a}$ of the constraint as well as on the number $e$ of constrained variables that have not been assigned yet; it is equal to $(\mathsf{N} - (\mathsf{a} - e))^e$ where $\mathsf{N}$ is the cardinality of the spatial dataset $O$.

**Algorithm 4.1** ExtendPMatching-Functional: Extends the partial matching $(\widetilde{\chi}, \widetilde{\omega})$ according to constraint $\xi$

**Input:**
$(\widetilde{\chi}, \widetilde{\omega})$: partial matching
$\xi = (\mathcal{R}, \chi)$: constraint to satisfy
**Output:**
$\Omega_\xi$: set of matchings extended from $(\widetilde{\chi}, \widetilde{\omega})$ and satisfying constraint $\xi$

1: **function** ExtendPMatching-Functional($\widetilde{\chi}, \widetilde{\omega}, \xi$)
2: $\Omega_\xi \leftarrow \emptyset$; ▷ initialize constraint matching set
3: $(\mathcal{R}, \chi) \leftarrow \xi$; ▷ get the relation set and the variable sequence of $\xi$
4: $\mathsf{a} \leftarrow |\chi|$; ▷ get the arity of $\xi$
5: $\boldsymbol{\mathcal{P}}_\xi \leftarrow \emptyset$ ▷ initialize set of calculi for $\xi$
6: **for all** R $\in \mathcal{R}$ **do**
7: $\boldsymbol{\mathcal{P}}_\xi \leftarrow \boldsymbol{\mathcal{P}}_\xi \bigcup \mathcal{M}(\text{R})$; ▷ add to $\boldsymbol{\mathcal{P}}_\xi$ the calculus R belongs to
8: **for all** $\omega \in O^{\mathsf{a}} | \omega[i] = \widetilde{\omega}[i]\ \forall\ i$ s.t. $\chi[i] \in \widetilde{\chi}$ **do** ▷ for each compatible $\omega$
9: $\mathcal{R}_\omega \leftarrow \emptyset$; ▷ initialize the set of relations holding on $\omega$
10: **for all** $\mathcal{M} \in \boldsymbol{\mathcal{P}}_\xi$ **do**
11: $\mathcal{R}_\omega \leftarrow \mathcal{R}_\omega \bigcup$ ComputeRelation$_{\mathcal{M}}(\omega)$;
12: **if** $\mathcal{R}_\omega = \mathcal{R}$ **then**
13: $\Omega_\xi \leftarrow \Omega_\xi \cup \{\widetilde{\omega} \uplus \omega\}$;
14: **return** $\Omega_\xi$;

## 4.2 Qualitative Storage Layer QSAM

The pre-qualification approach presented in Section 3.5.2 is a QSAM of the form

$$(\text{Qualify-QSL}, \mathcal{D}_{\text{QSL}}, \text{Retrieve-QSL})$$

The set of data structures $\mathcal{D}_{\text{QSL}} = \{\text{RT}_i | \forall\ i \in A\}$ is a set of $|A|$ Lookup Tables (LUTs) named *relation tables* where $A$ is the arity pool (cf. Definition 3.1). The

$i$-th relation table $\text{RT}_i$ contains entries of the form

$$(\text{R}_\omega, \omega)$$

where $\omega$ is an $i$-tuple of dataset objects and $\text{R}_\omega$ is an $i$-ary relation from the relation pool $\boldsymbol{\mathcal{B}}$.

The qualifier function, Qualify-QSL, $\boldsymbol{\mathcal{P}}$-qualifies the whole spatial dataset, accommodating each relation in the appropriate relation table according to its arity. The algorithmic realization of Qualify-QSL is reported in Algorithm 4.2.

**Algorithm 4.2** Qualify-QSL: Populates the relation tables in $\mathcal{D}$ with the relations induced by the spatial dataset

**Input:**
$O$: Spatial set
$\boldsymbol{\mathcal{P}}$: Calculus pool
**Output:**
LUTs in $\mathcal{D}$ are filled with relations from the $\boldsymbol{\mathcal{P}}$-qualified dataset $\mathcal{R}_{\boldsymbol{\mathcal{P}}}(O)$

1: **function** Qualify-QSL($O$,$\boldsymbol{\mathcal{P}}$)
2: **for all** $\mathcal{M} \in \boldsymbol{\mathcal{P}}$ **do**
3: $\mathsf{a} \leftarrow \mathsf{a}(\mathcal{M})$; ▷ get the arity of the calculus at hand
4: **for all** $\omega \in O^{\mathsf{a}}$ **do**
5: $\text{R}_\omega \leftarrow \text{ComputeRelation}_{\mathcal{M}}(\omega)$;
6: $\text{RT}_{\mathsf{a}} \leftarrow \text{RT}_{\mathsf{a}} \bigcup (\text{R}_\omega, \omega)$;

Also in this case, the retriever function, Retrieve-QSL, is identical to the one in Algorithm 3.1 with the function ExtendPMatching realized as in Algorithm 4.3: It gets the arity $\mathsf{a}$ of the constraint $\xi$ and calls the function Select that retrieves from the relation table $\text{RT}_{\mathsf{a}}$ all the entries $(\text{R}_\omega, \omega)$ with $\text{R}_\omega \subseteq \mathcal{R}$ and the object $\mathsf{a}$-tuple compatible with the partial matching $(\widetilde{\chi}, \widetilde{\omega})$ in input.

**Algorithm 4.3** ExtendPMatching-QSL: Extends the partial matching $(\widetilde{\chi}, \widetilde{\omega})$ according to constraint $\xi$

**Input:**
$(\widetilde{\chi}, \widetilde{\omega})$: partial matching
$\xi = (\mathcal{R}, \chi)$: constraint to satisfy
**Output:**
$\Omega_\xi$: set of matchings extended from $(\widetilde{\chi}, \widetilde{\omega})$ and satisfying constraint $\xi$

1: **function** ExtendPMatching-QSL($\widetilde{\chi}, \widetilde{\omega}, \xi$)
2: $(\mathcal{R}, \chi) \leftarrow \xi$; ▷ get the relation set and the variable sequence of $\xi$
3: $\mathsf{a} \leftarrow |\chi|$; ▷ get the arity of $\xi$
4: **return** Select($\mathcal{R}, \chi, \widetilde{\chi}, \widetilde{\omega}, \text{RT}_{\mathsf{a}}$);

## 4.3 Spatial Clustering QSAMs

In this section we define a family of QSAMs of the form

$$(\text{Qualify-SC}, \mathcal{D}_{\text{SC}}, \text{Retrieve-SC})$$

The data structures, the qualifier function, and the retriever function of this QSAM family are presented in Sections 4.3.2, 4.3.3, and 4.3.4, respectively, after having introduced some background notions in Section 4.3.1.

### 4.3.1 Background: Clustering Relations

Spatial Clustering QSAM (SC-QSAM) draws upon reduction and reconstruction strategies grounded on a special typology of qualitative spatial relations:

**Definition 4.2** (Clustering Relation) **-** An $\mathsf{a}$-ary qualitative spatial relation R is called a *clustering relation* if the following property holds for any $\mathsf{a}$-tuple $(o_1 \ldots, o_\mathsf{a}) \in \text{R}$ of objects from the spatial domain[1] $\mathcal{D}$:

$$(o_1, o_2 \ldots, o_\mathsf{a}) \in \text{R} \Rightarrow (o_1, \underline{o_2}, \ldots, \underline{o_\mathsf{a}}) \in \text{R} \; \forall \; \underline{o_i} \subseteq o_i \text{ with } i = 2, \ldots, \mathsf{a} \tag{4.1}$$

Practically, in the binary case, if $o_1$ is in a clustering relation with $o_2$, it is also in the same relation with any given object $\underline{o_2}$ contained in $o_2$.

**Corollary 4.1 -** *For any given qualitative spatial calculus $\mathcal{M}$, there exists a (possibly empty) subset $\underline{\mathcal{B}} \subseteq \mathcal{B}$ of its base relations which are clustering relations.*

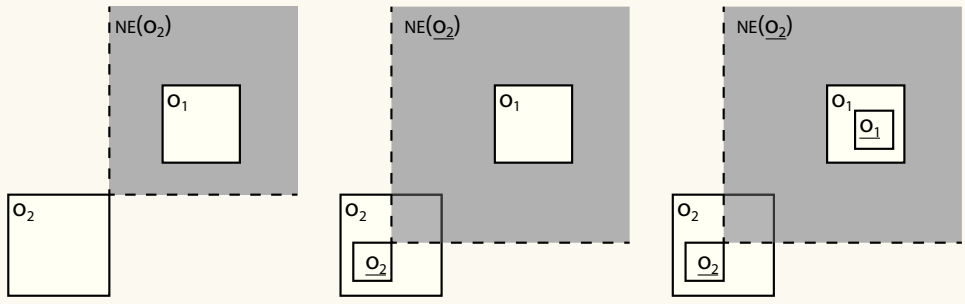
Since in the scope of this thesis we only consider connected regions without holes, the following corollary directly follows from Definition 4.2.

**Corollary 4.2 -** *If* R *is an* $\mathsf{a}$*-ary clustering relation, for any* $\mathsf{a}$*-tuple* $(o_1, \ldots, o_\mathsf{a}) \in$ R *the following equation holds:*

$$(o_1, o_2 \ldots, o_\mathsf{a}) \in \text{R} \Rightarrow (\underline{o_1}, \underline{o_2}, \ldots, \underline{o_\mathsf{a}}) \in \text{R} \; \forall \; \underline{o_i} \subseteq o_i \text{ with } i = 1, \ldots, \mathsf{a} \tag{4.2}$$

**Example 4.2** (Clustering Relation) **-** The binary Cardinal Direction Calculus (CDC) (cf. Section 2.2.4.4) relation *NorthEast* (NE) is a clustering relation. Figure 4.1(a) depicts a possible instantiation of the relation $\text{NE}(o_1, o_2)$, whereas Figure 4.1(b) shows the same CDC relation occurring between $o_1$ and a reference object $\underline{o_2} \subseteq o_2$. Finally, according to Corollary 4.2, we also have that $\text{NE}(\underline{o_1}, \underline{o_2})$ with $\underline{o_1} \subseteq o_1$, as depicted in Figure 4.1(c).

[1] The spatial domain $\mathcal{D}$ is the *infinite* set consisting of *any* connected, simple regions in $\mathbb{R}^2$ (cf. Section 2.2.2). It has not to be confused with a spatial dataset $O \subseteq \mathcal{D}$ that is a *finite* set of domain objects.




**(a)** NE$(o_1, o_2)$ **(b)** NE$(o_1, \underline{o_2})$ **(c)** NE$(\underline{o_1}, \underline{o_2})$

**Figure 4.1:** The Cardinal Direction Calculus relation *NorthEast* (NE) is a clustering relation.

The previous example suggests that, if the acceptance area (cf. Section 2.2.4.2) of a given relation consists of one connected and hole-free region, the relation can be identified to be a clustering one if the restriction of any of the reference objects implies an enlargement of the instantiation of the acceptance area. This is shown by the following:

**Theorem 4.1 -** *Let* R *be an* a*-ary qualitative spatial relation and* $Z_{\mathrm{R}}$ *its connected, hole-free, acceptance area. Then,* R *is a clustering relation if, and only if, the restriction of any reference object implies an enlargement of the acceptance area. That is:*

$$Z_{\mathrm{R}}(o_2, \ldots, o_{\mathsf{a}}) \subseteq Z_{\mathrm{R}}(\underline{o_2}, \ldots, \underline{o_{\mathsf{a}}}) \;\forall\; \underline{o_i} \subseteq o_i \text{ with } i = 2, \ldots, \mathsf{a} \tag{4.3}$$

*Proof. We start proofing that if* R *is a clustering relation, Equation 4.3 holds. By definition, the acceptance area* $Z_{\mathrm{R}}$ *of* R *is a region of the space parametric with respect to the reference objects* $(o_2, \ldots, o_{\mathsf{a}})$ *intervening in the relation such that* $(o_1, \ldots, o_{\mathsf{a}}) \in \mathrm{R} \iff o_1 \in Z_{\mathrm{R}}(o_2, \ldots, o_{\mathsf{a}})$ *for any* $o_1 \in \mathcal{D}$ *and for any* $(o_2, \ldots, o_{\mathsf{a}}) \in \mathcal{D}^{\mathsf{a}-1}$*. Since an acceptance zone instantiation* $Z_{\mathrm{R}}(o_2, \ldots, o_{\mathsf{a}})$ *is also a region of the modeled domain* $\mathcal{D}$ *which trivially satisfies the condition* $Z_{\mathrm{R}}(o_2, \ldots, o_{\mathsf{a}}) \subseteq Z_{\mathrm{R}}(o_2, \ldots, o_{\mathsf{a}})$*, even the* a*-tuple* $(Z_{\mathrm{R}}(o_2, \ldots, o_{\mathsf{a}}), o_2, \ldots, o_{\mathsf{a}})$ *is to be in* R*. Supposing that* R *is a clustering relation, from Definition 4.2 we have that* $(Z_{\mathrm{R}}(o_2, \ldots, o_{\mathsf{a}}), \underline{o_2}, \ldots, \underline{o_{\mathsf{a}}}) \in \mathrm{R}$ *for any* $(\underline{o_2}, \ldots, \underline{o_{\mathsf{a}}})$ *such that* $\underline{o_i} \subseteq o_i$ *with* $i = 2, \ldots, \mathsf{a}$*. Then, still from the definition of acceptance area, it follows that* $Z_{\mathrm{R}}(o_2, \ldots, o_{\mathsf{a}}) \subseteq Z_{\mathrm{R}}(\underline{o_2}, \ldots, \underline{o_{\mathsf{a}}})$

*We now need to proof that if Equation 4.3 holds,* R *is a clustering relation. Given that a spatial object* a*-tuple* $(o_1, o_2 \ldots, o_{\mathsf{a}})$ *is an element of* R *if, and only if,* $o_1 \subseteq Z_{\mathrm{R}}(o_2, \ldots, o_a)$ *and assumed that* $Z_{\mathrm{R}}(o_2, \ldots, o_{\mathsf{a}}) \subseteq Z_{\mathrm{R}}(\underline{o_2}, \ldots, \underline{o_{\mathsf{a}}})$*, it readily follows that* $o_1 \subseteq Z_{\mathrm{R}}(\underline{o_2}, \ldots, \underline{o_{\mathsf{a}}})$*. That is,* $(o_1, \underline{o_2} \ldots, \underline{o_{\mathsf{a}}}) \in \mathrm{R}$*.* □

### 4.3.2 Data Structure

The properties of clustering relations can be exploited to obtain a reduced qualified dataset via the generation of a set of fictitious geometries, each of which encloses a subset of the objects of the spatial dataset. We term such fictitious spatial objects *tiles* in order to distinguish them from the dataset ones.

**Definition 4.3** (Tileset) **-** Given a spatial dataset $O = \{o_1, \ldots, o_{\mathsf{N}}\}$, a set of geometries $T = \{t_1, \ldots, t_{\mathsf{q}}\}$ and a set function $\mathcal{T} : O \to T$, we say that $T$ is a *tileset* over $O$—generated through the *tiling function* $\mathcal{T}$—if, and only if, the following conditions hold simultaneously:

$$\forall\, t \in T\ \exists\, o \in O \text{ s.t. } t \cap o \neq \emptyset \tag{4.4}$$

$$\left(\bigcup_{t \in T} t\right) \cap \left(\bigcup_{o \in O} o\right) = \bigcup_{o \in O} o \tag{4.5}$$

The same spatial dataset can give rise to a multitude of tilesets according to the chosen tiling function $\mathcal{T}$. For example, Figure 4.2 shows two different tilesets over the same dataset. A tileset can be used to generate a clustering of the spatial dataset as follows.

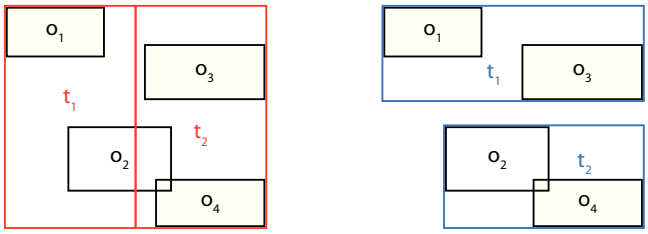



**Figure 4.2:** Two possible tilesets over the same spatial dataset.

**Definition 4.4** (Tileset-induced Spatial Clustering) **-** Given a spatial dataset $O$ and a tileset $T$ over it, it is possible to associate a subset of the spatial dataset $C \subseteq O$ to any tile $t \in T$ as follows: $C$ contains the dataset objects contained in $t$. If a dataset object is not contained within one tile, according to Definition 4.3, it is to be covered by the union of a subset of them and it is assigned to each corresponding cluster. If an object is contained within multiple tiles it is assigned to the cluster with minimum cardinality. We call the set $\mathcal{C} = \{C_1, \ldots, C_{\mathsf{q}}\}$ of such clusters a *$T$-induced clustering of $O$*.

We now have enough instruments to define what we call a *spatial clustering index*.

**Definition 4.5** (Spatial Clustering Index) **-** Let $O$ be a spatial dataset, $T$ a tileset of $\mathsf{q}$ tiles over it and $\mathcal{C}$ the spatial clustering induced by $T$. A *spatial clustering index* over $O$ via $T$—and $\mathcal{C}$—is a set $\mathcal{I} = \{\mathrm{E}_1, \ldots, \mathrm{E}_{\mathsf{q}} | \mathrm{E}_i = (t_i, C_i)$ with $i = 1, \ldots, \mathsf{q}\}$ where $t_i \in T$ is the $i$-th tile in the tileset and $C_i \in \mathcal{C}$ is the $i$-th cluster in the clustering.

For the sake of notation it is convenient to define a set of operators to easily refer to the index elements.

**Definition 4.6 -** Given an entry $\mathrm{E} = (t, C)$ from a spatial clustering index $\mathcal{I}$, we say that $t(\mathrm{E}) = t$ is the *tile associated with* E and that $C(\mathrm{E}) = C$ is the *cluster associated with* E. Similarly, given an $\mathsf{a}$-tuple of index entries $\epsilon = (\mathrm{E}_1, \ldots, \mathrm{E}_\mathsf{a})$, we say that $\tau(\epsilon) = (t(\mathrm{E}_1), \ldots, t(\mathrm{E}_\mathsf{a}))$ is the *tile* $\mathsf{a}$*-tuple associated with* $\epsilon$ and that $\sigma(\epsilon) = (C(\mathrm{E}_1), \ldots, C(\mathrm{E}_\mathsf{a}))$ is the *cluster* $\mathsf{a}$*-tuple associated with* $\epsilon$.

A spatial clustering index together with a set of relation tables provide the data structures underlying a spatial clustering QSAM. Note that, in this case, a relation table contains entries pointing either at dataset objects or at index entries.

### 4.3.3 Qualification

The qualification procedure for SC-QSAM resorts to a spatial clustering index to produce a reduced set of spatial relations. Before going through the algorithm, it is convenient, for the sake of exposition, to introduce the following concept.

**Definition 4.7 -** Given a dataset $O$ indexed by a spatial clustering index $\mathcal{I}$, any $\mathsf{a}$-tuple of index entries $\epsilon = (\mathrm{E}_1, \ldots, \mathrm{E}_\mathsf{a})$ gives rise to a set $\Omega_\epsilon$ of dataset object $\mathsf{a}$-tuples having the $i$-th term taken from the cluster $C(\mathrm{E}_i)$ associated with the $i$-th entry in $\epsilon$. We say that $\Omega_\epsilon$ *is induced by* $\epsilon$ and, if $|C(\mathrm{E}_i)| = \mathsf{N}_i$, we have that:

$$|\Omega_\epsilon| = \prod_{i=1}^{\mathsf{a}} \mathsf{N}_i$$

Given a dataset $O$ indexed by a spatial clustering index $\mathcal{I} = (T, \mathcal{C})$ and a calculus pool $\boldsymbol{\mathcal{P}}$, the dataset qualification procedure, reported in Algorithm 4.4, resorts to the tileset $T$ to exploit the properties of clustering relations.

For any $\mathsf{a}$-ary calculus in the pool $\mathcal{M} \in \boldsymbol{\mathcal{P}}$, the procedure iterates over every $\mathsf{a}$-tuple $\epsilon$ of entries from $\mathcal{I}$. If the relation $\mathrm{R}_\epsilon$, holding over the tile $\mathsf{a}$-tuple $\tau(\epsilon)$ associated with a given $\epsilon$, is a clustering relation—i.e. $\mathrm{R}_\epsilon \in \underline{\mathcal{B}}(\mathcal{M})$—then, according to Equation 4.2, the same relation occurs over every object $\mathsf{a}$-tuple $\omega \in \Omega_\epsilon$ induced by $\epsilon$. Accordingly, computing the relations holding over any $\omega \in \Omega_\epsilon$ is superfluous and we can simply store $\mathrm{R}_\epsilon$ in the relation table $\mathrm{RT}_\mathsf{a}$. Conversely, if $\mathrm{R}_\epsilon$ is not a clustering relation, it is necessary to compute and store the relations occurring over all the object $\mathsf{a}$-tuples mentioned above. Note that, if the tile $\tau$ is an $\mathsf{a}$-tuple having all terms equal we simply compute the relations occurring over all the objects in the corresponding cluster by resorting to the qualifier function Qualify-QSL described in Section 4.2.

**Algorithm 4.4** Qualify-SC: Populates the relation tables in $\mathcal{D}$ with the relations induced by the spatial dataset and reduced according to the spatial clustering index $\mathcal{I}$

**Input:**
$O$: Spatial set
$\boldsymbol{\mathcal{P}}$: Calculus pool
$\mathcal{I}$: Spatial Clustering index
**Output:**
LUTs in $\mathcal{D}$ are filled with relations from the $\boldsymbol{\mathcal{P}}$-qualified dataset $\mathcal{R}_{\boldsymbol{\mathcal{P}}}(O)$

```
1: function QUALIFY-SC(O,P,I)
2:     for all M ∈ P do
3:         B ← B(M);                    ▷ get the clustering relations of the calculus at hand
4:         a ← a(M);                    ▷ get the arity of the calculus at hand
5:         for all ϵ ∈ I^a do
6:             if ϵ[1] = ϵ[2] = ··· = ϵ[a] then QUALIFY-QSL(C(ϵ[1]),{M});
7:             else
8:                 R_ϵ ← COMPUTERELATION_M(τ(ϵ));
9:                 if R_ϵ ∈ B then RT_a ← RT_a ⋃ (R_ϵ, ϵ);
10:                else
11:                    for all ω ∈ Ω_ϵ do
12:                        R_ω ← COMPUTERELATION_M(ω);
13:                        RT_a ← RT_a ⋃ (R_ω, ω);
```

The qualification procedure reported in Algorithm 4.4 produces what we call an *$\mathcal{I}$-reduced, $\boldsymbol{\mathcal{P}}$-qualified dataset* $\mathcal{R}_{\boldsymbol{\mathcal{P}},\mathcal{I}}(O)$ such that

$$\mathcal{R}_{\boldsymbol{\mathcal{P}},\mathcal{I}}(O) = \bigcup_{\mathcal{M}\in\boldsymbol{\mathcal{P}}} \mathcal{R}_{\mathcal{M},\mathcal{I}}(O) \tag{4.6}$$

where $\mathcal{R}_{\mathcal{M},\mathcal{I}}(O)$ is the set of relations generated for a calculus $\mathcal{M}$ of arity $\mathsf{a}$ and such that

$$|\mathcal{R}_{\mathcal{M},\mathcal{I}}(O)| = |\mathcal{R}_{\mathcal{M}}(O)| - \rho \text{ with } \rho \geq 0 \tag{4.7}$$

The reduction $\rho$ performed by Qualify-SC is measured with respect to the complete $\mathcal{M}$-qualified dataset $\mathcal{R}_{\mathcal{M}}(O)$ (cf. Equation 3.2). It depends on the index $\mathcal{I}$ and on the number of clustering relations occurring over its tiles. A "bad" index may yield a null reduction. Moreover, it may cause the relations occurring over some object $\mathsf{a}$-tuples to be computed multiple times. This, in turn, causes the procedure to perform in time higher than that required by a full qualification.

**Example 4.3** (Bad Indexing) **-** Let us consider the case depicted in Figure 4.2(a). That is, $O = \{o_1, o_2, o_3, o_4\}$ and $\mathcal{I} = \{\mathrm{E}_1, \mathrm{E}_2\}$ with:

$$t(\mathrm{E}_1) = t_1 \ ; \ C(\mathrm{E}_1) = C_1 = \{o_1, o_2\}$$
$$t(\mathrm{E}_2) = t_2 \ ; \ C(\mathrm{E}_2) = C_2 = \{o_2, o_3, o_4\}$$

Let us assume that the calculus pool $\boldsymbol{\mathcal{P}} = \{\mathrm{RCC}\}$ consists only of the Region Connection Calculus (RCC) (cf. Section 2.2.4.4) which provides only one clustering relation: $\underline{\mathcal{B}}(\mathrm{RCC}) = \{\mathrm{DC}\}$. Then, Qualify-SC performs according to Table 4.1:

| $(\mathrm{E}_1, \mathrm{E}_1)$ | – | $(\mathrm{E}_1, \mathrm{E}_2)$ | EC | $(\mathrm{E}_2, \mathrm{E}_1)$ | EC | $(\mathrm{E}_2, \mathrm{E}_2)$ | – |
|---|---|---|---|---|---|---|---|
| $(o_1, o_1)$ | EQ | $(o_1, o_2)$ | DC | $(o_2, o_1)$ | DC | $(o_2, o_2)$ | EQ |
| $(o_1, o_2)$ | DC | $(o_1, o_3)$ | DC | $(o_2, o_2)$ | EQ | $(o_2, o_3)$ | DC |
| $(o_2, o_1)$ | DC | $(o_1, o_4)$ | DC | $(o_3, o_1)$ | DC | $(o_2, o_4)$ | PO |
| $(o_2, o_2)$ | EQ | $(o_2, o_2)$ | EQ | $(o_3, o_2)$ | DC | $(o_3, o_2)$ | DC |
| | | $(o_2, o_3)$ | DC | $(o_4, o_1)$ | DC | $(o_3, o_3)$ | EQ |
| | | $(o_2, o_4)$ | PO | $(o_4, o_2)$ | PO | $(o_3, o_4)$ | DC |
| | | | | | | $(o_4, o_2)$ | PO |
| | | | | | | $(o_4, o_3)$ | DC |
| | | | | | | $(o_4, o_4)$ | EQ |

**stored relations** non-stored relations
repeated computations

**Table 4.1:** Execution summary of Qualify-SC over the dataset in Figure 4.2(a)

The table reports in blue the relations stored in the relation table $\mathrm{RT}_2$ and in gray the non-stored ones. The symbol – indicates that the relation for the associated pair has not been computed at all. The relations computed more than once are reported in red. Each column represents the iteration over the index entry pair reported in the first row. Accordingly, the totality of blue entries corresponds to the set $\mathcal{R}_{\mathrm{RCC},\mathcal{I}}(O)$ whereas the non-gray ones give the order of the execution time.

So, for example, the last column indicates the iteration over the pair $(\mathrm{E}_2, \mathrm{E}_2)$: since the 2-tuple has all terms equal, the algorithm does not compute the relation occurring over it and qualifies the associated object cluster $C_1$.

The remainder of this section is devoted to analyze the qualification procedure we just described. In particular we are interested in identifying some index parameters that can be tuned to obtain $\mathcal{M}$-qualified datasets as lighter as possible.

In order to minimize $|\mathcal{R}_{\mathcal{M},\mathcal{I}}(O)|$ we have to maximize $\rho$. First off, we need to make explicit the cardinality of the reduced qualified dataset. To this end, let us assume that $U \subseteq T^{\mathsf{a}}$ is the set of tile $\mathsf{a}$-tuples having all the terms equal. The rest is split between the sets $V$ and $W$ in such a way that $V$ contains the $v$ tile

$\mathsf{a}$-tuples in a clustering relation. Then, according to Algorithm 4.4 we have that

$$|\mathcal{R}_{\mathcal{M},\mathcal{I}}(O)| = \sum_{\tau_j \in U} \prod_{k=1}^{\mathsf{a}} \mathsf{N}_{j,k} + v + \sum_{\tau_j \in W} \prod_{k=1}^{\mathsf{a}} \mathsf{N}_{j,k} \tag{4.8}$$

where $\mathsf{N}_{j,k}$ is assumed to be the cardinality of the object cluster $C_{j,k}$ associated with the $k$-th term of the $j$-th tile $\mathsf{a}$-tuple $\tau_j$.

To assess the reduction amount, it is convenient to express $|\mathcal{R}_{\mathcal{M}}(O)|$ as a function of the sets $U$, $V$ and $W$. Given a partition of the spatial dataset, this can be done easily (cf. Example 4.3 for an empirical demonstration). However, since we did not impose any restriction on the way the index $\mathcal{I}$ is generated, the clustering $\mathcal{C}$ may or may not provide a partition of the spatial dataset $O$. That is, the summation of the cardinality of its object clusters exceeds the cardinality of the spatial dataset of a quantity $e \geq 0$. Nonetheless, given the clustering $\mathcal{C} = \{C_1, \ldots, C_{\mathsf{q}}\}$, it is always possible to stem from it a partition $\widehat{\mathcal{C}} = \{\widehat{C}_1, \ldots, \widehat{C}_{\mathsf{q}}\}$ of $O$ such that:

$$\widehat{C}_i = \begin{cases} C_i & \text{if } i = 1 \\ C_i \setminus \bigcup_{j=1}^{i-1} \widehat{C}_j & \text{if } 2 \leq i \leq \mathsf{q} \end{cases}$$

If we denote with $\widehat{\mathsf{N}}_i$ the cardinality of the $i$-th cluster $C_i$ we have that $\mathsf{N}_i - \widehat{\mathsf{N}}_i = e_i$ and, consequently, that

$$e = \sum_{i=1}^{\mathsf{q}} e_i$$

Then, it is possible to rewrite Equation 3.2 as follows:

$$|\mathcal{R}_{\mathcal{M}}(O)| = \sum_{\tau_j \in U} \prod_{k=1}^{\mathsf{a}} \widehat{\mathsf{N}}_{j,k} + \sum_{\tau_j \in V} \prod_{k=1}^{\mathsf{a}} \widehat{\mathsf{N}}_{j,k} + \sum_{\tau_j \in W} \prod_{k=1}^{\mathsf{a}} \widehat{\mathsf{N}}_{j,k} \tag{4.9}$$

where $\widehat{\mathsf{N}}_{j,k}$ is assumed to be the cardinality of the object cluster $\widehat{C}_{j,k}$ associated with the $k$-th term of the $j$-th tile $\mathsf{a}$-tuple $\tau_j$.

Lastly, let us define the following symbols which allow for a leaner notation and a smoother analysis:

$$\rho_U = \sum_{\tau_j \in U} \prod_{k=1}^{\mathsf{a}} (\widehat{\mathsf{N}}_{j,k}) - \prod_{k=1}^{\mathsf{a}} \mathsf{N}_{j,k}$$

$$\rho_V = \sum_{\tau_j \in V} \prod_{k=1}^{\mathsf{a}} (\widehat{\mathsf{N}}_{j,k}) - v$$

$$\rho_W = \sum_{\tau_j \in W} \prod_{k=1}^{\mathsf{a}} (\widehat{\mathsf{N}}_{j,k}) - \prod_{k=1}^{\mathsf{a}} \mathsf{N}_{j,k}$$

Hence, from Equations 4.7, 4.8, and 4.9, the reduction performed by Algorithm 4.4

can be quantified in:

$$\rho = \rho_U + \rho_V + \rho_W \tag{4.10}$$

It is worth to recall that $\widehat{\mathsf{N}}_{j,k} = \mathsf{N}_{j,k} - e_{j,k}$ by construction, therefore the terms $\rho_U$ and $\rho_W$ are non-positive and we want their absolute values to be as small as possible, ideally 0. Conversely, $\rho_V$ is obviously non-negative and has to be maximized. Let us go through Equation 4.10 term by term:

- $\rho_U$ only depends on $e$. In particular we have that, if $e \to 0$, $\widehat{\mathsf{N}}_{j,k} \to \mathsf{N}_{j,k}$ and $\rho_U \to 0$.
- $\rho_V$ scales proportionally with the number of tile $\mathsf{a}$-tuples in a clustering relation, i.e. the higher $v$, the higher $\rho_V$.
- Since, by definition, $W = (T^{\mathsf{a}} \setminus U) \setminus V$, the term $\rho_W$ depends on both $e$ and $v$. In particular, the bigger is $v$ the smaller is $\rho_W$ and, just like $\rho_U$, it tends to zero as $e$ tends to zero.

In conclusion, to obtain a good reduction the index $\mathcal{I}$ has to be tuned in order to reduce the overlapping of the clusters in $\mathcal{C}$ (reduce $e$) and to augment the possibility that the tuples of tiles in $T$ are in a clustering relation (augment $v$). Given that the calculus pool $\boldsymbol{\mathcal{P}}$ possibly contains a variety of calculi, achieving the second goal is quite hard since adjusting the index for a calculus might yield an unsatisfactory solution for another. Accordingly, any given $\mathcal{I}$ is most suitable for a specific set of spatial calculi. In any case, to obtain acceptable reductions, a spatial clustering index will have to be tested and tuned to detect the best parameter settings for the calculus pool and the spatial dataset at hand.

**Example 4.4** (Good Indexing) **-** Let us resume Example 4.3. This time, we consider the case that the spatial dataset is indexed as depicted in Figure 4.2(b). Therefore, $O = \{o_1, o_2, o_3, o_4\}$ and $\mathcal{I} = \{\mathrm{E}_1, \mathrm{E}_2\}$ with:

$$t(\mathrm{E}_1) = t_1 \; ; \; C(\mathrm{E}_1) = C_1 = \{o_1, o_3\}$$
$$t(\mathrm{E}_2) = t_2 \; ; \; C(\mathrm{E}_2) = C_2 = \{o_2, o_4\}$$

| $(\mathrm{E}_1, \mathrm{E}_1)$ | – | $(\mathrm{E}_1, \mathrm{E}_2)$ | DC | $(\mathrm{E}_2, \mathrm{E}_1)$ | DC | $(\mathrm{E}_2, \mathrm{E}_2)$ | – |
|---|---|---|---|---|---|---|---|
| $(o_1, o_1)$ | EQ | $(o_1, o_2)$ | – | $(o_2, o_1)$ | – | $(o_2, o_2)$ | EQ |
| $(o_1, o_3)$ | DC | $(o_1, o_4)$ | – | $(o_2, o_3)$ | – | $(o_2, o_4)$ | PO |
| $(o_3, o_1)$ | DC | $(o_3, o_2)$ | – | $(o_4, o_1)$ | – | $(o_4, o_2)$ | PO |
| $(o_3, o_3)$ | EQ | $(o_3, o_4)$ | – | $(o_4, o_3)$ | – | $(o_4, o_4)$ | EQ |

**stored relations** non-stored relations
repeated computations

**Table 4.2:** Execution summary of Qualify-SC over the dataset in Figure 4.2(b)

Then Qualify-SC performs according to Table 4.2 (cf. Example 4.3 for interpretation directives). Note that, in this case the clustering $\mathcal{C}$ provides a partition of the spatial dataset $O$—i.e. $e = 0$. This implies that, consistently with Equation 4.9, the cardinality of the complete qualified dataset $\mathcal{R}_{\mathcal{M}}(O)$ equals the summation of the number of object pairs taken from any cluster and it is equal to $|\mathcal{R}_{\mathcal{M}}(O)| = 8 + 4 + 4 = 16$. Moreover, according to Equantion 4.8, Qualify-SC produces a qualified dataset $\mathcal{R}_{\text{RCC},\mathcal{I}}(O)$ reduced of $\rho = 6$ relations with respect to the complete one: $|\mathcal{R}_{\text{RCC},\mathcal{I}}(O)| = 8 + 2 + 0 = 10$. Finally, no relations are computed more than once, i.e. the algorithm runs faster than a plain qualification.

### 4.3.4 Retrieval

The retriever function for a Spatial Clustering qsam (sc-qsam) can be implemented as the generic one reported in Algorithm 3.1 where the function ExtendPMatching is realized as in Algorithm 4.5.

**Algorithm 4.5** ExtendPMatching-SC: Extends the partial matching $(\widetilde{\chi}, \widetilde{\omega})$ according to constraint $\xi$ and exploiting the index $\mathcal{I}$

**Input:**
$(\widetilde{\chi}, \widetilde{\omega})$: partial matching
$\xi = (\mathcal{R}, \chi)$: constraint to satisfy
$\mathcal{I}$: Spatial Clustering index
**Output:**
$\Omega_\xi$: set of matchings extended from $(\widetilde{\chi}, \widetilde{\omega})$ and satisfying constraint $\xi$

1: **function** ExtendPMatching-SC($\widetilde{\chi}, \widetilde{\omega}, \xi, \mathcal{I}$)
2: $(\mathcal{R}, \chi) \leftarrow \xi$; ▷ get the relation set and the variable sequence of $\xi$
3: $\mathsf{a} \leftarrow |\chi|$; ▷ get the arity of $\xi$
4: $\left(\underline{\mathcal{R}}, \overline{\mathcal{R}}\right) \leftarrow \mathcal{R}$;
5: **return** Select-SC($\underline{\mathcal{R}}, \chi, \widetilde{\chi}, \widetilde{\omega}, \text{RT}_\mathsf{a}, \mathcal{I}$) $\cap$ Select($\overline{\mathcal{R}}, \chi, \widetilde{\chi}, \widetilde{\omega}, \text{RT}_\mathsf{a}$);

The algorithm extracts relevant information from the input constraint $\xi$. In this case, beyond the set of constrained variables $\chi$ and the arity $\mathsf{a}$, we also assume that the enforced relation set $\mathcal{R}$ consists of two subsets: $\underline{\mathcal{R}}$ contains all the enforced relations that are clustering relations, whereas the set $\overline{\mathcal{R}}$ contains the non-clustering ones. Since the non-clustering relations among object $\mathsf{a}$-tuples are all stored in the relation table $\text{RT}_\mathsf{a}$ we simply retrieve them via the function Select that we already used in Algorithm 4.3.

With regard to the clustering relations: the relation table $\text{RT}_\mathsf{a}$ only stores those occurring among the tiles associated with index entries or among objects in the same cluster. Therefore, we assume that the function Select-SC behaves

similarly to Select: It retrieves from the relation table $\mathrm{RT}_{\mathsf{a}}$ all the entries $(\mathrm{R}, \cdot)$ with $\mathrm{R} \in \underline{\mathcal{R}}$. If a relation table entry $(\mathrm{R}, \omega)$ is about an object $\mathsf{a}$-tuple, the latter is appended to the result set. Otherwise, we have that $(\mathrm{R}, \mathrm{E})$ is about an index entry $\mathsf{a}$-tuple. In this case, by accessing $\mathcal{I}$, Select-SC generates and appends to the result the set $\Omega_{\mathrm{E}}$ of object $\mathsf{a}$-tuples induced by $\mathcal{I}$. Of course, similarly to the function Select, Select-SC only returns tuples compatible with the input (cf. Example 4.1). The intersection of the two retrievals is the set of (partial) matchings extended from the input one $(\widetilde{\chi}, \widetilde{\omega})$ according to the constraint $\xi$ in input.

**Example 4.5 -** Consider the following QSCQ to be executed against the $\mathcal{I}$-reduced, RCC-qualified dataset obtained in Example 4.4:

$$\mathcal{Q} = (\{x_1, x_2, x_3\}, \{(\{\mathrm{PO}\}, (x_1, x_2)), (\{\mathrm{DC}\}, (x_2, x_3))\}), \text{ that is } \begin{cases} x_1 \ \mathrm{PO} \ x_2 \\ x_2 \ \mathrm{DC} \ x_3 \end{cases}$$

Then the retrieval, according to Algorithms 3.1 and 4.5, works as follows. First, the constraint $x_1$ PO $x_2$ is considered. Since PO is not a clustering relation, only the function Select generates a set of results. Since no variables have been assigned yet, the procedure simply retrieves from the relation table $\mathrm{RT}_2$ all the entries of the form $(\mathrm{PO}, \cdot, \cdot)$. After the iteration over the first constraint we have a set of partial matchings represented by:

$$\widetilde{\chi} = (x_1, x_2) \text{ and } \Omega_1 = \left\{ \begin{array}{c} (o_2, o_4) \\ (o_4, o_2) \end{array} \right\}$$

Where $\widetilde{\chi}$ is the set of assigned variable and $\Omega_1$ is the set of object pairs satisfying the first constraint.

Then, Algorithm 3.1 iterates over the constraint $x_2$ DC $x_3$ and tries to extend the two partial matchings via the procedure ExtendPMatching-SC. Since DC is a clustering relation only the function Select-SC participates actively to the matching extension. Let us look at the extension of the two matchings separately. When extending the partial matching $(\widetilde{\chi}, \widetilde{\omega}) = ((x_1, x_2), (o_2, o_4))$, the function retrieves from the relation table $\mathrm{RT}_2$ all the entries of the form $(\mathrm{DC}, \cdot, \cdot)$:

$$\left\{ \begin{array}{c} (\mathrm{DC}, o_1, o_3) \\ (\mathrm{DC}, o_3, o_1) \end{array} \right\} \text{ and } \left\{ \begin{array}{c} (\mathrm{DC}, \mathrm{E}_1, \mathrm{E}_2) \\ (\mathrm{DC}, \mathrm{E}_2, \mathrm{E}_1) \end{array} \right\}$$

The first two are about dataset objects and are discarded since the object pairs are incompatible with the input matching. The other two are about index entry pairs which are used to access the corresponding clusters to reconstruct non-stored relations:

$$(\text{DC}, \text{E}_1, \text{E}_2) \rightarrow \left\{ \begin{array}{l} (\text{DC}, o_1, o_2) \\ (\text{DC}, o_1, o_4) \\ (\text{DC}, o_3, o_2) \\ (\text{DC}, o_3, o_4) \end{array} \right\} \text{ and } (\text{DC}, \text{E}_2, \text{E}_1) \rightarrow \left\{ \begin{array}{l} (\text{DC}, o_2, o_1) \\ (\text{DC}, o_2, o_3) \\ (\text{DC}, o_4, o_1) \\ (\text{DC}, o_4, o_3) \end{array} \right\}$$

The first set is discarded in its entirety since none of the object pairs are compatible with the input $\widetilde{\omega} = (o_2, o_4)$. The last two elements of the second set are compatible, therefore the object pairs are extracted, merged with the input, and returned to the main procedure:

$$\left\{ \begin{array}{l} (o_2, o_4, o_1) \\ (o_2, o_4, o_3) \end{array} \right\}$$

The procedure operates similarly for the extension of the second partial matching $(\widetilde{\chi}, \widetilde{\omega}) = ((x_1, x_2), (o_2, o_4))$.

The final result is:

$$\widetilde{\chi} = (x_1, x_2, x_3) \text{ and } \Omega_2 = \left\{ \begin{array}{l} (o_2, o_4, o_1) \\ (o_2, o_4, o_3) \\ (o_4, o_2, o_1) \\ (o_4, o_2, o_3) \end{array} \right\}$$

### 4.3.5 Final Remarks and Discussion

Spatial Clustering qsam (sc-qsam) makes use of a set of relation tables as underlying data structures and of a tile&cluster technique (spatial clustering index) that, possibly, allows for computing and storing a reduced number of relations with respect to a full qualified dataset. In the retrieval phase the properties of clustering relations are exploited to rebuild non-stored pieces of qualitative information without resorting to direct computation through computational geometry functions.

It is worth to remark that each calculus in the pool $\boldsymbol{\mathcal{P}}$ has to be manually analyzed in order to identify the set of clustering relations $\underline{\mathcal{B}}$. To do this the outcomes of Theorem 4.1 can be exploited.

In Section 4.3.3, it has been shown that the reduction amount and, consequently, the qscq retrieval time, strongly depend on the spatial clustering index adopted. It has been shown that a "bad" index can easily yield worse space-time performance than a full qualification. In particular the reduction has been expressed in terms of some index tuning parameters that *have to* be used in actual qsam implementations to assure a significant reduction.

A spatial clustering index has been constructively defined (cf. Definition 4.5) via a two-step process: (*i*) definition of a tileset and (*ii*) association of a cluster to each tile. However, it has to be noted that the role of clustering and tiling can be reversed. That is, one can first generate a spatial clustering of the dataset according to a given clustering function and then associate a geometry—e.g. the bounding box of the clustered objects—to each cluster. As long as the resulting data structure satisfies Equations 4.4 and 4.5, it is eligible to be considered a spatial clustering index and can be used for applying spatial clustering QSAM techniques.

Finally, the way a spatial clustering index has been defined is general enough to embrace a variety of standard spatial indexes—e.g. the GRID file (Nievergelt *et al.*, 1984)—and, it is easily extensible to comprehend hierarchical data structures—e.g. R-tree (Guttman, 1984), quad-tree (Finkel & Bentley, 1974)—as well:

**Definition 4.8** (Hierarchical Spatial Clustering Index) **-** Given a spatial dataset $O$, a *hierarchical spatial clustering index* $\mathcal{H} = (\mathcal{I}_1, \ldots, \mathcal{I}_l)$ over $O$ is a series of $l$ spatial clustering indexes such that $\mathcal{I}_1$ indexes the spatial dataset and each $\mathcal{I}_i$ (with $2 \leq i \leq l$) indexes the tileset of the underlying index $\mathcal{I}_{i-1}$.

Note that we do not impose any restriction on the indexes at different level of the hierarchy, i.e. each $\mathcal{I}_i$ can be generated according to a different tiling function $\mathcal{T}_i$, although it is easier to deal with a uniform tiling strategy. Of course, when dealing with a hierarchical spatial clustering index a recursive version of Algorithms 4.4 and 4.5 has to be implemented where the main logic is slightly modified to account the level of the hierarchy under consideration; this is addressed in Section 5.2.2 where the implementation of a spatial clustering QSAM based on R*-tree Beckmann *et al.* (cf. 1990) is discussed.

## 4.4 QSR-based QSAM

In this section we define a QSAM grounded on Qualitative Spatial Representation and Reasoning (QSR) of the form

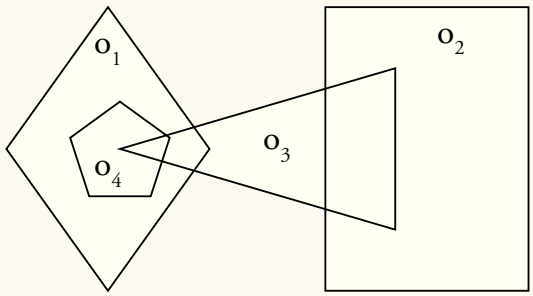
$$(\text{Qualify-QSR}, \mathcal{D}_{\text{QSR}}, \text{Retrieve-QSR})$$

The data structures, the qualifier function, and the retriever function of this QSAM are presented in Sections 4.4.2, 4.4.3, and 4.4.4, respectively, after having introduced some background notions in Section 4.4.1.

### 4.4.1 Background: the Inference Graph

The main underlying idea of QSR-based QSAM (QSR-QSAM) is that, given a spatial dataset $O$ and its corresponding $\boldsymbol{\mathcal{P}}$-qualification $\mathcal{R}_{\boldsymbol{\mathcal{P}}}(O)$, some of the relations in $\mathcal{R}_{\boldsymbol{\mathcal{P}}}(O)$ are *inferable* by others.

**Example 4.6** (Inferable Relation) **-** Let us consider the spatial dataset in Figure 4.3(a) and let us focus on the occurring topological relations. Note that $o_4$ *is inside* $o_1$*, which is disconnected from* $o_2$. Then, it is to be that $o_4$ *is also disconnected from* $o_2$. This is confirmed by the RCC-qualified dataset (cf. Section 2.2.4.4) in Figure 4.3(b). We say that the relation $\mathrm{DC}(o_4, o_2)$ is *inferable from* the relations $\mathrm{NTPP}(o_4, o_1)$ and $\mathrm{DC}(o_1, o_2)$.




**(a)** A spatial dataset

$$\left\{\begin{array}{cccc} \mathrm{EQ}(o_1,o_1) & \mathrm{DC}(o_1,o_2) & \mathrm{PO}(o_1,o_3) & \mathrm{NTPPI}(o_1,o_4) \\ \mathrm{DC}(o_2,o_1) & \mathrm{EQ}(o_2,o_2) & \mathrm{PO}(o_2,o_3) & \mathrm{DC}(o_2,o_4) \\ \mathrm{PO}(o_3,o_1) & \mathrm{PO}(o_3,o_2) & \mathrm{EQ}(o_3,o_3) & \mathrm{PO}(o_3,o_4) \\ \mathrm{NTPP}(o_4,o_1) & \mathrm{DC}(o_4,o_2) & \mathrm{PO}(o_4,o_3) & \mathrm{EQ}(o_4,o_4) \end{array}\right\}$$

**(b)** RCC-qualification

**Figure 4.3:** A spatial dataset and its RCC-qualification.

The QSR-QSAM qualification function produces a reduced qualified dataset obtained from $\mathcal{R}_{\boldsymbol{\mathcal{P}}}(O)$ by removing inferable relations. The dataset has to contain enough relations to allow for rebuilding, at retrieval time, the missing ones via QSR techniques—i.e. no need to run into computational geometry procedures.

In order to produce the reduced qualified dataset we have to be able to *automatically* detect inferable relations in $\mathcal{R}_{\boldsymbol{\mathcal{P}}}(O)$. To achieve this task we propose a novel data structure that explicitly encodes inferential dependencies:

**Definition 4.9** (Inference Graph) **-** Given a spatial dataset $O$ and a calculus pool $\boldsymbol{\mathcal{P}}$, for any calculus $\mathcal{M} \in \boldsymbol{\mathcal{P}}$ it is possible to build a B-graph (cf. Definition 2.1) $\mathcal{IG} = (\mathcal{N}, \mathcal{A})$. Its node set $\mathcal{N} = \mathcal{R}_{\mathcal{M}}(O)$ coincides with the $\mathcal{M}$-qualified dataset—i.e. any node stands for one relation induced by $O$—and it is referred to as $\mathcal{N}(\mathcal{IG})$. There exists an oriented hyper edge $a = (\mathcal{T}, h) \in \mathcal{A}$ if, and only if, the relation represented by $h$ can be *inferred* from the relations in $\mathcal{T}$. The edge set is referred to as $\mathcal{A}(\mathcal{IG})$.

**Example 4.7 -** Figure 4.4 depicts a very simple inference graph $\mathcal{IG} = (\mathcal{N}, \mathcal{A})$ consisting of three nodes $\mathcal{N} = \{\text{NTPP}(o_4, o_1), \text{DC}(o_1, o_2), \text{DC}(o_4, o_2)\}$ and one hyperarc $\mathcal{A} = \{a\}$ with $\mathcal{T}(a) = \{\text{NTPP}(o_4, o_1), \text{DC}(o_1, o_2)\}$ and $h(a) = \text{DC}(o_4, o_2)$. It corresponds to the logical inference process reported in Example 4.6.

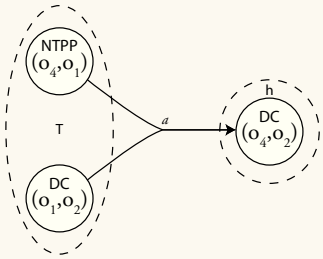



**Figure 4.4:** Simple inference graph.

Although an inference graph is a hypergraph, it is convenient, for reasons that will become clear soon, to represent it as a normal graph. To do so, it is necessary to interpret any hyperarc $a = (\mathcal{T}, h)$ as an *inverse* arborescence[1]. We call each such arborescence an *inference path* leading from $\mathcal{T}$ to $h$.

**Example 4.8 -** The inference graph in Figure 4.4 can be interpreted as in Figure 4.5.

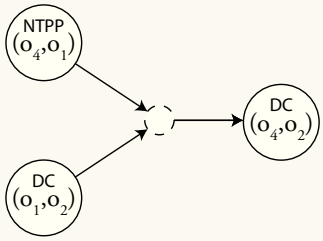



**Figure 4.5:** An inference path.

Given an inference graph $\mathcal{IG} = (\mathcal{N}, \mathcal{A})$ it is possible to obtain a graph representation of it $\mathcal{IG}_g = (\mathcal{N}_g, \mathcal{A}_g)$ representing every hyperarc $a = (\mathcal{T}, h)$ as an inference path $\mathcal{IP}$. The node set $\mathcal{N}_g = \{\mathcal{N}_R, \mathcal{N}_I\}$ of the resulting graph consists of two different types of nodes. The set $\mathcal{N}_R$ contains what we call *relational nodes* and corresponds to the $\mathcal{M}$-qualified dataset. The set $\mathcal{N}_I$ contains the nodes used to interpret hyperarcs as inference paths. Accordingly, we name such nodes *inferential nodes*.

[1]An arborescence is a directed, rooted tree in which all edges point away from the root. With the adjective inverse we intend that all the edges point towards the root.

**Proposition 4.1 -** *Inference paths always start from a set $\mathcal{T} \subset \mathcal{N}_R$ of relational nodes called tail and always end in one relational node $h \in \mathcal{N}_R$ called head. Proof. An inference path is a graph interpretation of an inference graph hyperarc. Since an inference graph is a* B*-graph (cf. Definition 4.9), each of its hyperarcs originates in a set of nodes $\mathcal{T}$ and heads to one node $h$.* □

An inference path is an instantiation of an inference rule: internal nodes represent the operations that have to be executed on the relations in the tail of the path to obtain its head relation.

#### 4.4.1.1 Generating the Inference Graph

A spatial calculus is typically equipped with some reasoning tables (cf. Section 2.2.3) that represent inference operations between the relations in the calculus. Thus, inference paths can be automatically generated from such tables.

**Reasoning Tables** Typically, reasoning tables are of two types: permutation and composition tables. Let us assume, without loss of generality, that the set of base relations $\mathcal{B}$ of a given $\mathsf{a}$-ary calculus $\mathcal{M} \in \boldsymbol{\mathcal{P}}$ are numbered from 1 to $\mathsf{b}$:

$$\mathcal{B} = \{\mathrm{R}_1, \ldots, \mathrm{R}_\mathsf{b}\}$$

Then, permutation tables are vectors of size $\mathsf{b}$ of the following form:

| $\mathrm{R}_1$ | $\mathrm{R}_2$ | $\cdots$ | $\mathrm{R}_\mathsf{b}$ |
|---|---|---|---|
| $P(\mathrm{R}_1)$ | $P(\mathrm{R}_2)$ | $\cdots$ | $P(\mathrm{R}_\mathsf{b})$ |

The $i$-th entry represents the relation $P(\mathrm{R}_i) \in 2^\mathcal{B}$ which holds over a given permutation $P$ of any object $\mathsf{a}$-tuple $\omega$ over which the relation $\mathrm{R}_i$ holds. The object permutation $P$ identifies the table among the $\mathsf{a} - 1$ possible ones.

Similarly, composition tables are $\mathsf{b} \times \mathsf{b}$ tables of the following form:

| | $\mathrm{R}_1$ | $\mathrm{R}_2$ | $\cdots$ | $\mathrm{R}_\mathsf{b}$ |
|---|---|---|---|---|
| $\mathrm{R}_1$ | $\mathrm{R}_1 \circ \mathrm{R}_1$ | $\mathrm{R}_1 \circ \mathrm{R}_2$ | $\cdots$ | $\mathrm{R}_1 \circ \mathrm{R}_\mathsf{b}$ |
| $\mathrm{R}_2$ | $\mathrm{R}_2 \circ \mathrm{R}_1$ | $\mathrm{R}_2 \circ \mathrm{R}_2$ | $\cdots$ | $\mathrm{R}_2 \circ \mathrm{R}_\mathsf{b}$ |
| $\vdots$ | $\vdots$ | $\vdots$ | $\ddots$ | $\vdots$ |
| $\mathrm{R}_\mathsf{b}$ | $\mathrm{R}_\mathsf{b} \circ \mathrm{R}_1$ | $\mathrm{R}_\mathsf{b} \circ \mathrm{R}_2$ | $\cdots$ | $\mathrm{R}_\mathsf{b} \circ \mathrm{R}_\mathsf{b}$ |

The entry individuated by the pair $(\mathrm{R}_i, \mathrm{R}_j)$ represents the relation holding over the object $\mathsf{a}$-tuple $\omega$ obtained from a given concatenation of two other object $\mathsf{a}$-tuples $\omega_i$ and $\omega_j$ such that $\mathrm{R}_i$ holds over $\omega_i$ and relation $\mathrm{R}_j$ holds over $\omega_j$. Two object $\mathsf{a}$-tuples can be concatenated in $(\mathsf{a}!)^3$ different ways, each of which raises a different composition table.

**Example 4.9** (RCC Reasoning Tables) **-** Let us consider again RCC (cf. Section 2.2.4.4). Since it is a binary calculus, it admits only one type of permutation:

$$\mathrm{R}_i(x_1, x_2) \rightarrow \mathrm{R}_j(x_2, x_1)$$

Its permutation table is given in Table 4.3(a). Again, RCC admits up to 8 different compositions tables but the one that is typically given (Cui *et al.*, 1993) is of the kind:

$$\mathrm{R}_i(x_1, x_2) \diamond \mathrm{R}_j(x_2, x_3) \rightarrow \mathrm{R}_k(x_1, x_3)$$

The corresponding composition table is reported in Table 4.3(b).

| DC | EC | PO | EQ | TPP | NTPP | TPPI | NTPPI |
|---|---|---|---|---|---|---|---|
| DC | EC | PO | EQ | TPPI | NTPPI | TPP | NTPP |

**(a)** Permutation table for RCC-8.

| | DC | EC | PO | EQ | TPP | NTPP | TPPI | NTPPI |
|---|---|---|---|---|---|---|---|---|
| DC | $\mathcal{B}_{\mathrm{RCC}}$ | DC, EC, PO, TPP, NTPP | DC, EC, PO, TPP, NTPP | DC | DC, EC, PO, TPP, NTPP | DC, EC, PO, TPP, NTPP | DC | DC |
| EC | DC, EC, PO, TPPI, NTPPI | DC, EC, PO, EQ, TPP, TPPI | DC, EC, PO, TPP, NTPP | EC | EC, PO, TPP, NTPP | PO, TPP, NTPP | DC, EC | DC |
| PO | DC, EC, PO, TPPI, NTPPI | DC, EC, PO, TPPI, NTPPI | $\mathcal{B}_{\mathrm{RCC}}$ | PO | PO, TPP, NTPP | PO, TPP, NTPP | DC, EC, PO, TPPI, NTPPI | DC, EC, PO, TPPI, NTPPI |
| EQ | DC | EC | PO | EQ | TPP | NTPP | TPPI | NTPPI |
| TPP | DC | DC, EC | DC, EC, PO, TPP, NTPP | TPP | TPP, NTPP | NTPP | DC, EC, PO, TPP, TPPI, NTPPI | DC, EC, PO, TPPI, NTPPI |
| NTPP | DC | DC | DC, EC, PO, TPP, NTPP | NTPP | NTPP | NTPP | DC, EC, PO, TPP, NTPP | $\mathcal{B}_{\mathrm{RCC}}$ |
| TPPI | DC, EC, PO, TPPI, NTPPI | EC, PO, TPPI, NTPPI | PO, TPPI, NTPPI | TPPI | PO, TPP, TPPI, NTPPI | PO, TPP, NTPP | TPPI, NTPPI | NTPPI |
| NTPPI | DC, EC, PO, TPPI, NTPPI | PO, TPPI, NTPPI | PO, TPPI, NTPPI | NTPPI | PO, TPPI, NTPPI | PO, TPP, EQ, NTPP, TPPI, NTPPI | NTPPI | NTPPI |

**(b)** Composition table for RCC-8.

**Table 4.3:** Reasoning tables for RCC-8.

**Inference Templates** Reasoning tables can be processed in order to detect single entries containing a base relation or series of entries containing disjunctive relations whose intersection yields a base relation. All such entries can be readily mapped onto so-called *inference templates*: inference paths defined over spatial constraints rather than spatial relations. Given a qualified spatial dataset, inference templates can be used to instantiate inference paths holding over its spatial relations in order to generate an inference graph.

**Example 4.10** (RCC Inference Templates) **-** First, let us consider Table 4.3(b). The entry individuated by the pair (NTPP, DC) gives rise to the inference template in Figure 4.6(a). The relations in the starting and terminal nodes indicate the relations among which an inference path can be instantiated according to this template. The variable pairs indicate the object equalities that have to hold among the spatial object pairs over which such relations occur. Finally, the intermediate node denotes the operation that generates the terminal relation from the starting ones, in this case C(omposition).

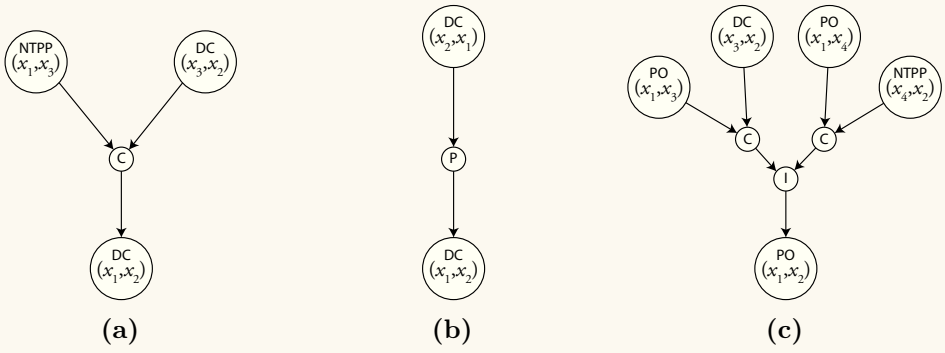



**Figure 4.6:** Three inference templates from the RCC reasoning tables.

The application of the above template over the qualified dataset reported in Figure 4.3(b) produces the inference path in Figure 4.5. Note that no other inference paths of this kind are admissible in this dataset. Although there are several DC relations, none of them occurs over object pairs such that both, relation and object equalities enforced by the template, are satisfied.

The same process can be applied to any other available reasoning table, for example, the first entry in Table 4.3(a) yields the inference template depicted in Figure 4.6(b), where the intermediate node stands for the P(permutation) operation.

Again, more complex inference templates can be identified by considering intersections of multiple disjunctive relations. For example, still from Table 4.3(b), the intersection of the entries identified by the pairs (PO, DC) and (PO, NTPP) yields the base relation PO. Accordingly, the inference template reported in Figure 4.6(c) can be generated where the intermediate nodes denote C(omposition) and I(ntersection) operations.

As seen in the above example, inferential nodes can be of three different types.

- **Permutational**: generated from permutation table, they have fixed in-degree and out-degree equal to 1. Incoming edges always originate in a relational node whereas outgoing edges always head toward intersection or relational nodes.

- **Compositional**: generated from composition table, they have fixed in-degree and out-degree: the first equal to 2, the latter to 1. Incoming edges always originate in a relational node whereas outgoing edges always head toward intersection or relational nodes.

- **Intersection**: generated from intersection of disjunctive relations that, in turn, result from composition or permutation operations. They have fixed in-degree and out-degree: the first equal to 2, the latter to 1. Incoming edges can originate in any kind of inferential node. Outgoing edges always head toward intersection or relational nodes.

According to such properties we have that any inference template (resp. path) is structured as follows: Moving from any tail node to the head, the first non-relational node encountered is either permutational or compositional. It necessarily follows a series of intersection nodes and, finally, the head of the inference template (resp. path).

**Definition 4.10** (Length of an Inference Graph) **-** An inference template (resp. path) is said to be of length $l$ if $l$ is the maximal number of inferential nodes encountered when moving from any relational node $n \in \mathcal{T}$ in its tail to its head $h$. Accordingly, an inference graph is said of length $l$ if its longest inference path has length equal to $l$.

From the properties of the inferential nodes and from the above definition it readily follows the following:

**Proposition 4.2 -** *An inference template (resp. path) of length $l$ contains $l-1$ intersection nodes and the cardinality of its tail $|\mathcal{T}|$ is such that: $l \leq |\mathcal{T}| \leq 2l$.*

Finally, we have that:

**Proposition 4.3 -** *Given a calculus $\mathcal{M} \in \boldsymbol{\mathcal{P}}$ defining $\mathsf{b} = |\mathcal{B}|$ base relations no inference templates (resp. paths) longer than $l = \mathsf{b}$ exists. Proof. A disjunctive relation consists at most of the disjunction of $\mathsf{b}$ base relations. As such a relation results from a permutational or compositional node, then, we need at most $\mathsf{b} - 1$ intersection nodes to filter out all but one base relation.* □

### 4.4.2 Data Structure

The basic set of data structures underlying the QSR-QSAM consists of $(i)$ relation tables and $(ii)$ template tables. Template tables are designed to maintain inference templates. The $i$-th template table $\text{TT}_i$ stores inference templates of length $i$ that are not contained in lower-indexed template tables. It contains entries of the form:

$$(\text{IN}_1, \ldots, \text{IN}_i, \text{OUT})$$

The IN elements are pairs of constraints where the second one is possibly empty. They stand for the composition or permutation operations that have to be performed and intersected to obtain OUT.

**Example 4.11 -** The inference templates in Figures 4.6(a) and 4.6(b) have length $l = 1$, thus are stored in the template table $\text{TT}_1$ reported in Table 4.4(a). Note that the second template is an instance of a permutation operation; accordingly the second element of the pair $\text{IN}_1$ is empty.

Similarly, the template in Figure 4.6(c) is of length $l = 2$ and is maintained in the template table $\text{TT}_2$ reported in Table 4.4(b).

| $\text{IN}_1$ | | OUT |
|---|---|---|
| $\text{NTPP}(x_1, x_3)$ | $\text{DC}(x_3, x_2)$ | $\text{DC}(x_1, x_2)$ |
| $\text{DC}(x_2, x_1)$ | - | $\text{DC}(x_1, x_2)$ |
| $\vdots$ | $\vdots$ | $\vdots$ |

**(a)**

| $\text{IN}_1$ | | $\text{IN}_2$ | | OUT |
|---|---|---|---|---|
| $\text{PO}(x_1, x_3)$ | $\text{DC}(x_3, x_2)$ | $\text{PO}(x_1, x_4)$ | $\text{NTPP}(x_4, x_2)$ | $\text{PO}(x_1, x_2)$ |
| $\vdots$ | $\vdots$ | $\vdots$ | $\vdots$ | $\vdots$ |

**(b)**

**Table 4.4:** Template tables containing inference templates of length $l = 1$ (a) and $l = 2$ (b).

### 4.4.3 Qualification

In Example 4.10 it was intuitively explained how to apply inference templates on a qualified dataset to obtain an inference graph. Now, a basic algorithmic procedure is to be presented since it is a fundamental part of the QSR-QSAM qualification function reported in Algorithm 4.9. A basic version of the procedure is reported in Algorithm 4.6 and is explained along with Example 4.12.

**Algorithm 4.6** Make-IG: Given a spatial dataset $O$ and a spatial calculus $\mathcal{M}$ constructs the inference graph $\mathcal{IG}$ of given length $l$.

**Input:**
$O$: spatial dataset
$\mathcal{M}$: qualitative spatial calculus equipped with reasoning tables
$l$: length of the inference graph to be generated
**Output:**
$\mathcal{IG}$: the inference graph of length $l$

1: **function** Make-IG($O$,$\mathcal{M}$,$l$)
2: $\mathcal{IG}_g = (\mathcal{N}_R, \mathcal{N}_I, \mathcal{A}_g)$;
3: $\mathcal{N}_R \leftarrow$ Qualify($O, \{\mathcal{M}\}$);
4: **for** $1 \leq i \leq l$ **do**
5: **for all** $\mathcal{IT} \in \text{TT}_i$ **do**
6: IPS $\leftarrow$ Get-By-Rels($\mathcal{N}_R, \mathcal{IT}$);
7: IPS $\leftarrow$ Filter-By-Out-Obj(IPS, $\mathcal{IT}$);
8: IPS $\leftarrow$ Filter-By-In-Obj(IPS, $\mathcal{IT}$);
9: **for all** $\mathcal{IP} \in$ IPS **do**
10: $n_{last} \leftarrow 0$;
11: **for all** IN $\in \mathcal{IP}$ **do**
12: **if** $n_{last} \neq 0$ **then**
13: $n_{\text{I}} \leftarrow$ AddIntersectionNode($\mathcal{IG}_g$);
14: AddEdge($\mathcal{IG}_g, (n_{last}, n_{\text{I}})$);
15: $n_{last} \leftarrow n_{\text{I}}$;
16: **if** IN.$second = 0$ **then**
17: $n_{\text{P}} \leftarrow$ AddPermutationalNode($\mathcal{IG}_g$);
18: AddEdge($\mathcal{IG}_g$, (IN.$first, n_{\text{P}}$));
19: **if** $n_{last} \neq 0$ **then** AddEdge($\mathcal{IG}_g, (n_{\text{P}}, \rightarrow n_{last})$);
20: **else** $n_{last} \leftarrow n_{\text{P}}$;
21: **else**
22: $n_{\text{C}} \leftarrow$ AddCompositionalNode($\mathcal{IG}_g$);
23: AddEdge($\mathcal{IG}_g$, IN.$first \rightarrow n_{\text{C}}$);
24: AddEdge($\mathcal{IG}_g$, IN.$second \rightarrow n_{\text{C}}$);
25: **if** $n_{last} \neq 0$ **then** AddEdge($\mathcal{IG}_g, (n_{\text{C}}, n_{last})$);
26: **else** $n_{last} \leftarrow n_{\text{C}}$;
27: AddEdge($\mathcal{IG}_g, (n_{last}, \mathcal{IP}$.OUT));
28: **return** $\mathcal{IG}_g$;

**Example 4.12** (Building the Inference Graph) **-** The procedure Make-IG requires as input (*i*) a spatial dataset $O$, (*ii*) a calculus $\mathcal{M}$, and (*iii*) an integer $l$. The output is an inference graph $\mathcal{IG}_g$ of length $l$. Let us consider the case that the spatial dataset is the one depicted in Figure 4.3(a), $\mathcal{M} =$ RCC and $l = 2$. The resulting inference graph is depicted in Figure 4.7.

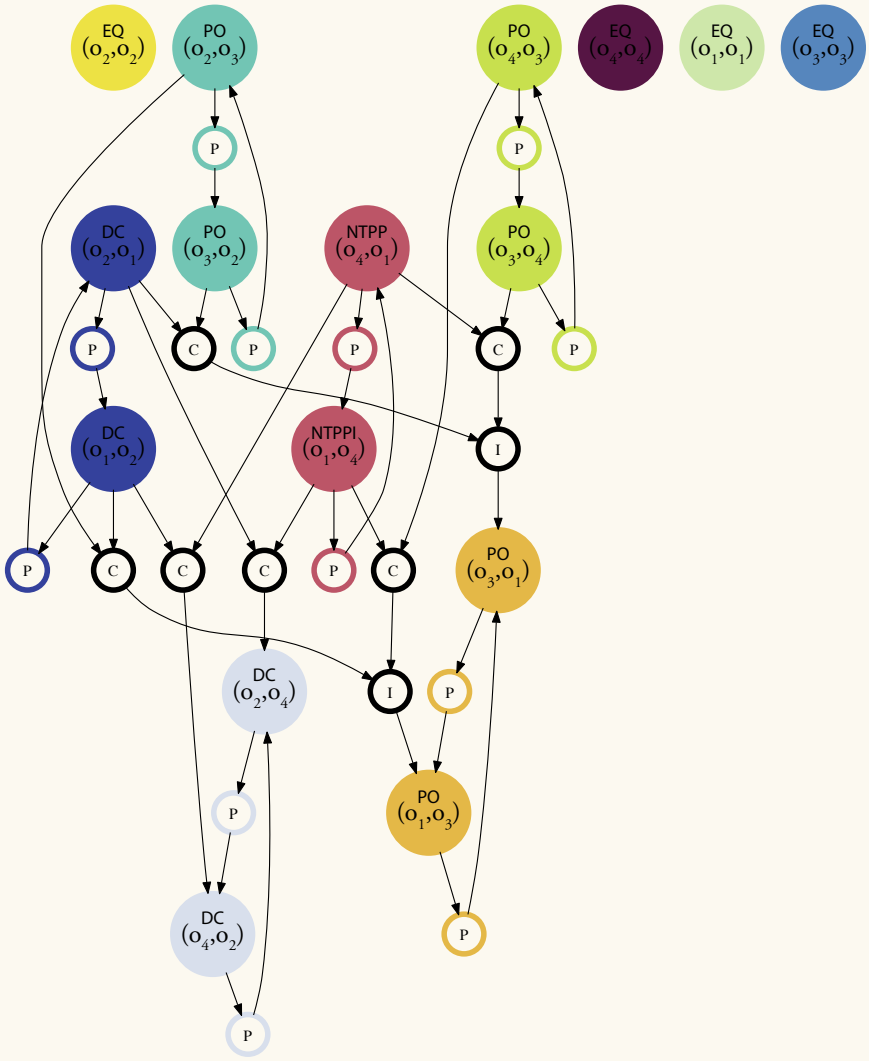



**Figure 4.7:** RCC inference graph of length $l = 2$ for the dataset in Figure 4.3(a).

First off, the procedure RCC-qualifies the spatial dataset by means of the function QUALIFY. Such a function works exactly as the one reported in Algorithm 4.2 with the only difference that the qualified dataset is returned instead of being stored in the relation tables. The qualified dataset corresponds to the relational node set $\mathcal{N}_R$.

Next, for any integer $1 \leq i \leq l$ the procedure iterates over the inference templates of length $i$ stored in the template table $\text{TT}_i$. From each inference template $\mathcal{IT}$ a set IPS of inference paths is generated according to three main steps. Let us go through such steps while considering the inference template appearing in the first line of Table 4.4(b).

($i$) The function GET-IP-BY-RELS prepares a vector for each relation appearing in the template. Then it scans the whole relational node set $\mathcal{N}_R$ comparing the relation of each node against the relations appearing in the template. When a match is found the relational

node is placed in the corresponding vector. Finally the Cartesian product of all the vectors is computed and returned in the following form:

| IN$_1$ | | OUT |
|---|---|---|
| $first$ | $second$ | |
| NTPP$(o_4, o_1)$ | DC$(o_1, o_2)$ | DC$(o_1, o_2)$ |
| NTPP$(o_4, o_1)$ | DC$(o_1, o_2)$ | DC$(o_2, o_1)$ |
| NTPP$(o_4, o_1)$ | DC$(o_1, o_2)$ | DC$(o_2, o_4)$ |
| NTPP$(o_4, o_1)$ | DC$(o_1, o_2)$ | DC$(o_4, o_2)$ |
| NTPP$(o_4, o_1)$ | DC$(o_2, o_1)$ | DC$(o_1, o_2)$ |
| NTPP$(o_4, o_1)$ | DC$(o_2, o_1)$ | DC$(o_2, o_1)$ |
| NTPP$(o_4, o_1)$ | DC$(o_2, o_1)$ | DC$(o_2, o_4)$ |
| NTPP$(o_4, o_1)$ | DC$(o_2, o_1)$ | DC$(o_4, o_2)$ |
| NTPP$(o_4, o_1)$ | DC$(o_2, o_4)$ | DC$(o_1, o_2)$ |
| NTPP$(o_4, o_1)$ | DC$(o_2, o_4)$ | DC$(o_2, o_1)$ |
| NTPP$(o_4, o_1)$ | DC$(o_2, o_4)$ | DC$(o_2, o_4)$ |
| NTPP$(o_4, o_1)$ | DC$(o_2, o_4)$ | DC$(o_4, o_2)$ |
| NTPP$(o_4, o_1)$ | DC$(o_4, o_2)$ | DC$(o_1, o_2)$ |
| NTPP$(o_4, o_1)$ | DC$(o_4, o_2)$ | DC$(o_2, o_1)$ |
| NTPP$(o_4, o_1)$ | DC$(o_4, o_2)$ | DC$(o_2, o_4)$ |
| NTPP$(o_4, o_1)$ | DC$(o_4, o_2)$ | DC$(o_4, o_2)$ |

Each line represents a *candidate* inference path according to the relation symbols appearing in the template. Such result set has to be filtered exploiting the equality constraints enforced on the object pairs by the template. This is done in the two remaining steps of the analyzed process.

(*ii*) The function Filter-By-Out-Obj exploits the equality constraints enforced by the output relation to perform a first refinement of ips. Analyzing the template at hand we have that the first object appearing in the relation IN$_1$.$first$ has to be equal to the first object in the output relation. Similarly, the second object in the relation IN$_1$.*second* has to be equal to the second object of the output relation. The result of the filtering operation is the following:

| IN$_1$ | | OUT |
|---|---|---|
| $first$ | $second$ | |
| NTPP$(o_4, o_1)$ | DC$(o_1, o_2)$ | DC$(o_4, o_2)$ |
| NTPP$(o_4, o_1)$ | DC$(o_4, o_2)$ | DC$(o_4, o_2)$ |

(*iii*) The function Filter-By-In-Obj behaves similarly. It exploits the equality constraints enforced by the template among the object pairs of the input relations to rule out invalid inference paths. In the considered case, the second object appearing in the relation IN$_1$.$first$ has to be equal to the first object in the relation IN$_1$.*second*. The result of the filtering operation is the following:

| $\text{IN}_1$ | | OUT |
|---|---|---|
| $first$ | $second$ | |
| $\text{NTPP}(o_4, o_1)$ | $\text{DC}(o_1, o_2)$ | $\text{DC}(o_4, o_2)$ |

Now, the set IPS contains all the inference paths induced on the relational node set $\mathcal{N}_R$ by the template $\mathcal{IT}$ at hand. The remaining step consists in using each such inference path to populate the inferential node set $\mathcal{N}_I$ and edge set $\mathcal{A}_g$ of the graph $\mathcal{IG}_g$. For each path $\mathcal{IP} \in$ IPS the procedure iterates over the input relational nodes $\text{IN}_1, \ldots, \text{IN}_i$. In our exemplary case there is only one pair $\text{IN}_1$. Since both, $\text{IN}_1.first$ and $\text{IN}_1.second$ are not null, we are dealing with a composition operation, therefore the procedure skips to line 22. The function AddCompositionalNode adds a compositional node to the graph, specifically to the inferential node set $\mathcal{N}_I$. Then, the edges $(\text{NTPP}(o_4, o_1), \text{C})$ and $(\text{DC}(o_1, o_2), \text{C})$, going from the relational nodes indicated in the pair $\text{IN}_1$ to the fresh generated node, are added to the edge set by means of the function AddEdge. Finally, the compositional node is saved in the variable $n_{last}$ which always contains the inferential node farthest from the input relational nodes. Such a variable is used to generate intersection nodes and to ensure they are correctly connected in the inference path structure.

After having iterated over all the input pairs, the procedure completes the path generating the last edge going from $n_{last}$ to the output relational node stored in $\mathcal{IP}$.OUT.

Now that the problem of building an inference graph has been tackled, let us focus on the reduction strategy: Given a calculus from the pool $\mathcal{M} \in \boldsymbol{\mathcal{P}}$ and the corresponding inference graph $\mathcal{IG}$ of length $l$ we want to produce a qualified dataset $\mathcal{R}_{\mathcal{M},\mathcal{IG}}(O)$ reduced with respect to the full one $\mathcal{R}_{\mathcal{M}}(O)$ and such that, at retrieval time, it is possible to rebuild all the missing relations $\mathcal{R}_{\mathcal{M}}(O) \setminus \mathcal{R}_{\mathcal{M},\mathcal{IG}}(O)$ by only resorting to QSR.

The first step is to find in $\mathcal{IG}$ groups of nodes reachable from each other:

**Definition 4.11** (Relaxed Strongly Connected Component) **-** Let $\mathcal{H} = (\mathcal{N}, \mathcal{A})$ be a B-graph. A *relaxed strongly connected component* (RSCC) $C = (\mathcal{N}_C, \mathcal{A}_C)$ is a subhypergraph of $\mathcal{H}$ with $\mathcal{N}_C = \{n_1, \ldots, n_p\}$ such that the following conditions hold simultaneously:

- for each node $n_i \in \mathcal{N}_C$ in the component there exist a hyperpath $\Pi_{S_i,n_i}$ ending in $n_i$ whose source node set $S_i \subseteq \mathcal{N}_C \setminus \{n_i\}$ is completely contained in the component;
- the component node set is equal to the union of source node sets of the above hyperpaths: $\bigcup_{i=1}^{p} S_i = \mathcal{N}_C$
- for each node $n_j \in \mathcal{N}_C$ in the component there exists at least a hyperpath $\Pi_{S_i,n_i}$ to another node $n_i \in \mathcal{N}_C$ having $n_j \in S_i$ in its source node set.

Practically, a RSCC is a maximal subhypergraph of $\mathcal{IG}$ such that each node can be reached by at least a subset of the other nodes in the component and such that each node takes part in the source node set of at least a hyperpath to another node in the component. Note that the above definition shares some properties with hypercycles (cf. Definition 2.7), strongly connected components

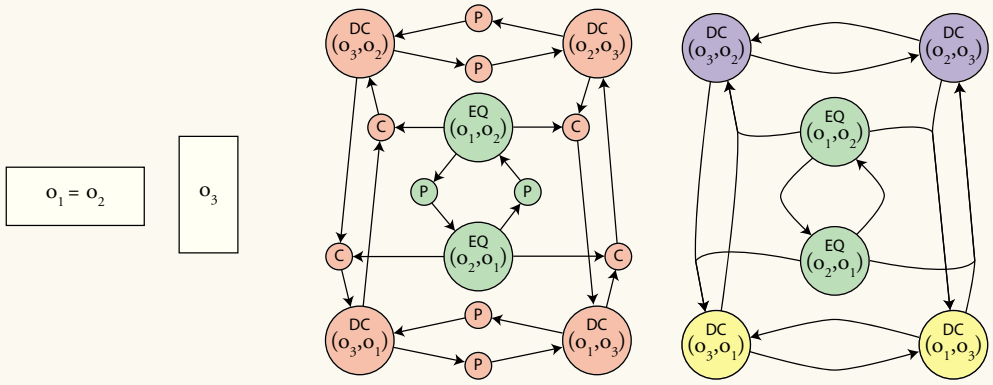
(cf. Definition 2.9) and weakly connected components (cf. Definition 2.10) but it does not coincides with any of them. At the best of the author's knowledge, this is a novel concept in hypergraphs. More precisely, we have that, these concepts are related according to the following schema

$$\text{SCC} \Longrightarrow \text{RSCC} \Longrightarrow \text{WSCC}$$

and that a RSCC contains at least a hypercycle.

More interestingly, a RSCC in $\mathcal{IG}$ is always a SCC in $\mathcal{IG}_g$. In addition, a SCC $C_g = (\mathcal{N}_g, \mathcal{A}_g)$ in $\mathcal{IG}_g$, with $\mathcal{N}_g = \{\mathcal{N}_R, \mathcal{N}_I\}$, corresponds to a RSCC $C = (\mathcal{N}, \mathcal{A})$ in $\mathcal{IG}$ if none of the inference paths associated with the inferential nodes in $\mathcal{N}_I$ has tail nodes outside $C_g$. Let us show this by means of the following:

**Example 4.13 -** Let us consider the spatial dataset in Figure 4.8(a). Note that it consists of three objects $o_1$,$o_2$ and $o_3$ and that $o_1$ and $o_2$ coincide. The corresponding inference graph $\mathcal{IG}_g$ of length 2 obtained from the RCC reasoning tables in Table 4.3 is reported in Figure 4.8(b). Figure 4.8(c) depicts the hypergraph representation $\mathcal{IG}$. The red nodes in $\mathcal{IG}_g$ are one SCC in the graph, whether the green ones stand for another component. Note that none of the inferential nodes in the green component have incoming arcs originating outside the component. In fact it corresponds to a RSCC in $\mathcal{IG}$. Conversely, the red component does not correspond to a RSCC in $\mathcal{IG}$ since all of the compositional nodes have in-arcs coming from the green component. Actually, the red SCC splits into two different RSCC in $\mathcal{IG}$: blue and yellow.




**(a)** Spatial dataset **(b)** $\mathcal{IG}_g$ **(c)** $\mathcal{IG}$

**Figure 4.8:** Relation between the strongly connected components in an inference graph and those in its graph representation.

Consequenlty, RSCCs can be generated according to Algorithm 4.7. First, the procedure computes the SCCs in $\mathcal{IG}_g$ using a standard graph algorithm, e.g. Tarjan (1971) which runs in $\mathcal{O}(|\mathcal{IG}_g|)$ worst case time. The obtained components are subsequently analyzed to make sure they correspond to RSCCs in the hypergraph: All the inference paths taking part in a component that have their tails originating in a different one are removed from the SCC. Then, the strongly connected

components are computed again on the residual subgraph. The last two steps are repeated until a fixed point is reached: all the SCCs in $\mathcal{IG}_g$ are also RSCCs in $\mathcal{IG}$. Note that a fixed point is always reached: in the worst case the algorithm keeps decomposing a component until the decomposition yields single nodes that are trivial RSCC instances.

Once all the RSCCs have been found, we can condensate (cf. Example 2.2) the inference graph. We call the condensation:

**Definition 4.12** (Inference Kernel) **-** Let $\mathcal{IG} = (\mathcal{N}, \mathcal{A})$ be an inference graph. The *inference kernel* $\mathcal{IK}(\mathcal{IG}) = (\mathcal{N}_{\mathcal{IK}}, \mathcal{A}_{\mathcal{IK}})$ of $\mathcal{IG}$ is the B-graph obtained by contracting each RSCC of $\mathcal{IG}$ into a single node in $\mathcal{IK}(\mathcal{IG})$ and removing duplicate hyperarcs. Conversely, a subset $\underline{\mathcal{N}_{\mathcal{IK}}} \subseteq \mathcal{N}_{\mathcal{IK}}$ of the kernel nodes identifies in $\mathcal{IG}$ the subhypergraph induced by the nodes in the corresponding RSCCs and denoted by $\mathcal{IG}(\underline{\mathcal{N}_{\mathcal{IK}}})$.

**Algorithm 4.7** RELAXED-STRONGLY-CONNECT: Computes the relaxed strongly connected components for a given inference graph.

**Input:**
$\mathcal{IG}_g = (\mathcal{N}_g, \mathcal{A}_g)$: the graph representation of an inference graph
**Output:**
RSCC$s$: relaxed strongly connected components in $\mathcal{IG}_g$

```
 1: function RELAXED-STRONGLY-CONNECT(IG_g)
 2:     SCCs ← STRONGLY-CONNECT(IG_g);                    ▷ computes SCCs
 3:     RSCCs ← RELAX-COMPONENTS(SCCs);
 4:     return RSCCs;

 5: function RELAX-COMPONENTS(SCCs)
 6:     RSCCs ← ∅;
 7:     for all C = (N_C, A_C) ∈ SCCs do
 8:         REMOVED ← ∅;
 9:         for all n ∈ N_C do
10:             if n is inferential and SON(n) is relational then
11:                 ANCESTORS ← GET-ANCESTORS(n);
12:                 TO-REMOVE ← FALSE;
13:                 for all n_ANC ∈ ANCESTORS do
14:                     if n_ANC ∉ N_C then TO-REMOVE ← TRUE;
15:                 if TO-REMOVE then
16:                     REMOVED ← REMOVED ∪ REMOVE-INF-PATH(C, n);
17:         if REMOVED ≠ ∅ then
18:             SCCs_C ← STRONGLY-CONNECT(C);
19:             RSCCs ← RSCCs ∪ RELAX-COMPONENTS(SCCs_C);
20:             PUT-INF-PATHS-BACK(C, REMOVED);
21:         else
22:             RSCCs ← RSCCs ∪ C;
23:     return RSCCs;
```

Now, it has to be noted that, given an inference kernel hyperarc $a = (\mathcal{T}, h) \in \mathcal{A}_{\mathcal{IK}}$, we have that all the relations corresponding to the nodes in $\mathcal{IG}(h)$ can be inferred through reasoning operations by the relational nodes in $\mathcal{IG}(\mathcal{T})$. Accordingly, given a series of hyperpaths spanning the whole kernel node set $\mathcal{N}_{\mathcal{IK}}$, the source node sets of such hyperpaths suffice to infer all the other nodes. To find the source nodes we propose Algorithm 4.8 that runs in $\mathcal{O}(|\mathcal{IG}|)$.

**Algorithm 4.8** GetSources: Computes a set of hyperpaths spanning the node set of a given B-graph $\mathcal{H}$ and returns the source node set.

**Input:**
$\mathcal{H} = (\mathcal{N}, \mathcal{A})$: B-graph
**Output:**
$\mathcal{S}$: set of source nodes from which is possible to reach all the others

```
1: function GetSources(H)
2:     S; ▷ global array
3:     ToVisit; ▷ global array of size |N|
4:     ON; ▷ array of nodes ordered according to the number of outgoing arcs
5:     while ON is not empty do
6:         s ← ON.pop();
7:         if ToVisit[s] then
8:             S.push(s);
9:             Visit(s, H)
10:    return S;

11: function Visit(n, H)
12:    ToVisit[n] ← false;
13:    for all outgoing a_o do
14:        if ToVisit[h(a_o)] and h(a_o) ∉ S then
15:            for all n_t ∈ T(a_o) \ n do
16:                if ToVisit[n_t] and n_t ∉ S then
17:                    S.push(n_t);
18:            Visit(h(a_o), H);
19:    return ;
```

The same principle applies to each source RSCC: it is possible to find a set of source relational nodes that reach all the other nodes in the component. Accordingly, storing only the relation corresponding to such nodes is enough to rebuild, at runtime, (*i*) the relation corresponding to the nodes in the same component and (*ii*) the other components.

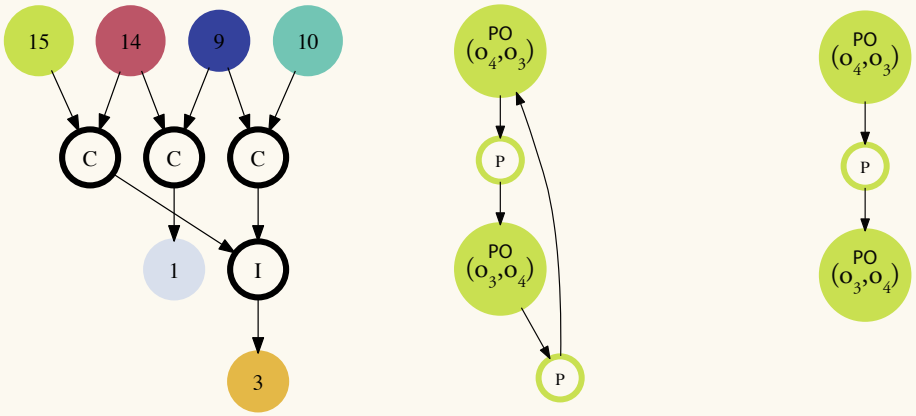
The whole qualification procedure, as explained above, is reported in Algorithm 4.9. Example 4.14 traces the execution for the inference graph obtained from the previous examples.

**Example 4.14** (QSR-qualify) **-** Let us consider again the spatial dataset in Figure 4.3(a) and assume that $\mathcal{P} = \{\text{RCC}\}$. The corresponding inference graph $\mathcal{IG}$ of length 2 is depicted in Figure 4.7. It was obtained automatically through the implementation of Algorithm 4.6[1].

Now, let us go through the execution of Algorithm 4.9. The function KERNEL is assumed to compute the RSCC of the inference path according to Algorithm 4.7 and, from them, to generate the condensation $\mathcal{IK}(\mathcal{IG})$. The graph version of the inference kernel for this example is depicted in Figure 4.9(a). Note that, beyond a numerical id, for the sake of visual clarity, each kernel node is associated a unique color. Such a color corresponds to that associated to nodes belonging to the same RSCC in the inference graph of Figure 4.7.

The call GETSOURCES($\mathcal{IK}$) identifies the source nodes $\mathcal{S}_{\mathcal{IK}} = \{15, 14, 9, 10\}$ of the kernel. Each such source node $n \in \mathcal{S}_{\mathcal{IK}}$ is used to access the corresponding RSCC in $\mathcal{IG}$ and to identify its source relational nodes. For instance, Figure 4.9(b) shows the RSCC corresponding to the leftmost kernel node—numbered 15. In this case, all the relational nodes have equal out-degree, therefore the visiting procedure used by GETSOURCES can start the visit from any of them. Assuming it starts from the node PO$(o_4, o_3)$ it finds the spanning hyperpath reported in Figure 4.9(c). Accordingly, by calling GETSOURCES($\mathcal{IG}(15)$), the source node set $\mathcal{S} = \{\text{PO}(o_4, o_3)\}$ is identified.

Finally, each relation corresponding to the nodes (only one in this case) in $\mathcal{S}$ is stored in the opportune relation table according to its arity—2 in this example.




**(a)** Inference kernel. **(b)** RSCC number 15. **(c)** Hyperpath spanning RSCC 15.

**Figure 4.9:** Reducing the inference graph in Figure 4.7 through Algorithm 4.9.

[1]Graph visualization is done through Graphviz (http://www.graphviz.org/)

**Algorithm 4.9** Qualify-QSR: Populates the relation tables in $\mathcal{D}$ with the relations induced by the spatial dataset and reduced according to the inference graph $\mathcal{IG}$ of length $l$

**Input:**
$O$: spatial dataset
$\boldsymbol{\mathcal{P}}$: calculus pool
$l$: length of the inference graph to be generated
**Output:**
LUTs in $\mathcal{D}$ are filled with relations from the $\boldsymbol{\mathcal{P}}$-qualified dataset $\mathcal{R}_{\boldsymbol{\mathcal{P}}}(O)$

```
1: function QUALIFY-QSR(O, P, l)
2:     for all M ∈ P do
3:         IG ← MAKE-IG(O, M, l);
4:         IK ← KERNEL(IG);                    ▷ compute the kernel hypergraph of IG
5:         S_IK ← GETSOURCES(IK);              ▷ retrieve the sources of K
6:         for all n ∈ S_IK do
7:             S ← GETSOURCES(IG(n));          ▷ retrieve the sources of a RSCC
8:             for all R(ω) ∈ S do
9:                 a ← |ω|;
10:                RT_a ← RT_a ⋃ (R, ω);
11:    return ;
```

Algorithm 4.9 produces an *$\mathcal{IG}$-reduced, $\boldsymbol{\mathcal{P}}$-qualified dataset* $\mathcal{R}_{\boldsymbol{\mathcal{P}},\mathcal{IG}}(O)$ such that

$$\mathcal{R}_{\boldsymbol{\mathcal{P}},\mathcal{IG}}(O) = \bigcup_{\mathcal{M}\in\boldsymbol{\mathcal{P}}} \mathcal{R}_{\mathcal{M},\mathcal{IG}}(O)$$

where $\mathcal{R}_{\mathcal{M},\mathcal{IG}}(O)$ is a subset of the full qualified dataset $\mathcal{R}_{\mathcal{M}}(O)$ generated for the single calculus $\mathcal{M}$ and such that

$$|\mathcal{R}_{\mathcal{M},\mathcal{IG}}(O)| = |\mathcal{R}_{\mathcal{M}}(O)| - \rho \text{ with } \rho \geq 0$$

The cardinality of $\mathcal{R}_{\mathcal{M},\mathcal{IG}}(O)$ grows inversely with the length $l$ of the inference graph $\mathcal{IG}$. In particular we have that, if the graph length $l = 0$ is equal to zero the edge set is empty and each node is a component on itself. Accordingly the performed reduction $\rho = 0$ is also equal to zero.

If $l > 0$, the amount of the reduction depends on the structure of the reasoning tables coming with the calculus $\mathcal{M}$ as well as on the arrangement of the objects in the spatial dataset—i.e. the relations in the qualified dataset. Note that the structure of the reasoning tables reflects, in turn, the algebraic properties of the calculus: a calculus with weak properties tends to have reasoning tables containing many disjunctive relations. As a consequence, the stronger the properties of a calculus, the higher the number of templates of a given length $l$.

The mere existence of an inference template does not guarantee a reduction

since the qualified dataset may not induce any instance of it in $\mathcal{IG}$. However, the higher the number of inference templates, the higher the probability to instantiate inference paths from them. The number of inference templates for a calculus can be augmented equipping it with multiple reasoning tables.

**Example 4.15** (Number of Inference Templates for RCC) **-** The number of inference templates generable from Tables 4.3(a) and 4.3(b) is reported in Table 4.5 according to the length $l$. Note that, for $l > 6$ no new templates are generable from such tables.

| Lenght | Inference Templates |
|---|---|
| 1 | 35 |
| 2 | 160 |
| 3 | 1867 |
| 4 | 8483 |
| 5 | 17079 |
| 6 | 19287 |
| 7 | 19287 |
| 8 | 19287 |

**Table 4.5:** Number of inference templates for RCC.

The cardinality of a $\mathcal{IG}$-reduced, $\mathcal{M}$-qualified dataset also depends on the detection of spanning hyperpaths: The smaller the cardinality of the source node set of the hyperpaths spanning the inference kernel, the higher the reduction. The same applies to hyperpaths spanning source RSCCS in the inference graph.

Algorithm 4.8 is a greedy algorithm, thus it does not guarantee to find a global optimum. Indeed, the procedure GETSOURCES assumes that the array of nodes to visit is ordered according to their out-degree that is a realization of the following heuristics: the smaller the out-degree of a node, the smaller the probability that hyperpaths having such a node in their source node set can cover the full hypergraph. Such a heuristics does not guarantee to produce a minimal number of spanning hyperpaths. This, in turn, implies that the cardinality of the source node set may not be minimal and, eventually, that the reduction is not guaranteed to be maximal.

Maximal reduction can be guaranteed by slightly modifying the procedure GETSOURCES in such a way that it produces *all* the sets of possible spanning hyperpaths in order to pick one with minimal source node set cardinality. Such a modification obviously yields a computational overhead due to the increased number of hyperpaths that has to be generated.

Finally, if a calculus $\mathcal{M}$ is provided with as many reasoning tables as necessary to generate *any* possible inference template for that calculus, the $\mathcal{IG}$-reduced, $\mathcal{M}$-qualified dataset corresponds to the *minimum* relation set which fully describes a spatial dataset. If this would not be the case it means that $\mathcal{R}_{\mathcal{M},\mathcal{IG}}(O)$ contains

relations that can be inferred from other relations. This is obviously impossible given that the algorithm is designed to detect all the inferable relations.

### 4.4.4 Retrieval

The basic retrieval function for QSR-QSAM draws upon standard QSR techniques. Given its constructive properties, the $\mathcal{IG}$-reduced, $\boldsymbol{\mathcal{P}}$-qualified dataset $\mathcal{R}_{\boldsymbol{\mathcal{P}},\mathcal{IG}}(O)$ contains enough spatial relations to rebuild all the missing ones by only applying symbolic reasoning.

Accordingly, the algorithm to solve a QSCQ consists of two main steps: for each calculus $\mathcal{M}$ involved in the query (*i*) run the *algebraic closure* algorithm (cf. Section 2.2.5) on $\mathcal{R}_{\mathcal{M},\mathcal{IG}}(O)$ to rebuild the full qualified dataset $\mathcal{R}_{\mathcal{M}}(O)$. (*ii*) Solve the query with the basic retrieval procedure reported in Algorithm 3.1 with the function ExtendPMatching realized as in Algorithm 4.3

### 4.4.5 Final Remarks and Discussion

QSR-based QSAM (QSR-QSAM) makes use of the inference graph (cf. Definition 4.9) to identify inferential dependencies among the relations in a full qualified dataset $\mathcal{R}_{\boldsymbol{\mathcal{P}}}(O)$. Inferable relations are not stored since they can be rebuilt at retrieval time via qualitative reasoning techniques.

In Section 4.4.3 a basic qualification procedure has been presented that produces reduced, but not minimum, qualified dataset. It was pointed out that such a procedure can be slightly modified to compute any possible set of spanning hyperpaths. This would allow for selecting the set with the smallest source node set and, thus, for producing minimal qualified datasets.

The problem of finding *minimum qualitative representations* has been raised, for example, in (Egenhofer & Sharma, 1992, 1993) where it is claimed that consistency checking algorithms can be used to find minimum topological representations. The idea of Egenhofer & Sharma is to remove the arcs of a complete Qualitative Constraint Network (QCN) one by one until it becomes ambiguous. In this way it is possible to identify the smallest possible set of relations forming a consistent description for the scene. In (Egenhofer & Sharma, 1993) the identification of minimal relation set is obtained as a side-product of consistency and the algorithm to find it is obviously not optimized. The qualification procedure we presented, instead, is purposely designed for reducing a qualified dataset, i.e. the arcs in a QCN. In Section 4.4.3 it was also argued that, if a calculus is provided with as many reasoning tables as necessary to generate *any* possible inference template, Qualify-QSR generates a qualified dataset with *absolute minimal* cardinality, i.e. a *minimum qualitative representation*.

In Section 4.4.1.1 it was shown how inference templates can be automatically generated from reasoning tables. However, it has to be noted that the set of inference templates can be enriched with manually-defined ones. In this way, for example, one can introduce inter-calculi inferential dependencies that allow

for unifying inference graphs generated for different calculi and, accordingly, for producing even more compact $\boldsymbol{\mathcal{P}}$-qualified datasets. For example, one might observe that if $o_1$ is *North* of $o_2$ then it has to be that $o_1$ and $o_2$ are *Disconnected*: $\text{N}(o_1, o_2) \rightarrow \text{DC}(o_1, o_2)$ and $\text{N}(o_1, o_2) \rightarrow \text{DC}(o_2, o_1)$. Of course this requires to define new types of inferential nodes. For example, in this case one might define a new type named *implication* and manually add to the template tables a series of templates based on this operation. In his Ph.D thesis, Sharma (1996) developed a series of inter-calculus reasoning tables that can be exploited for this purpose.

Lastly, it is worth to remark that the strategies presented in the previous sections are basic strategies that can be improved in several ways. For example, the set of data structures underlying the QSAM can be extended to allow for memorizing which inference templates are used to perform the reduction. The retrieval approach can exploit this information to "guide" the algebraic closure algorithm during the reconstruction phase.

## 4.5 Summary

In this chapter we defined Qualitative Spatial Access Methods (QSAMs): a typology of Spatial Access Method (SAM) suited for solving Qualitative Spatial Configuration Queries (QSCQs). A QSAM has been formalized (cf. Definition 4.1) as a triple $(q, \mathcal{D}, r)$ where $q$ is a *dataset qualifier function* that populates the set $\mathcal{D}$ of data structures underlying the QSAM and $r$ is a *retriever function* that exploits the information stored in $\mathcal{D}$ to solve a given QSCQ.

Four kinds of QSAMs have been developed: Functional QSAM (F-QSAM), Qualitative Storage Layer QSAM (QSL-QSAM), Spatial Clustering QSAM (SC-QSAM), and QSR-based QSAM (QSR-QSAM). They all draw upon the basic retrieval strategy presented in Section 3.4.1, but tackle differently the dataset qualification problem (cf. Section 3.5).

F-QSAM (cf. Section 4.1) encodes the runtime qualification strategy presented in Section 3.5.1. Accordingly, it has a void qualifier function and an empty set of data structures. The retriever function qualifies the dataset at QSCQ execution time. Such a solution provides the best option in terms of storage space but is extremely costly with respect to retrieval time.

QSL-QSAM (cf. Section 4.2) encodes the pre-qualification strategy presented in Section 3.5.2. The qualifier function performs a full qualification of the spatial dataset and stores the resulting qualitative relations into a set of appositely designed Lookup Tables (LUTs) called *relation tables*. Such an approach allows for transforming QSCQ solving into a series of look-up operations. Although allowing for high retrieval time performance, this solution presents two main drawbacks: (*i*) It requires an enormous quantity of extra storage space needed to maintain the qualified dataset. (*ii*) The high computational cost demanded by a full pre-qualification makes hard to deal with the dynamism of the real world. Indeed,

by the time a spatial dataset is qualified, the produced qualified dataset might be already out-of-date, calling for a new pre-qualification.

The SC-QSAM family is introduced in Section 4.3. It draws upon the joint exploitation of a special type of spatial relations called Clustering Relations together with a spatial clustering index (cf. Definition 4.5). If opportunely tuned, the spatial clustering index produces a subdivision of the spatial dataset that allows the qualifier function to compute and store a reduced number of qualitative relations. Beyond the relation tables, the set of data structures underlying a SC-QSAM contains other data structures designed to represent the spatial clustering index used to make the subdivision. At QSCQ execution time, the retriever function exploits the index structure to rebuild the qualitative relations missing in the relation tables. This approach provides a valuable compromise between the previous solutions: it shorten the retrieval time with respect to F-QSAM as well as the pre-qualification time and produces more compact qualified datasets. The main shortcoming of this approach is that it requires a careful calibration of the spatial clustering index parameters. Indeed, as shown in Section 4.3.3, a "bad" index may yield worse space-time performance than a full pre-qualification. Consequently this solution requires a mindful tuning of the index parameters that also depends on the spatial calculi in the calculus pool $\boldsymbol{\mathcal{P}}$.

Finally QSR-QSAM is presented in Section 4.4. It exploits a novel data structure named inference graph (cf. Definition 4.9) to obtain a reduced qualified dataset from which it is possible to infer back all the missing relations by only resorting to symbolic reasoning. The advantages of this strategy are that it is independent of the spatial calculus and it might be used to produce the minimum set of qualitative relations needed to describe a spatial dataset (cf. discussion on the reduction at the end of Section 4.4.3). The main drawbacks are that, in order to perform the reduction, QSR-QSAM requires a full pre-qualification plus the instantiation of an inference graph. This is a very costly operation, that make this solution unsuitable for being applied on a full spatial dataset.

A possible alternative that takes advantage of both SC-QSAM and QSR-QSAM is described in the Outlook section of Chapter 7.

Chapter 5

# Developing and Benchmarking Qualitative Spatial Access Methods

In this chapter we tackle the problem of developing and benchmarking Qualitative Spatial Access Methods (QSAMs) from a practical perspective. The chapter is split into two main parts: Section 5.1 introduces a software framework called MyQual whose main purpose is to provide a development, testing, and production environment for Spatial Access Methods (SAMs) suited for qualitative spatial queries, with a particular focus on QSAMs and Qualitative Spatial Configuration Queries (QSCQs). MyQual has been used to implement, test and compare all of the QSAMs presented in Chapter 4 (cf. Chapter 6 for the experimental results).

As explained in the previous chapter, the Spatial Clustering QSAM (SC-QSAM) family (cf. Section 4.3) requires a careful choice of the spatial clustering index as well as a mindful tuning of its parameters according to the spatial calculi that have to be treated. Section 5.2 is devoted to present the implementation of two instances of the SC-QSAM family.

## 5.1 MyQual: an Extensible Development and Benchmark Framework

MyQual is a novel software framework realized in the scope of this thesis to facilitate the development of SAMs for qualitative spatial queries (cf. Section 3.1). It has been designed as an extension for spatially-enabled PostgreSQL[1] databases

[1] http://www.postgresql.org/

(Douglas & Douglas, 2005). The spatial extension for PostgreSQL is provided by a software layer called PostGIS[1] (Ramsey, 2005).

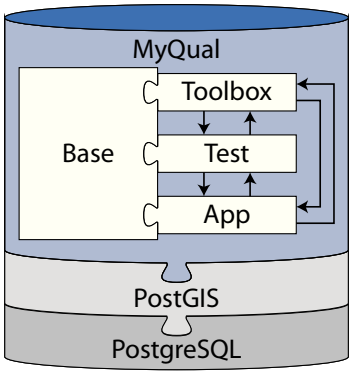



**Figure 5.1:** MyQual, logical framework overview

As depicted in Figure 5.1 MyQual lies upon PostGIS and consists of four main components:

- MyQual-Base: provides management functionalities;
- MyQual-Toolbox: allows for extensibility;
- MyQual-Test: provides a test environment;
- MyQual-App: provides a production environment.

Each component raises a new schema in the database where all the tables and functions required from the component itself are installed. This design choice allows for maintaining a lean interaction with the underlying layers and for reducing the possibility of conflicts with underlying database tables and functions.

MyQual-Base provides a set of database functions that implement the basic qualification and retrieval strategies described in Section 3.4. Such functions implement features common to each calculus and QSAM and call user-defined functions when some specific operations are needed. This approach allows for reaching generality through the definition of five user-generated functions:

- *relation-computer*: depends on a set of a-ary spatial calculi. Given an a-tuple of geometries, it executes opportune computational geometry operations to detect, for each calculus, which relation holds on the input tuple.
- *data-structure-table-maker*: depends on one QSAM. It takes care of preparing a set of database tables aimed at representing the data structures underlying the QSAM within the database.
- *data-structure-computer*: depends on one QSAM. It fills in, if necessary, the data structure tables.

[1] http://www.postgis.org/

- *qualifier*: depends on a set of calculi and one QSAM. Qualifies the spatial dataset according to the QSAM-specific reduction strategy.
- *retriever*: depends on a set of calculi and one QSAM. Given one spatial constraint retrieves from the relation tables all the entries that satisfy the constraint, opportunely exploiting the associated QSAM to rebuild non-stored relations.

The framework function implementing the basic matching algorithm (cf. Section 3.4.1) to solve a QSCQ has been designed to exploit the database execution plan optimization. It is implemented as a series of join operations, i.e. each constraint in a QSCQ is solved separately by calling the appropriate *retriever* function and the equality constraints on the query variables are used to join the results.

MyQual also allows for defining some *auxiliary* functions that can be used in the functions above. User-defined functions can be implemented in SQL or PL/PGSQL (the PostgreSQL procedural query language) as internal database functions. Alternatively they can be defined as external functions in one of the many programming languages supported by PostgreSQL. For example, PostgreSQL allows for defining external functions written in any C-like language. This makes the framework greatly flexible and scalable: allowing for possibly resorting to third-party external libraries.

### 5.1.1 MyQual-Base

MyQual has been designed to be a distributed framework, in the sense that a local installation can use the components installed in local or remote MyQual installations to operate on local or remote spatial databases.

Figure 5.2 shows the database schema of the Base component. It mainly consists of a small set of tables designed to keep trace of bound databases and to store access credentials. Moreover it allows for maintaining information about the actual installation and activation status of MyQual components on the listed databases.

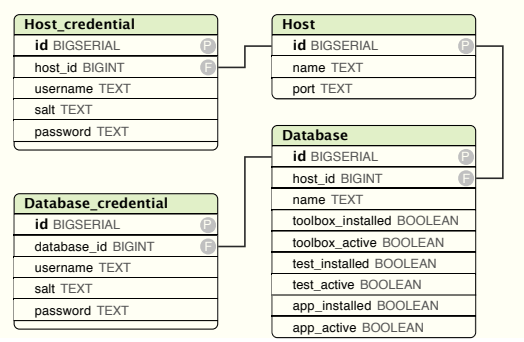



**Figure 5.2:** MyQual-Base, database schema.

MyQual provides a clean web-based interface that allows for easily administering the installation and the management of the different components as well as for putting new databases under control.

Figure 5.3 depicts a screenshot of the interface. The top part is a status-bar that continuously reports about the location of active components. Immediately below, follows the menu-bar which allows for readily moving among the four different parts of the application. Each part provides basic Graphical User Interfaces (GUIs) for the main framework functionalities that are outlined in the next sections.

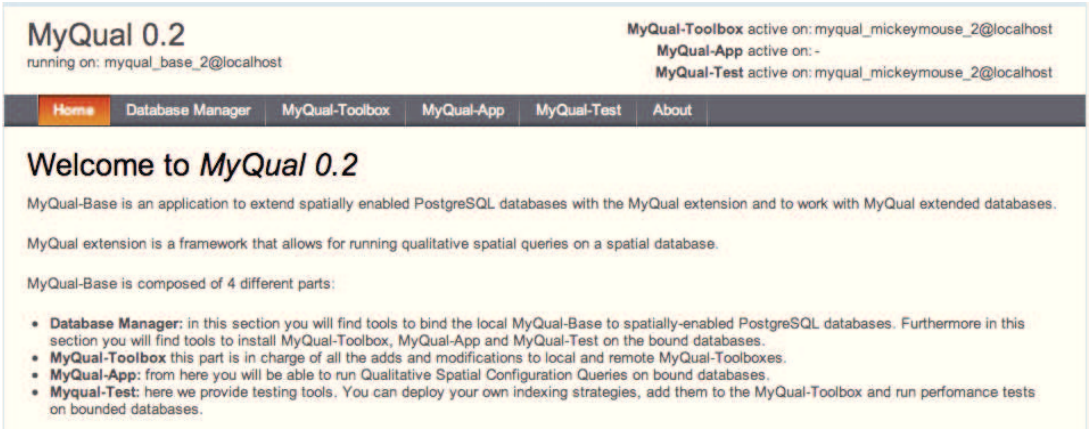



**Figure 5.3:** MyQual, interface screenshot.

### 5.1.2 MyQual-Toolbox

The Toolbox component is an extensible repository for spatial calculi and QSAMs. As such, it provides a practical realization to the assumption that the calculus pool (cf. Definition 3.1) is rich enough for encoding any spatial description (cf. Section 3.2). Its diagrammatic representation is reported in Figure 5.4 where the elements used to represent spatial calculi are reported in blue and those for QSAMs in green. Auxiliary elements are reported in red.

MyQual is capable of dealing with both *single-tile* as well as *multi-tile* calculi (cf. Section 2.2.4.3). An $\mathsf{a}$-ary spatial calculus is represented by (*i*) a set of qualitative relations, (*ii*) a set of reasoning tables and (*iii*) a *relation-computer* function. Basically, a qualitative relation consists of a symbol—i.e. the *name* of the relation—and of a reference to the calculus it belongs to. A reasoning table embodies either a permutation or a binary composition operation. It has associated a *reasoning-type* that specifies the type of operation the table represents. As explained in Section 2.2.3, for an $\mathsf{a}$-ary calculus can be defined up to $\mathsf{a}! - 1$ types of permutation operations and up to $(\mathsf{a}!)^3$ types of binary compositions. MyQual allows for possibly defining all of them.

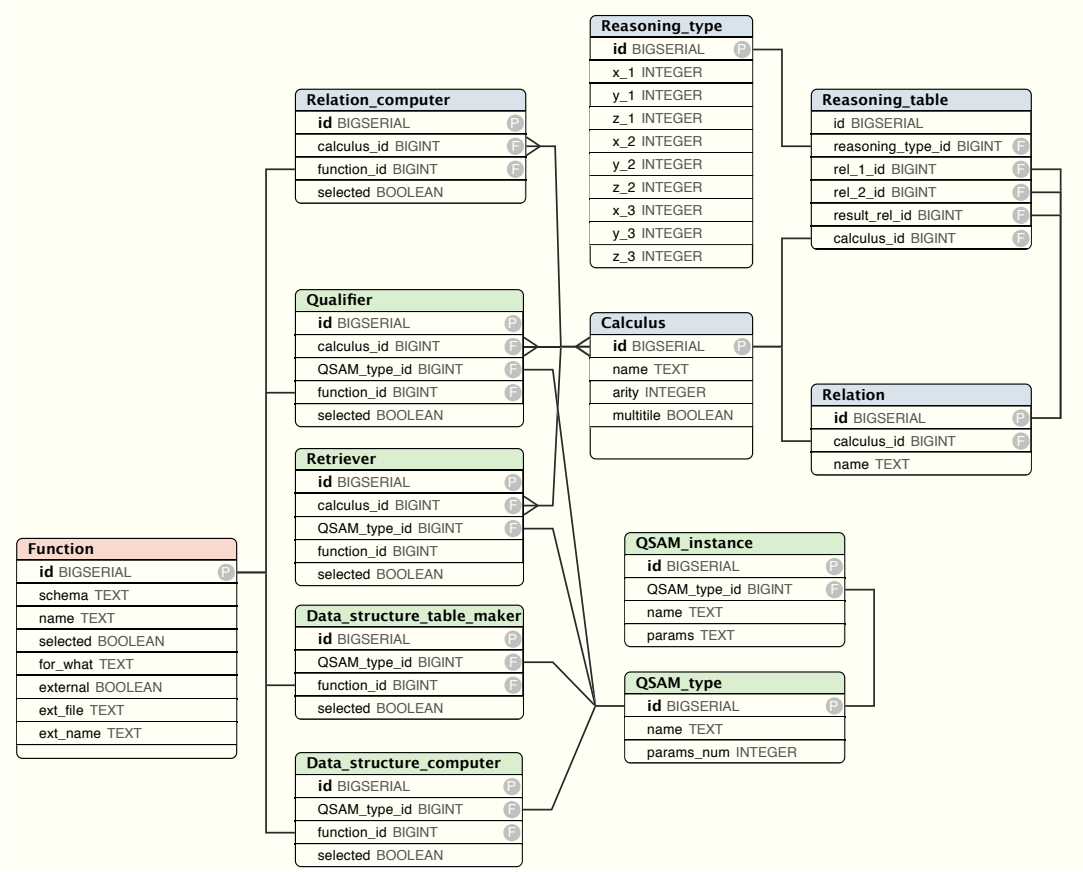



**Figure 5.4:** MyQual-Toolbox, database schema.

A QSAM has been defined (cf. Definition 4.1) as a triple $(q, \mathcal{D}, r)$ where $q$ is a *qualifier* function, $\mathcal{D}$ is a set of data structures and $r$ is a *retriever* function. In MyQual, the *qualifier* and *retriever* functions are represented as first-class objects, whereas the set of data structures is specified by means of two functions: the *data-structure-table-maker* function and the *data-structure-computer* function.

The framework differentiates between QSAM-*type* and QSAM-*instance*. A QSAM-*type* has a variable number of tuning parameters e.g. the Qualitative Spatial Representation and Reasoning (QSR) QSAM type has the parameter *length* that is the length of the underlying inference graph. A QSAM-*instance* is identified by a given assignment of the tuning parameters.

**Extending the Toolbox** The Toolbox can be extended in mainly two ways: defining new calculi and defining new QSAMs. The graphical interface provides suitable means for easily defining all the necessary elements explained above, possibly at different times. The extension goes through two main and uncorrelated points: the definition of a new entity (calculus or QSAM) and the definition of the corresponding functions.

MyQual allows for defining multiple functions for each calculus (resp. QSAM), but only one can be *selected* at any given time. This allows for possibly preparing a variety of implementations of the same function and for choosing the most appropriate one.

### 5.1.3 MyQual-Test: the Test Environment

MyQual-Test provides functionalities to execute performance tests on spatial calculi and QSAMs defined in the Toolbox component. Figure 5.5 depicts most relevant information about its database schema which can be conceptually split into two parts: pink colored entities are about testing whereas the yellow ones are concerned with backup/restore operations.

The main purpose of MyQual-Test is that of providing a benchmark environment to find best QSAM parameter values according to some spatial dataset properties, e.g. average object size. Such an environment allows for running, backing-up, restoring and re-running performance tests for dataset qualifications and QSCQs executions.

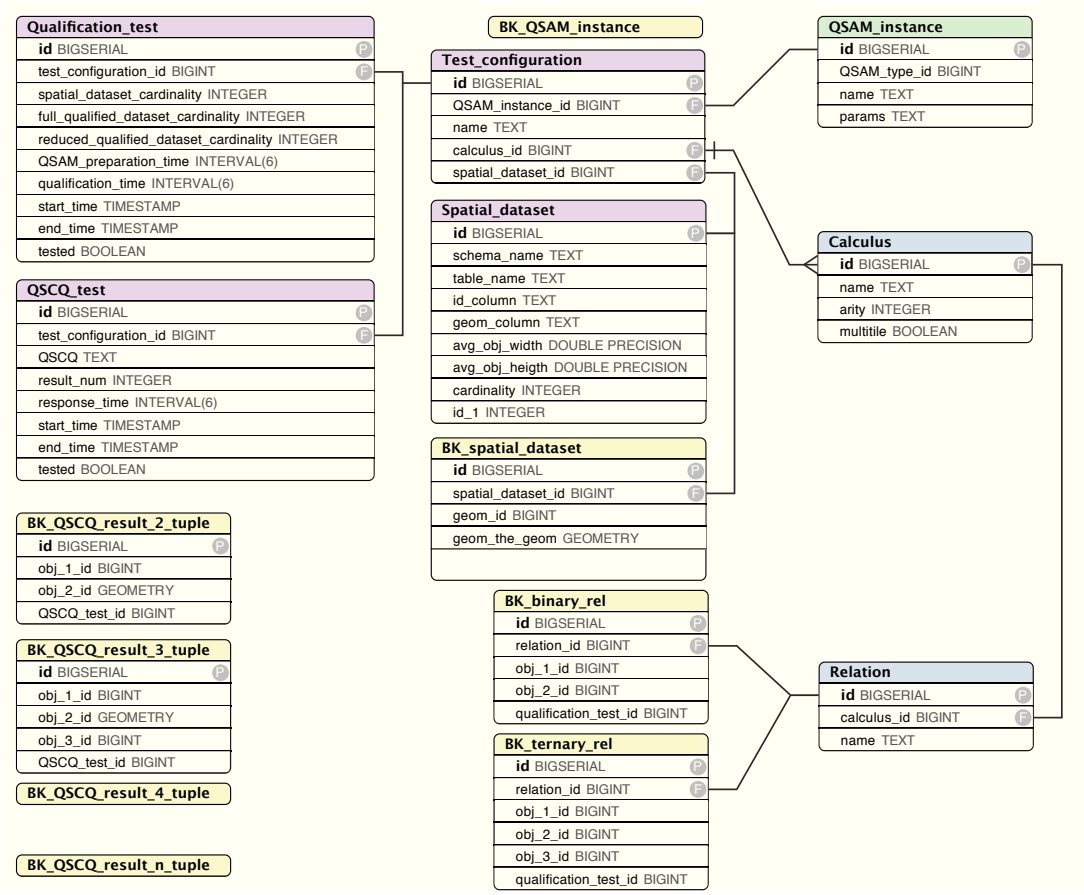



**Figure 5.5:** MyQual-Test, database schema.

**Preparing a test** At the core of this component lies the *test-configuration* which allows for preparing 3-tuples of the form

(QSAM-*instance*, *spatial calculus*, *spatial dataset*)

to be tested. Spatial datasets are intended to be database tables containing geometric objects. They can be uniquely located within the database by pairs of the kind

(*schema name*, *table name*)

which are collected in the *spatial-dataset* table. An entry in the *spatial-dataset* table can be enriched by some dataset-related parameters that might turn useful for the evaluation.

**Running a dataset qualification test** Once a *test-configuration* has been prepared, it is possible to run a dataset qualification test. The process is fully automated and the main workflow is the following: (*i*) Create empty relation tables to store the qualified dataset according to the calculi in the configuration. (*ii*) Prepare all data structures needed by the QSAM in the configuration by sequentially calling the corresponding functions: *data-structure-table-maker* and *data-structure-computer*. (*iii*) Execute the dataset qualification by calling the *qualifier* function associated with the calculi and QSAM in the configuration. (*iv*) Backup the geometric and the qualified dataset. (*v*) Store the test results in the *qualification-test* table. (*vi*) Delete the tables used for the test.

**Running a QSCQ execution test** A QSCQ execution test requires, besides a *test-configuration*, the specification of the QSCQ to be tested. The framework has an integrated QSCQ builder module that allows for easily defining a QSCQ by means of the basic GUI depicted in Figure 5.6. The query builder allows for adding spatial variables (objects to be searched) at will and for specifying spatial constraints over them, taking from the relations provided by the spatial calculi in the Toolbox component.

Before executing a QSCQ-*test* the application checks if the associated *test-configuration* has been already used in a previous test. If this is the case the qualified dataset is restored from the backup tables; otherwise a *qualification-test* is executed first and the qualified dataset is not deleted. The main execution workflow is the following: (*i*) If the QSAM specified in the *test-configuration* requires a dataset qualification and a *qualification-test* was already executed, restore the qualified dataset and QSAM data structures. Otherwise, create a new *qualification-test* and execute it. (*ii*) Encode the QSCQ into a SQL statement. (*iii*) Execute the SQL statement. (*iv*) Backup the query results. (*v*) Store the test results in the QSCQ-*test* table. (*vi*) Delete the tables used for the test.

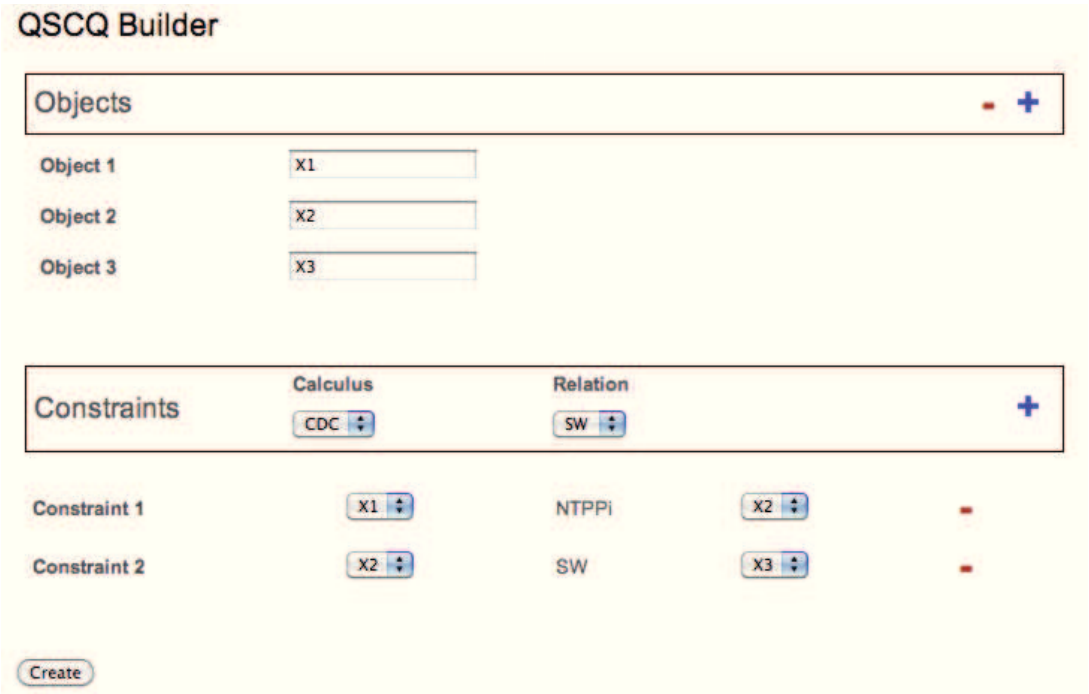



**Figure 5.6:** Qualitative Spatial Configuration Query builder interface.

Note that, since the relation symbols are available in the toolbox component, the SQL encoding can be done automatically (cf. Section 3.2.2). This operation is delayed until the test has to be executed to allow the user to prepare a QSCQ test even if the functions necessary for its execution have not been defined yet. Before performing any test, the application checks that all the necessary functions are available and, if this is not the case, skip the test.

### 5.1.4 MyQual-App: the Production Environment

MyQual-App has been designed to provide the *production environment*: a publicly accessible web interface for QSCQ-based spatial searches. The most relevant part of its database schema is depicted in Figure 5.7. It basically traces the part of the test environment dedicated to the configuration specification.

Once a QSAM has been sufficiently tested and opportunely tuned in the test environment, the best performing *test-configuration* can be moved to the production environment in order to make it available to the users of the actual service. That is, when a user accesses MyQual-App and specifies a QSCQ, the system uses the QSAM indicated in the *app-configuration* table to resolve the query.

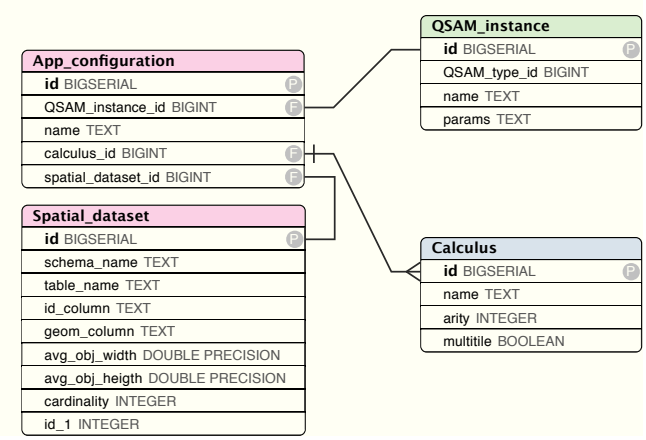



**Figure 5.7:** MyQual-App, database schema.

## 5.2 Implemented Spatial Clustering QSAMs

In this section two possible instances of the SC-QSAM family are presented. They are exemplary implementations that embody the most relevant peculiarities of this QSAM family. The first one, discussed in Section 5.2.1, resorts to a space tessellation based on a *grid* as a spatial clustering index. It aims at showing a case when the clustering does not provide a partition of the spatial dataset (cf. Section 4.3.3). The second implementation (Section 5.2.2) employs an R*-tree (cf. Section 2.3.4) as spatial clustering index and illustrate the case of a hierarchical indexing that always provides a partition of the spatial dataset.

Two binary spatial calculi are considered: the Region Connection Calculus (RCC) to model topological relations and the Cardinal Direction Calculus (CDC) for directional ones (cf. Section 2.2.4.4).

**Spatial Clustering Relations** RCC provides a set $\mathcal{B}(\text{RCC}) = \{\text{DC}, \text{EC}, \text{PO}, \text{EQ}, \text{TPP}, \text{TPPI}, \text{NTPP}, \text{NTPPI}\}$ of eight base relations (cf. Section 2.2.2). According to Definition 4.2 only one is a clustering relation: $\underline{\mathcal{B}}(\text{RCC}) = \{\text{DC}\}$.

CDC is a multi-tile calculus (cf. Section 2.2.4.2) and defines a set $\mathcal{B}(\text{CDC}) = \{\text{NW}, \text{N}, \text{NE}, \text{E}, \text{B}, \text{W}, \text{SW}, \text{S}, \text{SE}\}$ of nine single-tile base relations and 218 multi-tile base relations (when dealing with simple regions). Of these, four are clustering relations: $\underline{\mathcal{B}}(\text{CDC}) = \{\text{NW}, \text{NE}, \text{SW}, \text{SE}\}$.

**Relation Computer Functions** According to the "Simple Features Specification for SQL" (OpenGIS Consortium, 1998), PostGIS includes the definition of topological operators (cf. Section 2.3.2) which are used in the implementation of the RCC *relation-computer* function (cf. Section 5.1.2).

Since PostGIS does not provide directional operators, the CDC *relation-computer* function resorts to computational geometry.

### 5.2.1 *Grid*-based Spatial Clustering QSAM

*Grid*-based Spatial Clustering QSAM (*grid* SC-QSAM) is an instance of the SC-QSAM family. *Grid* SC-QSAM adopts a tiling strategy based on space tessellation by means of a *grid* aligned with the Cartesian axes and composed of uniform square cells with edge length $c$. The tileset $T$ (cf. Defintion 4.3) consists of all the *grid* cells that overlap with at least one dataset object. The corresponding clustering $\mathcal{C}$ is done according to Definition 4.4, i.e. an object belongs to each cluster associated with the cells that it spans. Note that, a *grid*-induced clustering is not guaranteed to provide a partition of the spatial dataset $O$ (cf. Example 5.1).

**Example 5.1** (*Grid*-based clustering) **-** Let us consider the dataset depicted in Figure 5.8(a). The clustering induced by the *grid* in Figure 5.8(b) is a partition of the dataset. This is not the case for the *grid* in Figure 5.8(c) since $o_1$ and $o_3$ fall in more than one cell—i.e. they are included in more than one cluster.

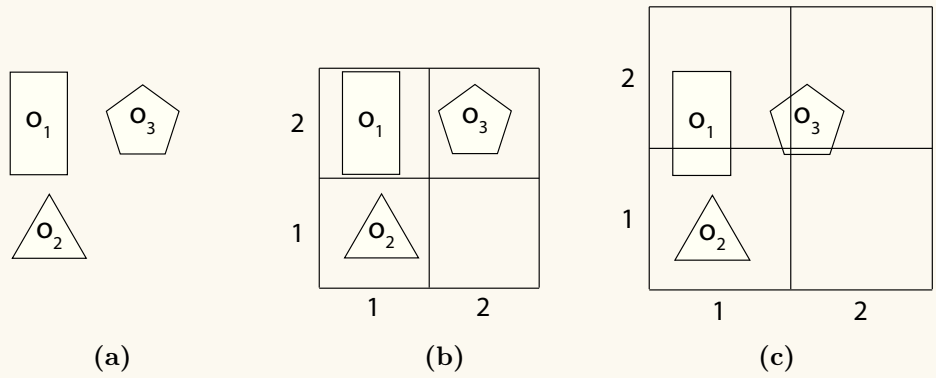



**Figure 5.8:** A spatial dataset (a) clustered by means of two *grid*s. One (b) generates a partition of the dataset but not the other (c).

For *Grid* SC-QSAM, both parameters $e$ and $v$ which, according to Equation 4.10, affect the reduction $\rho$ depend on the *grid* cell dimension. Thus, $c$ is the only parameter of interest when applying *grid*-based spatial clustering index.

**Data Structure** Given a *grid*, each of its cells is uniquely identified by a pair of integers $(X, Y)$ indicating the row and the column it is located at. We shall refer to such a pair as *grid-coordinates*. In Figure 5.8(b), for example, the cell containing $o_1$ is identified by the pair $(1, 2)$.

To produce the clustering $\mathcal{C}$ the bounding box of the objects is considered rather than their actual geometry. On the one hand, this approach slightly worsens the performed reduction $\rho$ since it augments the cluster overlapping—i.e. the parameter $e$. On the other hand, it allows for a quick computation of the clustering. Given the cell size $c$ and the Minimum Bounding Rectangle (MBR)

$\text{MBR}(o) = ((x_{min}, y_{min}), (x_{max}, y_{max}))$ of a dataset object $o$, the set of cells spanned by $\text{MBR}(o)$ can be computed as follows:

$$X_{min} = \left\lfloor \frac{x_{min}}{c} \right\rfloor ; Y_{min} = \left\lfloor \frac{y_{min}}{c} \right\rfloor ; X_{max} = \left\lfloor \frac{x_{max}}{c} \right\rfloor ; Y_{max} = \left\lfloor \frac{y_{max}}{c} \right\rfloor ;$$

where $\lfloor \cdot \rfloor$ is the floor operator and, $(X_{min}, Y_{min})$ and $(X_{max}, Y_{max})$ indicate, respectively, the bottom-left and the top-right cell of the spanned ones.

**Example 5.2** (Minimum bounding box approximation affects cluster overlap.) **-** Let us consider the situation depicted in Figure 5.9 where the minimum bounding box $\text{MBR}(o)$ of $o$ is reported in dashed lines. Note that $o$ only spans three cells, whereas its bounding box spans four.

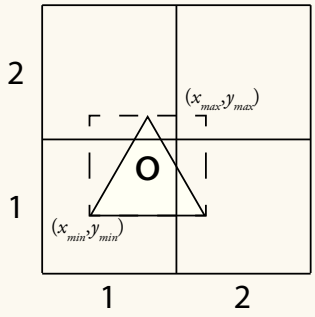



**Figure 5.9:** Grid-clustering with minimum bounding boxes.

The *grid* spatial clutering index is encoded in a database table like the one depicted in Figure 5.10: The field pair ($X$, $Y$) identifies the cluster and the field *obj-id* refers the id field of the geometric table being indexed. The name of such an id field is maintained in the *Spatial-dataset* table of the test (resp. production) environment, whose schema is reported in Figure 5.5 (resp. 5.7).

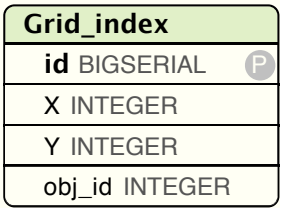



**Figure 5.10:** Database representation of a *grid* spatial clustering index.

**Qualification** The regularity of the *grid* structure benefits the qualification phase for both the considered calculi: The tile-relation holding among two cells can be directly implied by their *grid-coordinates*.

**RCC** A *grid*-reduced, RCC-qualified dataset is obtained by computing only the topological relations among pairs of objects in the same cluster.

Relations among objects taken from different clusters are completely disregarded, that is like assuming that all the tile pairs are in the only clustering relation DC. In fact, this is not true for pairs of adjacent cells since they share a boundary line but the assumption is justified by the manner in which we build the clusters: in the limit case when an object $o$ touches the boundary line among two cells it is included in the clusters associated to both such cells. Accordingly, $o$ will be considered in the relation computation of the two object clusters. This ensures that all the non-clustering relations holding among $o$ and any other object are stored in the relation tables.

**CDC** A *grid*-reduced, CDC-qualified dataset is obtained as follows. For each cluster $C$ associated to a tile (*grid* cell) $t = (X, Y)$, we compute

- the relations among pairs of objects in $C$
- the relations among pairs of the kind $(o_i, o_j)$ where $o_i$ is in $C$ and $o_j$ is in one of the clusters associated to the cells having one *grid*-coordinate equal to $(X, Y)$

This approach perfectly fits the theoretical framework traced in Section 4.3.3: The relations excluded from the computation are those between object pairs $(o_i, o_j)$ where $o_i$ is in $C$ and $o_j$ is in one of the clusters associated to the cells that are *not* in a clustering relation with $t$.

**Retrieval** The retriever functions for RCC and CDC draw upon the same main logic described in Section 4.3.4. However, given that RCC provides only one clustering relation, for the reconstruction of missing relations it is possible to resort to a technique known as *negation as failure* (Clark, 1978) which allows for a more efficient implementation (see below).

**RCC** Let $\xi = (\mathrm{R}, \chi)$ be the constraint to be satisfied, then the RCC retriever function is implemented as follows:

- If R is not a clustering relation ($\mathrm{R} \notin \underline{\mathcal{B}}$), the retrieval consists in a look-up from the *relation table* $\mathrm{RT}_2$.
- Otherwise, the retrieval strategy comprehends a relation reconstruction stage since, in the qualification phase, only a subset of clustering relations holding between object pairs have been computed and stored in $\mathrm{RT}_2$. Missing relations are rebuilt via a set-difference operation: subtracting from the set of all object pairs the set of those among which relation R does not hold (*negation as failure*).

**CDC** Let $\xi = (\mathrm{R}, \chi)$ be the constraint to be satisfied. The CDC retriever function differs from the one for RCC in the relation reconstruction phase: If R is a clustering relation ($\mathrm{R} \in \underline{\mathcal{B}}$), for each tile pair $(t_1, t_2)$ in $\mathrm{RT}_2$ such that $\mathrm{R}(t_1, t_2)$ return all the object pairs $(o_i, o_j)$ with $o_i \in C_1$ and $o_j \in C_2$. Where $C_1$ and $C_2$ are the object clusters associated with the tiles $t_1$ and $t_2$, respectively.

### 5.2.2 R*-tree-based Spatial Clustering QSAM

R*-tree-based Spatial Clustering QSAM (R*-tree SC-QSAM) employs an R*-tree (cf. Section 2.3.4) as a spatial clustering index. The R*-tree fits with Definition 4.8 and is therefore a hierarchical spatial clustering index: The lower layer of the three provides a spatial clustering index for the dataset and any other layer provides a spatial clustering index for the tiles in the layer below.

The shape of the R*-tree data structure depends on a small set of parameters that also affect the reduction $\rho$; they are:

- $m$: the minimum number of entries in a tree node
- $M$: the maximum number of entries in a tree node
- $rp$: the percentage of nodes to reinsert in case of node overflow
- $cs$: the maximum number of nodes considered in the *choose-sub-tree* function called in the insertion phase

Beyond such parameters we shall also consider a further one that indicates the level of the tree which the operations of qualification and retrieval are applied recursively downwards from. We call such a parameter *sl* (starting level).

**Data Structure** The R*-tree spatial clustering index is encoded in a database table like the one depicted in Figure 5.11. Each entry represents a tree node. The *parent-id* field reports the id of the parent node, the *b-box* field maintains information about the geometric extent of the node and the *lvl* field indicates the level of the tree the node is located at. The leaves of the tree are bounding boxes of dataset objects and are located at level zero. Entries corresponding to leaves also maintain, in the field *obj-id*, a reference to the id of the contained object.

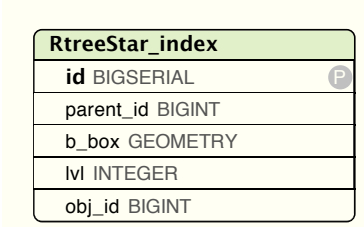



**Figure 5.11:** Database representation of an R*-tree spatial clustering index.

**Qualification** Differently than a *grid*, the construction process of an R*-tree is data-driven and its final structure depends on, besides the index parameters, the spatial arrangement of the indexed dataset. Accordingly, it is not possible to know in advance what the relation among two tiles (directory rectangles of the tree nodes) is.

The qualifier functions for RCC and CDC realize a recursive variant of the general procedure described in Algorithm 4.4 (Section 4.3.3). Namely, the basic qualification is applied on the tiles at the starting level $sl$. The procedure computes the relation R holding on each tile pair $(t_i, t_j)$ in the current level. If $\mathrm{R} \in \underline{\mathcal{B}}$ is a clustering relation, all the children of the tree node corresponding to $t_i$ are to be in the same relation R with all the children of $t_j$. Then the qualifier function store the relation $\mathrm{R}(t_i, t_j)$ in the relation table $\mathrm{RT}_2$. Otherwise, if the level in consideration is not the leaf one and $\mathrm{R} \notin \underline{\mathcal{B}}$ is not a clustering relation, the procedure recurses on the lower level, only considering the tree nodes having $t_i$ or $t_j$ as parent node. Note that the relations holding on objects at the leaf level are computed using the actual geometries of the dataset objects instead of their bounding box.

**Retrieval** Similarly to the qualifier functions, the retriever functions for RCC and CDC realize a recursive variant of the general procedure described in Algorithm 4.5 (Section 4.3.4). Let $\xi = (\mathrm{R}, \chi)$ be the constraint to be satisfied, then the retriever function is implemented as follows:

- If R is not a clustering relation ($\mathrm{R} \notin \underline{\mathcal{B}}$), the retrieval consists in a look-up from the *relation table* $\mathrm{RT}_2$.
- Otherwise, for each level $lvl \leq sl$ and for each tile pair $(t_1, t_2)$ in $\mathrm{RT}_2$ such that $t_1$ and $t_2$ are at level $lvl$ of the tree and such that $\mathrm{R}(t_1, t_2)$, retrieve all the children leaf nodes of $t_1$ and $t_2$. Return all the object pairs $(o_i, o_j)$ with $o_i$ being a child of $t_1$ and $o_j$ being a child of $t_2$.

Note that, as for *grid*-based SC-QSAM, also in this case for the RCC retriever function the reconstruction phase can be empowered by applying *negation as failure* technique.

## 5.3 Summary

In this chapter a novel software framework called MyQual has been presented . Its main purpose is to provide a development, test and production environment for SAMs for qualitative spatial queries, with a particular focus on QSAMs and QSCQs.

MyQual has been designed as an extension for PostGIS-enabled PostgreSQL databases and is logically divided in four parts: MyQual-Base, MyQual-Toolbox,

MyQual-Test and MyQual-App. Each part, when installed in a database, produces a dedicated schema where all the tables, functions and indexes used by the component are kept separated from the rest of the database.

MyQual-Base encodes all the basic functionalities of the framework which comprehend the management of bound databases. All the other components can be installed separately and can be used from different instances of MyQual-Base.

MyQual-Toolbox is an extensible repository for qualitative spatial calculi and QSAMs. It provides both a realization of the assumption that the calculus pool (cf. Definition 3.1) is rich enough for encoding any natural spatial description (cf. Section 3.2) and a development environment for QSAMs.

MyQual-Test is the test environment where different configurations of QSAMs and spatial calculi can be tested upon different spatial datasets. It provides two main test typologies: qualification tests and QSCQ execution tests. The environment provides GUIs to readily prepare and run such tests. The results are automatically backed-up to allows for comparison and reproduction. QSCQs can be generated via a web-form GUI.

MyQual-App has been thought as the production environment where most promising QSAMs and spatial calculi can be made available to casual users willing to execute qualitative spatial queries.

MyQual achieves generality by being grounded in the definition of five user-defined functions. Such functions depends on spatial calculi and QSAMs and can be implemented in a variety of programming languages.

The second part of the chapter has been devoted to present the implementation of two instances of SC-QSAM within MyQual. The first instance presented, resorts to a simple *grid* as spatial clustering index and serves as an example of the case in which the clustering does not provide a dataset partition. The second implementation grounds in the R*-tree data structure and provides an exemplary case of hierarchical spatial clustering index.

Chapter 6

# Empirical Evaluation

This chapter provides an empirical evaluation of the Qualitative Spatial Access Methods (QSAMs) presented in Chapter 4. In Section 6.1 the main evaluation criteria are outlined. Section 6.2 is devoted to testbed definition. Section 6.3 discusses experiments on Functional QSAM and Qualitative Storage Layer QSAM. Performance of *grid*-based and R*-tree-based Spatial Clustering QSAM are analyzed in Sections 6.4 and 6.5, respectively; whereas QSR-based QSAM is discussed in Section 6.6. Section 6.7 is devoted to a comparison of the most promising QSAMs on a real geospatial dataset. The chapter ends with a comparison of the results in Section 6.8.

## 6.1 Evaluation Criteria and Experiments Setup

In order to assess the QSAMs presented in Chapter 4, we are interested in evaluating the following features as the cardinality of the addressed spatial dataset changes: (*i*) dataset qualification time, (*ii*) space occupancy of the qualified dataset, and (*iii*) Qualitative Spatial Configuration Query (QSCQ) execution time.

Qualified dataset occupancy directly depends on the number of pre-computed relations, which is independent of the characteristics of the hosting machine. Accordingly, we measure point (*ii*) in terms of the cardinality of the produced qualified datasets.

QSCQ retrieval time scales with the output size, i.e. the number of returned configurations. This, in turn, depends on the number and on the type of relations encoded in the QSCQ. Accordingly, we also discuss the correlation between the retrieval time of a given QSCQ and the size of its output.

MyQual (cf. Section 5.1) has been used to implement and test all the QSAMs presented in this work (cf Chapter 4 and Section 5.2) with two binary qualitative

spatial calculi:the Region Connection Calculus (RCC) and the Cardinal Direction Calculus (CDC).

Experiments have been run in two main stages: spatial dataset qualification and QSCQ execution. The qualification stage is executed first. MyQual automatically backs up the qualified dataset and the data structures underlying the tested QSAM, if any, in order to allow for reproducing the experiment. Once backed up, these items are deleted from the database. For each QSCQ execution the qualified dataset and the QSAM data structures are restored from the backup tables, then the QSCQ is resolved and timed. Also in this case MyQual automatically backs up the results of the query and deletes all the restored items. Note that, if two QSCQs have to be tested against the same qualified dataset, the latter is restored and deleted twice (even if the executions are consequential). This is done to reduce to the minimum the interference of the optimization instruments of the underlying Database Management System (DBMS) in the timing: A freshly restored table is treated as a new object by the DBMS, i.e. no statistics are available for the query optimizer that can be used to boost the execution of a query.

Finally, note that the relation tables, where qualified datasets are stored, have been purposely designed and implemented without any indexing technique in order to reduce to the minimum the interference of DBMS optimization components on the outcomes. However, as claimed in Section 3.5.2 one of the advantages of pre-qualification is that the resulting dataset can be indexed by means of standard indexing techniques such as B-trees that allow to further shorten QSCQ response times.

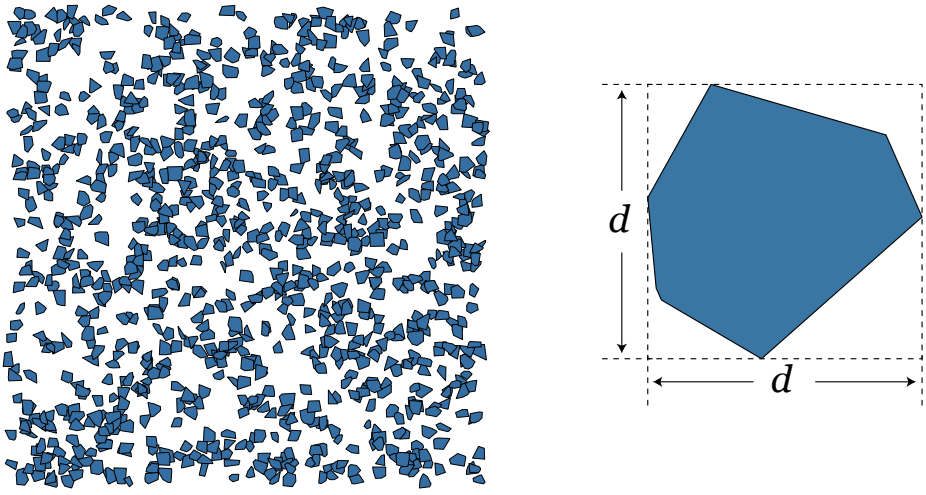
The reported computing times refer to an Apple iMac mounting a 2.7 GHz Intel Core i5 (4 cores) processor and a 12 GB 1333 MHz DDR3 RAM unit.

## 6.2 Evaluation Testbed

Most of the QSAMs have been tested over a common testbed in order to allow for a direct comparison of the results. When a different testbed has been used, because of some QSAM-specific features, it is specified in the corresponding section.

We shall discriminate the testbed according to the two experiment types: The *qualification testbed* consists of a set of randomly generated spatial datasets. Each dataset consists of convex polygons[1] uniformly distributed in a $D \times D = 500 \times 500$ fixed-size workspace. The dataset cardinality $\mathsf{N}$ ranges over the set $\{100, 250, 500, 750, 1000\}$. Figure 6.1(a) depicts a random dataset with $\mathsf{N} = 1000$. The average dataset object extent on each dimension (x- and y- projection) $d$ varies over the set $\{7.5, 15\}$—refer to Figure 6.1(b) for a pictorial explanation. The *qualification testbed* consists of 10 dataset instances for every pair $(\mathsf{N}, d) \in \{\{100, 250, 500, 750, 1000\} \times \{7.5, 15\}\}$, for a total of 100 randomly generated

[1]The polygons have a variable number of vertices ranging between 3 and 8. Note, however, that the number of vertices is unimportant in the scope of this evaluation since it does not affect the outcomes of the experiments.




(a) (b)

**Figure 6.1:** A dataset from the *qualification testbed* (a) consisting of 1000 uniformly distributed polygons with average extent $d$ (b) equal to 15.

spatial datasets. Note that the dataset statistical parameter $d$ is useful only for *grid* Spatial Clustering QSAM (SC-QSAM), therefore, for all the other QSAMs it is not used as a dataset discriminant.

The *execution testbed* is reported in Table 6.1 and consists of 3 QSCQs for each tested calculus. Such QSCQs have been accurately chosen on the base of some preliminary experiments as samples of the different QSCQ response performance. All other QSCQs behave similarly to one of those in the testbed, for example the preliminary experiments showed that there is no relevant difference in performance between single-tile and multi-tile CDC constraints. $\mathcal{Q}_{\text{RCC},1}$ and $\mathcal{Q}_{\text{CDC},1}$ are instances of single-constraint QSCQs with a high selectivity on qualified datasets—i.e. they produce a small number of results—whereas $\mathcal{Q}_{\text{RCC},2}$ and $\mathcal{Q}_{\text{CDC},2}$ are single-constraint QSCQs with low selectivity. $\mathcal{Q}_{\text{RCC},3}$ and $\mathcal{Q}_{\text{CDC},3}$ represent instances of multi-constraint QSCQs.

| Calculus | Name | Spatial Variables | Constraints | ID in Graphics |
|---|---|---|---|---|
| RCC | $\mathcal{Q}_{\text{RCC},1}$ | $x_1, x_2$ | $\text{PO}(x_1, x_2)$ | PO |
| | $\mathcal{Q}_{\text{RCC},2}$ | $x_1, x_2$ | $\text{DC}(x_1, x_2)$ | DC |
| | $\mathcal{Q}_{\text{RCC},3}$ | $x_1, x_2, x_3$ | $\text{PO}(x_1, x_2), \text{DC}(x_2, x_3)$ | PO + DC |
| CDC | $\mathcal{Q}_{\text{CDC},1}$ | $x_1, x_2$ | $\text{N}(x_1, x_2)$ | N |
| | $\mathcal{Q}_{\text{CDC},2}$ | $x_1, x_2$ | $\text{NW}(x_1, x_2)$ | NW |
| | $\mathcal{Q}_{\text{CDC},3}$ | $x_1, x_2, x_3$ | $\text{N}(x_1, x_2), \text{NW}(x_2, x_3)$ | N + NW |

**Table 6.1:** The *execution testbed* consists of 2 single-constraint and 1 multi-constraint QSCQs for each tested calculus.

The outcomes of the experiments have been used to draw conclusions about what are the more suitable values for QSAMs' parameters, i.e. the values providing

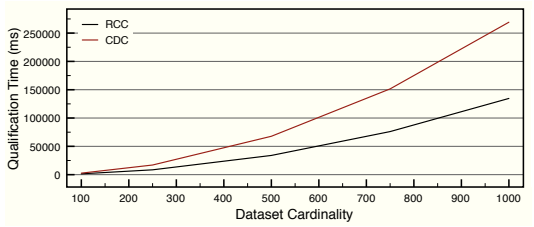
best space-time performance. Such information has been used to tune up the most promising QSAMs in a final comparison experiment run on a real geospatial dataset: the OpenStreetMap dataset of the city of Bremen (Germany).

## 6.3 Functional and Qualitative Storage Layer QSAMs

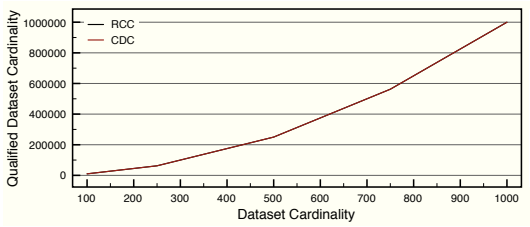
Functional QSAM (F-QSAM) and Qualitative Storage Layer QSAM (QSL-QSAM) have been introduced in Sections 4.1 and 4.2, respectively. They embody the basic dataset qualification strategies presented in Section 3.5 and do not have any tuning parameter. These QSAMs have been tested on the testbed described in Section 6.2.

**Qualification** F-QSAM does not require any pre-qualification, thus it has null qualification time and null storage space occupation.

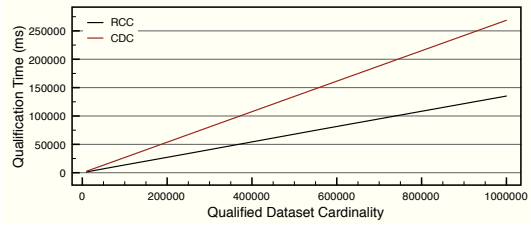
Its direct counterpart is QSL-QSAM which pre-qualifies the whole dataset into the opportune relation tables. Figure 6.2 shows the results of the dataset qualification procedure for both RCC and CDC.




**(a)** Qualification time as the spatial dataset cardinality varies.




**(b)** Qualified dataset cardinality as the spatial dataset cardinality varies.




**(c)** Qualification time as the qualified dataset cardinality varies.

**Figure 6.2:** Spatial dataset qualification by means of the Qualitative Storage Layer QSAM.

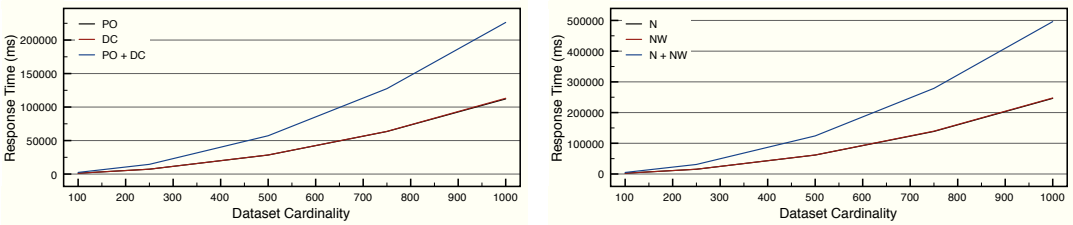
As depicted in Figure 6.2(b), since these are both 2-ary calculi, the cardinalities of the corresponding qualified datasets $\mathcal{R}_{\text{RCC}}(O)$ and $\mathcal{R}_{\text{CDC}}(O)$ coincide. According to Equation 3.2, they are equal to

$$|\mathcal{R}_{\text{RCC}}(O)| = |\mathcal{R}_{\text{CDC}}(O)| = \mathsf{N}^2$$

where N is the cardinality of the spatial dataset $O$.

The graphics in Figure 6.2(a) reports the average qualification time while varying the spatial dataset cardinality N. Figure 6.2(c) shows the linear dependency between qualification time and qualified dataset cardinality. The deviation between the two curves indicates that the CDC relation computer function (cf. Section 5.1) used in the qualification procedure runs in longer time with respect to the RCC relation computer function. In particular, according to the shown results, CDC relation computation takes approximately twice the time than RCC. This behavior is due to the way the *relation computer functions* (cf. Section 5.1) for the two calculi have been implemented and to the number of relations that have to be checked: 8 for RCC vs 128 for CDC. As a consequence, we expect this bias to occur in all qualification time results.

**QSCQ** Average QSCQ response times for F-QSAM are reported in Figures 6.3(a) and 6.3(b): RCC and CDC, respectively.



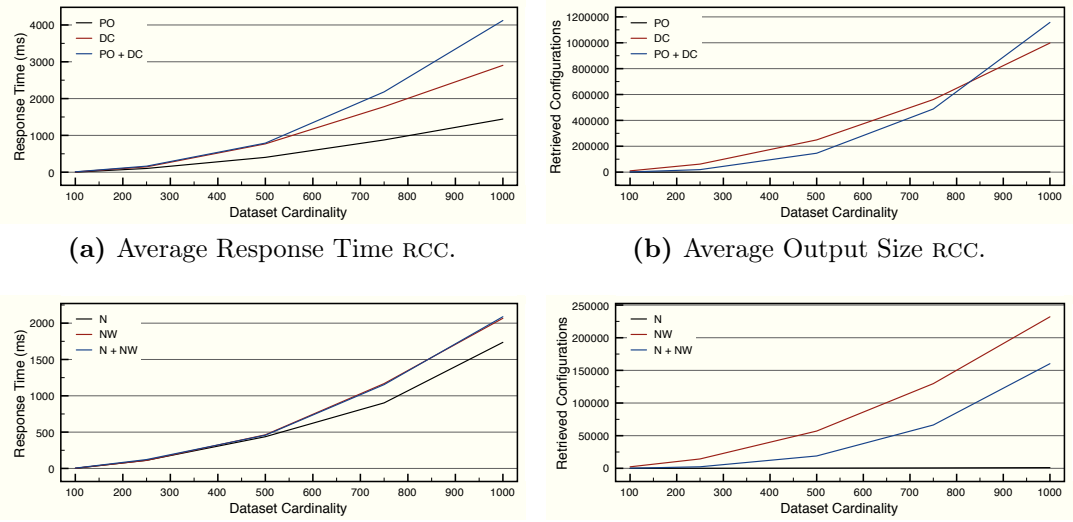

**(a)** Average Response Time RCC. **(b)** Average Response Time CDC.

**Figure 6.3:** QSCQ execution outcomes when applying Functional QSAM.

The graphics show that response time grows quadratically with the spatial dataset cardinality N independently of the solved QSCQ and, thus, of the number of retrieved configurations. This is due to the fact that, at any QSCQ execution, F-QSAM performs a full dataset qualification. Consistently, response times for single-constraint QSCQs coincide (approximately) with qualification times of QSL-QSAM reported in Figure 6.2: qualification times are slightly higher since they comprehend disk writing operations. Note that multi-constraint QSCQs ($\mathcal{Q}_{\text{RCC},3}$ : $\{\text{PO}(x_1, x_2), \text{DC}(x_2, x_3)\}$ and $\mathcal{Q}_{\text{CDC},3}$ : $\{\text{N}(x_1, x_2), \text{NW}(x_2, x_3)\}$), run in time double than single-constraint queries. This is due to the manner the general solving procedure has been implemented in MyQual: the two constraints are resolved independently and the variable equalities are used to join the two result sets.

Figure 6.4 summarizes QSCQ execution outcomes for RCC (upper graphics) and CDC (lower graphics) when resorting to QSL-QSAM.

The graphics on the left report average execution times while varying the spatial dataset cardinality N. On the right, average QSCQ output sizes are reported. Note that graphics for RCC and CDC report on different scales. For both RCC and CDC the graphics on the left show that response times for single-constraint QSCQs




**(a)** Average Response Time RCC. **(b)** Average Output Size RCC.

**(c)** Average Response Time CDC. **(d)** Average Output Size CDC.

**Figure 6.4:** QSCQ execution results when applying Qualitative Storage Layer QSAM. Response times for RCC (a) and CDC (c) grow proportionally with the number of retrieved configurations shown in (b) and in (d), respectively.

are quadratic with N but grow proportionally with the corresponding number of retrieved results reported on the right.

Beyond the number of returned results, response times obtained for multi-constraint QSCQs are affected by two more factors related to the join operation intervening when solving a QSCQ and to the query optimizer of the underlying DBMS: (*i*) The join operation increases the response time by a factor that grows proportionally with the output size of every single constraint in the query. (*ii*) Given that all the relations are stored in the same database table, answering a multi-constraint query requires accessing such a table twice. Since the accesses are in read-mode, the table does not undergo any change and this allows the underlying DBMS to optimize the second access, e.g. resorting to caching.

For the multi-constraint RCC QSCQ—i.e. $\mathcal{Q}_{\text{RCC},3} : \{\text{PO}(x_1, x_2), \text{DC}(x_2, x_3)\}$—the high number of results and, thus, the time increase adduced by the join operation dominate the reduction provided by the optimizer. This is not the case for $\mathcal{Q}_{\text{CDC},3} : \{\text{N}(x_1, x_2), \text{NW}(x_2, x_3)\}$. In this case the boost introduced by the optimizer allows for response times very close to those obtained for the single-constraint QSCQ with low selectivity—i.e. $\mathcal{Q}_{\text{CDC},2} : \text{NW}(x_1, x_2)$.

**Discussion** F-QSAM and QSL-QSAM provide some goodness thresholds for our evaluation. F-QSAM corresponds to run-time qualification, hence it provides an upper bound for QSCQ response time: a QSAM answering QSCQs in longer time than F-QSAM constitutes a bad solution for QSCQ execution. QSL-QSAM fully pre-qualifies a spatial dataset, thus it provides an upper bound for the extra storage space: a QSAM producing larger qualified datasets than QSL-QSAM is a bad solution for dataset qualification.

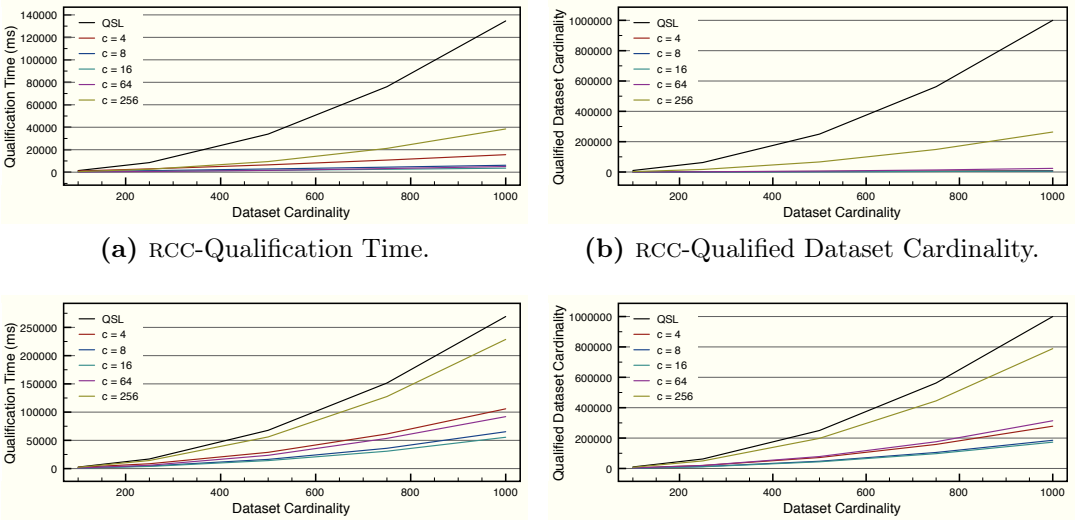
## 6.4 Grid-based Spatial Clustering QSAM

*Grid*-based Spatial Clustering QSAM (*grid* SC-QSAM) is a particular instance of the Spatial Clustering QSAM (SC-QSAM) family described in Section 4.3. As discussed in Section 5.2.1, *grid* SC-QSAM clusters the spatial dataset by superimposing a *grid* aligned with the Cartesian axes and composed of uniform square cells. The cell edge length $c$ is the only tuning parameter.

*Grid* SC-QSAM has been tested over the *qualification* and *execution testbed* described in Section 6.2. The cell size $c$ has been ranged over the value set $\{4, 8, 16, 64, 256\}$ giving rise to a total of 500 dataset qualification experiments and 4000 QSCQ execution tests.

The goal of the experimentation[1] is to trace the behavior of the QSAM as the dataset statistical parameters and the grid cell $c$ change to possibly detect optimal values for $c$. That is, the size of the grid cell that produces qualified datasets of minimum cardinality and shortest QSCQ response times. As a reference value to judge the goodness of the boost introduced by *grid* SC-QSAM the results are compared with Qualitative Storage Layer QSAM (QSL-QSAM).

**Qualification** The outcomes of the dataset qualification experiments are reported in Figure 6.5. The results show that *grid* SC-QSAM always outperforms QSL-QSAM in both qualified dataset cardinality and qualification time and for both the tested calculi.




**(a)** RCC-Qualification Time. **(b)** RCC-Qualified Dataset Cardinality.

**(c)** CDC-Qualification Time. **(d)** CDC-Qualified Dataset Cardinality.

**Figure 6.5:** Spatial dataset qualification by means of *grid*-based Spatial Clustering QSAM. Qualification times for RCC (a) and CDC (c) scale proportionally with the cardinality of the qualified dataset shown in (b) and in (d), respectively.

[1]Note that part of the results presented in this section have been published in (Fogliaroni *et al.*, 2011).

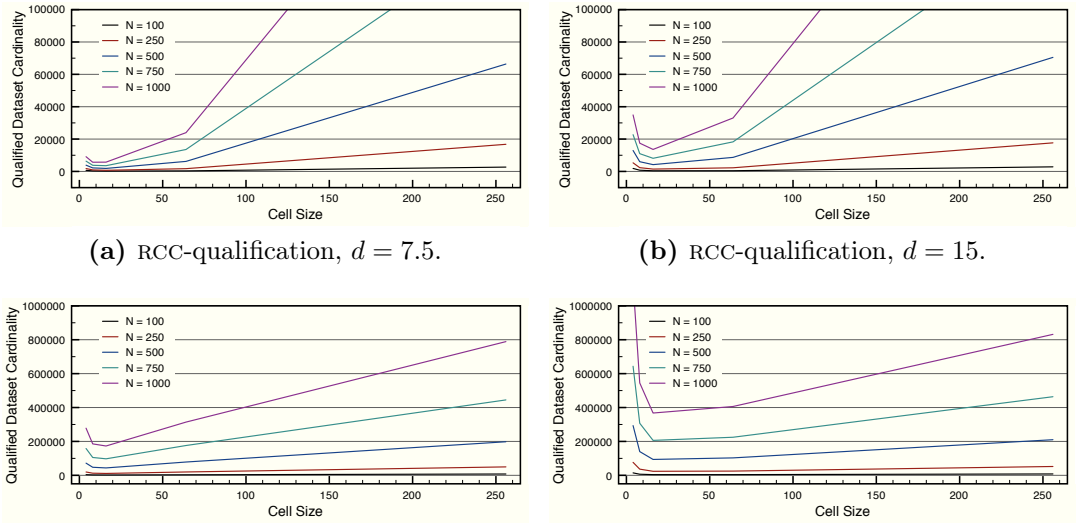
The graphics in Figure 6.5(b) depicts the behavior of a set of (colored) curves representing the RCC-qualified dataset cardinality for different values of the cell size $c$ as the dataset cardinality $\mathsf{N}$ varies. Although the cardinality of the *grid*-reduced RCC-qualified dataset $|\mathcal{R}_{\text{RCC},grid}(O)|$ grows with $\mathsf{N}$ in a manner similar to $|\mathcal{R}_{\text{RCC}}(O)|$, the *grid* approach performs several orders of magnitude better as $c$ decreases. Figure 6.5(d) shows a similar behavior for CDC.

Figures 6.5(a) and 6.5(c) report qualification times for RCC and CDC, respectively. As in the case of QSL-QSAM, qualification time for *grid* SC-QSAM grows quadratically with the spatial dataset cardinality $\mathsf{N}$ but is linear with the cardinality of the qualified dataset.

Such results let infer that the smaller the cell size $c$, the better *grid* SC-QSAM performs. However, for uniformly distributed spatial datasets, if $c$ is smaller than the average object size $d$, an object spans, on average, multiple cells and thus is assigned to multiple clusters. According to the analysis of the reduction performed by SC-QSAM (cf. Section 4.3.3), this yields an increase of the parameter $e$ (indicating the amount of overlap among object clusters) and, consequently, a performance degradation. Similarly, if cells are greater than or equal to the whole workspace $D$, one cell contains all the dataset objects and *grid* SC-QSAM degenerates into QSL-QSAM.

The expectation is that by reducing $c$, the cardinality of the qualified dataset shrinks to a minimal value when $c$ reaches the optimal value $\hat{c}$, whereas reducing $c$ beyond this limit causes $|\mathcal{R}_{\text{RCC},grid}(O)|$ to increase again.

The graphics in Figure 6.6 show a different view of the previous results that allows for easily verifying the above argumentation. Each curve represents the average qualification outcomes for a given spatial dataset cardinality $\mathsf{N}$ while varying the cell dimension $c$.




**(a)** RCC-qualification, $d = 7.5$. **(b)** RCC-qualification, $d = 15$.

**(c)** CDC-qualification, $d = 7.5$. **(d)** CDC-qualification, $d = 15$.

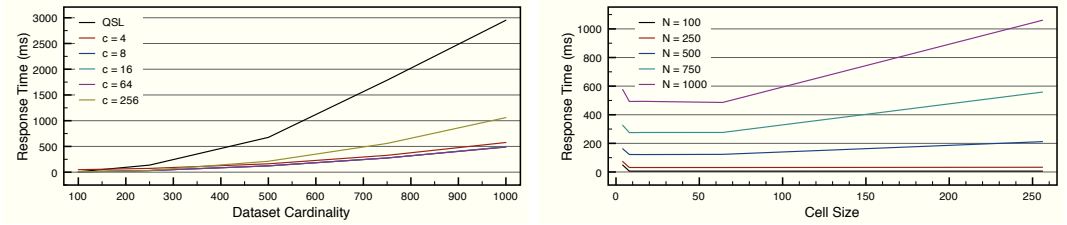
**Figure 6.6:** Qualified dataset cardinality while varying *grid* cell dimension $c$.

Figures 6.6(a) and 6.6(b) report RCC-qualified dataset cardinality for spatial datasets with average object size $d = 7.5$ and $d = 15$, respectively. Note that for any given $\mathsf{N}$ the number of computed relations decreases with $c$ until a minimum value below which it suddenly rises. The value of $c$ for which the curves reach the minimum represents the searched optimal cell size $\hat{c}$. It is independent of the dataset cardinality and approximately equal to the average dataset object size $d$.

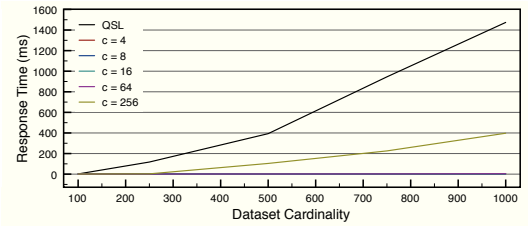
The main difference between the two graphics lies in the growing rates of the qualified dataset cardinality when $c < \hat{c}$. This is due to the fact that the bigger the average object size, the more the cell an object spans and the more the augmentation of the parameter $e$.

Figures 6.6(a) and 6.6(b) show that CDC-qualification behaves similarly.

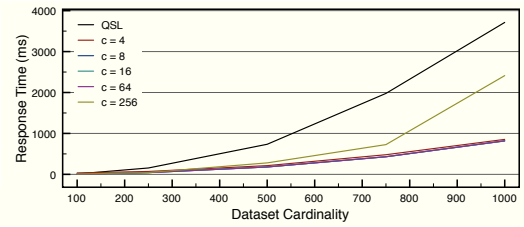
**QSCQ** Figure 6.7 depicts average response times for RCC QSCQ experiments when resorting to *grid*-based Spatial Clustering QSAM. In this case we only report graphics showing the results of the experiments run on spatial datasets with average object dimension $d = 7.5$. Experiments on datasets with $d = 15$ yields similar results.




**(a)** $\mathcal{Q}_{\text{RCC},2} : \{\text{DC}(x_1, x_2)\}$.




**(b)** $\mathcal{Q}_{\text{RCC},1} : \{\text{PO}(x_1, x_2)\}$.




**(c)** $\mathcal{Q}_{\text{RCC},3} : \{\text{PO}(x_1, x_2), \text{DC}(x_2, x_3)\}$.

**Figure 6.7:** Response times for QSCQs encoding RCC constraints when applying *grid*-based Spatial Clustering QSAM.

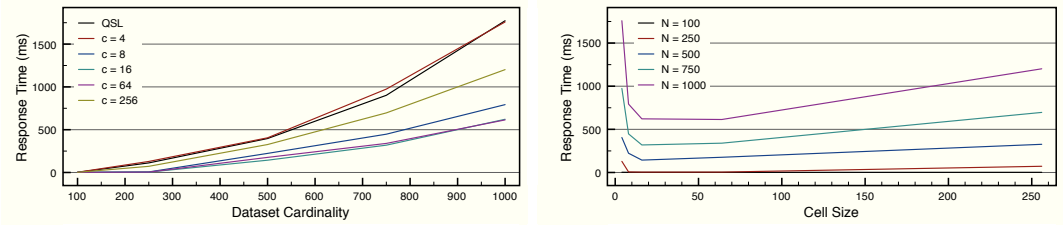
Figure 6.7(b) shows the results obtained for the $\mathcal{Q}_{\text{RCC},2} : \{\text{DC}(x_1, x_2)\}$. The curves depicted in the graphics on the left represent the average response time for different cell size $c$ (colored curves) while varying the spatial dataset cardinality $\mathsf{N}$. As usual, also the outcomes obtained for QSL-QSAM are reported (black curve) to provide a goodness reference value. Similarly to the dataset qualification experiments, the graphics demonstrates that *grid* SC-QSAM outperforms QSL-QSAM: the smaller the cell size, the better the performance. The graphics on the right traces the average response times while varying the cell size $c$. Each

curve represents the behavior for spatial datasets of a given cardinality. Similarly to the qualification experiments QSCQ execution performance improves as the cell size decreases until an optimal value $\hat{c}$ is reached. Below this value the QSAM performance suddenly deteriorates. The empirical results show that also in this case the best performance is obtained for grids with cell size nearly equal to the average object size: $\hat{c} \approx d$.

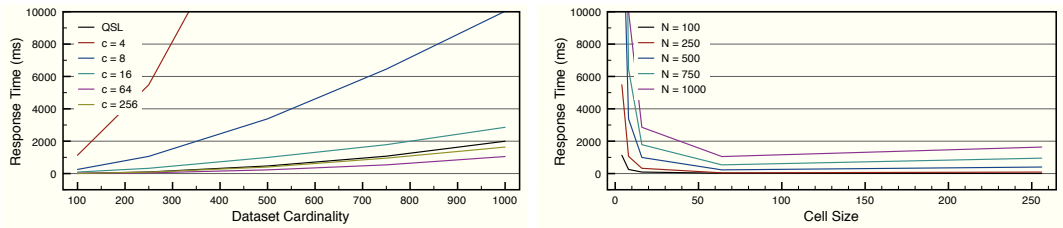
Figures 6.7(a) and 6.7(c) show graphics representing the average response times for QSCQs $\mathfrak{Q}_{\text{RCC},1} : \{\text{PO}(x_1, x_2)\}$ and $\mathfrak{Q}_{\text{RCC},3} = \{\text{PO}(x_1, x_2) + \text{DC}(x_2, x_3)\}$. The graphics demonstrate a behavior similar to the case of $\mathfrak{Q}_{\text{RCC},1}$ we just discussed. Also in this case the empirical results indicate the best performance to occur when resorting to a grid with cell size approximately equal to the average object dimension.

The graphics in Figure 6.8 summarize response times for QSCQs with CDC constraints tested on spatial datasets with average object size $d = 15$. Experiments on datasets with $d = 7.5$ yield similar outcomes.

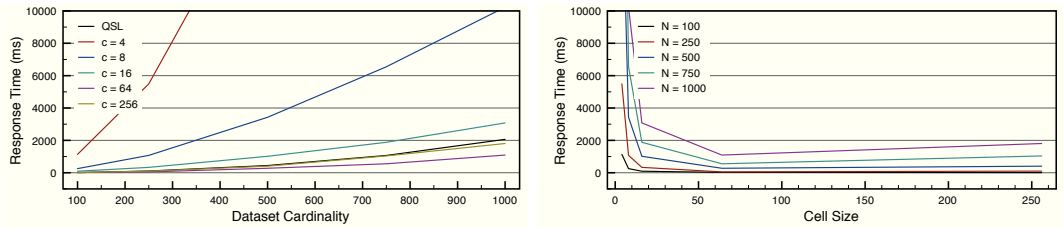
Figure 6.8(a) depicts graphics summarizing the experimentation on the QSCQ $\mathfrak{Q}_{\text{CDC},1} : \{\text{N}(x_1, x_2)\}$. The graphics on the left shows a behavior similar to that




**(a)** $\mathfrak{Q}_{\text{CDC},1} : \{\text{N}(x_1, x_2)\}$.




**(b)** $\mathfrak{Q}_{\text{CDC},2} : \{\text{NW}(x_1, x_2)\}$.




**(c)** $\mathfrak{Q}_{\text{CDC},3} : \{\text{N}(x_1, x_2), \text{NW}(x_2, x_3)\}$.

**Figure 6.8:** Response times for QSCQs encoding CDC constraints when applying *grid*-based Spatial Clustering QSAM.

obtained for $\mathcal{Q}_{\text{RCC},1}$ and reported in Figure 6.7(b). The main difference lies in the fact that for $\mathcal{Q}_{\text{CDC},1}$ and cell size $c = 4$ *grid* SC-QSAM performs worse than QSL-QSAM. The graphics on the right show that the best performance are obtained for grids with cell size approximately equal to the average object dimension $d$.

The graphics in Figure 6.8(b) summarize the outcomes obtained for the QSCQ $\mathcal{Q}_{\text{CDC},2} : \{\text{NW}(x_1, x_2)\}$. Note that when employing grids with cell size smaller than the average object size—i.e. $c = 4$ and $c = 8$—*grid* SC-QSAM performs significantly worse than QSL-QSAM. Moreover the grid with $c = 16$ produces response times very similar to QSL-QSAM. The graphics on the right, provides a better view of such results: In this case best performance is reached for $c = 64$. This behavior is due to the way the retriever function (cf. Section 5.2.1) implements the reconstruction of clustering relations: The function iterates over all cells associated with an object cluster. For each of such cells the retriever looks up from the relation table all the cells located NW of the given one in order to rebuild object pairs arranged according to the searched relation. Accordingly, in this case there is a new factor affecting the response time: the number of cells produced by the *grid* partition. The smaller the cell size, the higher the number of cells and, consequently, the higher the response time.

Finally, Figure 6.8(c) summarizes response times for the multi-constraint QSCQ $\mathcal{Q}_{\text{CDC},3} : \{\text{N}(x_1, x_2), \text{NW}(x_2, x_3)\}$. The curves representing the results show a trend very similar to the case of $\mathcal{Q}_{\text{CDC},2}$ letting intend that the retrieval times for clustering relations (NW in this case) dominate those for non-clustering ones (N).

**Summary** Dataset qualification experiments demonstrate that, for both the tested calculi, *grid* SC-QSAM performs as much better as the grid cell size reduces and reaches the optimum when it is nearly equal to the average object size: $d$.

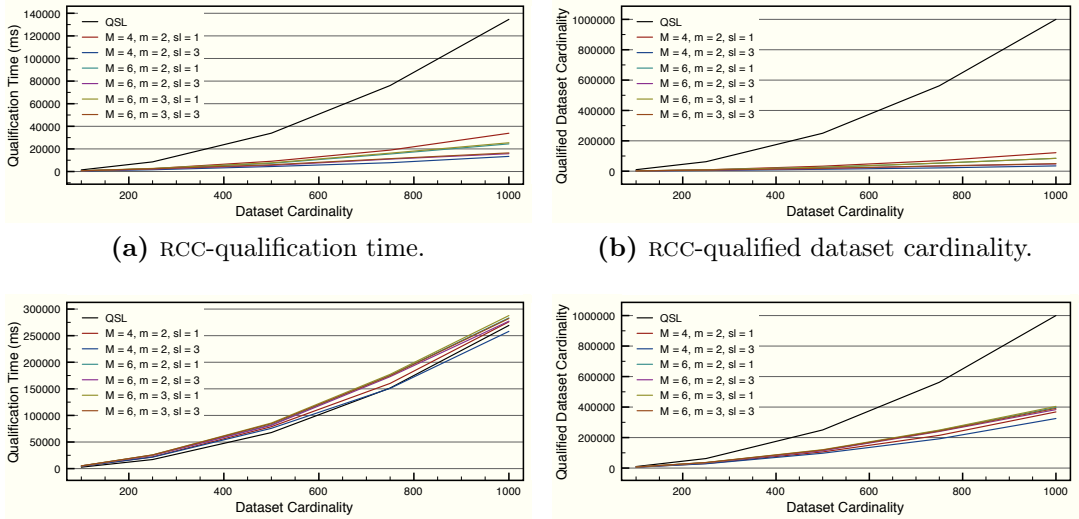
QSCQ execution experiments show that for RCC best performance is obtained when $c$ is approximately equal to $d$, whereas for CDC, and for the tested parameter values, SC-QSAM performs better with $c = 64$.

## 6.5 R*-tree-based Spatial Clustering QSAM

R*-tree-based Spatial Clustering QSAM (R*-tree SC-QSAM) has been introduced in Section 5.2.2. It resorts to an R*-tree (Beckmann *et al.*, 1990) as a hierarchical spatial clustering index (cf. Definition 4.8) and has been tested with the *qualification* and *execution testbed* described in Section 6.2.

R*-tree is an instance of the R-tree (Guttman, 1984) family (cf. Section 2.3.4) resulting from an engineering analysis of the original R-tree. The result of such an analysis yielded the definition of four index parameters which are used in different phases of the data structure management in order to optimize its performance. The parameters are the maximum ($M$) and minimum ($m$) number of entries a tree node is allowed to contain, the percentage ($rp$) of nodes that are reinserted when a node overflow occurs, and the number ($cs$) of nodes considered at each level of the tree to choose the most suitable subtree to accommodate an object being inserted.

Beckmann *et al.* (1990) conducted an extensive empirical evaluation of R*-tree which led to the identification of parameter values producing best performance. In our experiments we ranged $M$, $m$, $rp$, and $cs$ consistently with such findings: The parameter $M$ influences the number of tree levels. We calibrated it according to the cardinality of the spatial datasets under consideration in order to obtain trees of different height: We opted for ranging $M$ on the values $\{4, 6\}$, obtaining trees of height comprised between 3 and 5.




**(a)** RCC-qualification time. **(b)** RCC-qualified dataset cardinality.

**(c)** CDC-qualification time. **(d)** CDC-qualified dataset cardinality.

**Figure 6.9:** Spatial dataset qualification by means of R*-tree-based Spatial Clustering QSAM. Qualification times for RCC (a) and CDC (c) grow proportionally with the cardinality of the qualified dataset shown in (b) and in (d), respectively.

Beckmann *et al.* showed that R*-tree produces best retrieval performance when the minimum number of node entries $m$ and the percentage of reinserted nodes $rp$ are set, respectively, to 40% and to 30% of $M$. Accordingly we ranged $m$ on the value set $\{2, 3\}$ and kept $rp$ fixed to 30% of $M$.

Finally, Beckmann *et al.* found that, when indexing objects in 2-D space, the best value for the parameter $cs$ is 32. This setting is incongruous with the values we chose for the other parameters. Given that our trees have a branching factor at most equal to 6, setting $cs = 32$ would correspond to checking all the subtrees rooted in any node traversed when inserting an object in order to find the most suitable one. Then, we decided to keep $cs$ fixed and equal to $m$ in all our tests. This value still ensures a good optimization in the R*-tree insertion phase.

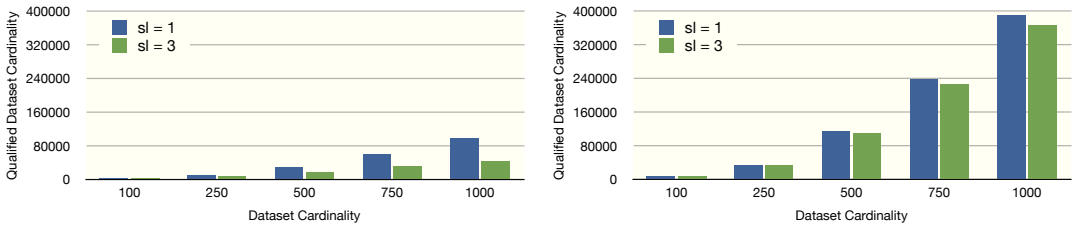
Beyond R*-tree parameters, R*-tree SC-QSAM has a further parameter: $sl$ indicates the starting level of the tree from which the qualification and retrieval operations are carried out. We ranged $sl$ on the values $\{1, 3\}$ as, according to the previous settings and to the cardinality of the spatial datasets in the testbed, an R*-tree is always guaranteed to have at least 3 levels.

The settings above produced 6 different QSAM setups which, in turn, gave rise to a total of 600 dataset qualification experiments and 3600 QSCQ execution tests.

**Qualification** Given its constructive properties, R*-tree provides always a partition of the spatial dataset. Accordingly, the reduction performed in the dataset qualification phase (cf. Section 4.3.3) is not affected by the parameter $e$ (since it is always equal to 0). As a consequence the reduction only depends on the parameter $v$ (which indicates the number of tile 2-tuples in a clustering relation).

The outcomes of the dataset qualification experiments are reported in Figure 6.9. The graphics in the upper subfigures refer to RCC and show that R*-tree SC-QSAM (colored curves) always outperforms QSL-QSAM (black curve) in both qualification time—Figure 6.9(a)—and qualified dataset cardinality—Figure 6.9(b).

The set of curves in Figure 6.9(d) represent average cardinality of CDC-qualified datasets for different QSAM parameter setups, whereas Figure 6.9(c) reports about average qualification time. The graphics show that R*-tree SC-QSAM executes in time slightly longer than QSL-QSAM but produces a qualified dataset of cardinality remarkably smaller than that of the full CDC-qualified dataset.



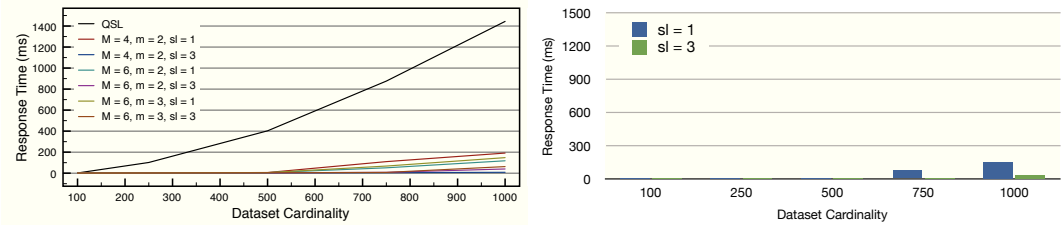

**(a)** RCC-qualified dataset cardinality. **(b)** CDC-qualified dataset cardinality.

**Figure 6.10:** Qualified dataset cardinality for different values of R*-tree parameter $sl$ as the spatial dataset cardinality varies.

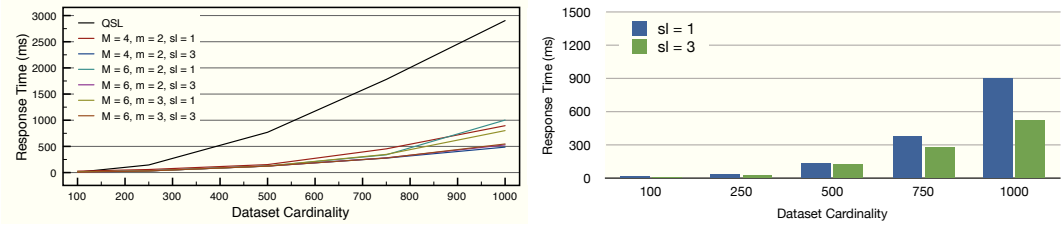
For both the tested calculi, the cardinality of qualified dataset obtained when applying R*-tree SC-QSAM is mainly influenced by parameter $sl$: Independently of the other parameters, tests with $sl = 3$ produced more compact qualified datasets. This is better highlighted in the graphics in Figure 6.10 which show the cardinality of the qualified datasets for the two values of $sl$ as the dataset cardinality varies.

This behavior is due to the qualifier function (cf. Section 5.2.2) that avoids to compute the qualitative relation holding among two dataset objects if the tiles they are contained into have been found to be in a clustering relation. Accordingly, the higher the level of the tree in which a tile 2-tuple is found to be in a clustering relation, the higher the number of object-relation the algorithm does not compute.

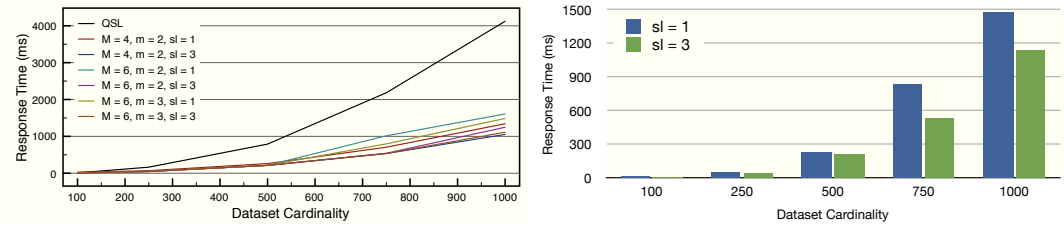
**QSCQ** The results for QSCQ execution are reported in Figures 6.11 and 6.12: respectively for RCC and CDC. Each subfigure refers to a given QSCQ from the *execution testbed* in Section 6.2. The graphics on the left show response time while varying the spatial dataset cardinality N with the main aim of providing performance comparison with respect to QSL-QSAM (black curve). Note that




**(a)** $\mathcal{Q}_{\text{RCC},1} : \{\text{PO}(x_1, x_2)\}$.




**(b)** $\mathcal{Q}_{\text{RCC},2} : \{\text{DC}(x_1, x_2)\}$.




**(c)** $\mathcal{Q}_{\text{RCC},3} : \{\text{PO}(x_1, x_2), \text{DC}(x_2, x_3)\}$.

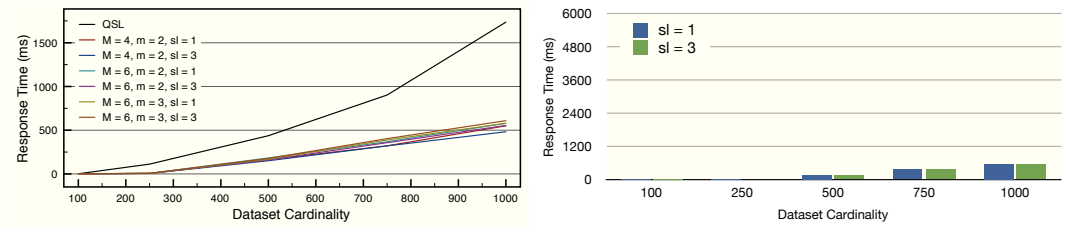
**Figure 6.11:** Response times for QSCQs encoding RCC constraints when applying R*-tree-based Spatial Clustering QSAM.

different graphics report on different scales. The graphics on the right show the dependency between the response time and the parameter *sl*. They report data on a uniform scale; thus they also provide an immediate means for comparing the results obtained for different QSCQs.

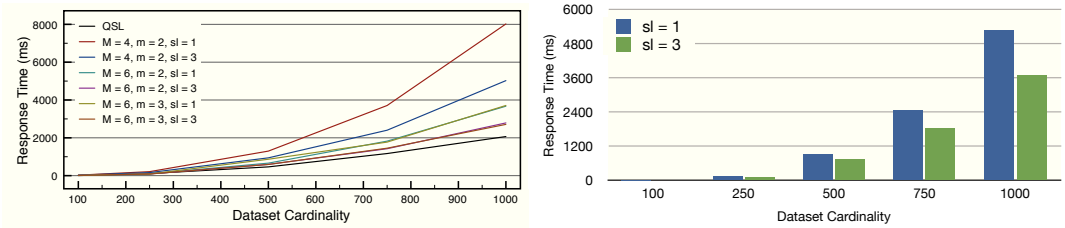
The results obtained for RCC (Figure 6.11) show that R*-tree SC-QSAM outperforms QSL-QSAM independently of the parameter setting and of the QSCQ. Moreover, performance mainly depends on the parameter *sl*: Settings with $sl = 3$ led, on average, better performance than the corresponding cases with $sl = 1$.

The graphics on the right allow for drawing more specific conclusions. $\mathcal{Q}_{\text{RCC},2} : \{\text{PO}(x_1, x_2)\}$ is resolved significantly faster than the other QSCQs. This is due to the fact that PO is not a clustering relation; therefore all its occurrences are stored and can be readily looked up from the relation table. Response time for

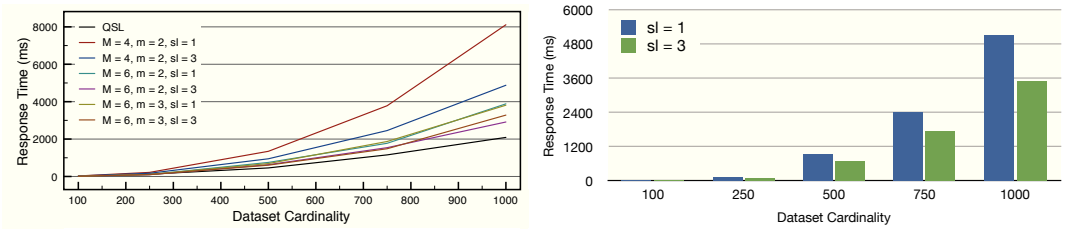
the multi-constraint QSCQ $\mathcal{Q}_{\text{RCC},3} : \{\text{PO}(x_1, x_2), \text{DC}(x_2, x_3)\}$ equals the sum of the response times of the single constraints plus the time needed to join the two result sets. The optimization components of the underlying DBMS slightly improve the retrieval of the second constraint and, thus, the overall performance.




**(a)** $\mathcal{Q}_{\text{CDC},1} : \{\text{N}(x_1, x_2)\}$.




**(b)** $\mathcal{Q}_{\text{CDC},2} : \{\text{NW}(x_1, x_2)\}$.




**(c)** $\mathcal{Q}_{\text{CDC},3} : \{\text{N}(x_1, x_2), \text{NW}(x_2, x_3)\}$.

**Figure 6.12:** Response times for QSCQs encoding CDC constraints when applying R*-tree-based Spatial Clustering QSAM.

The results reported in Figure 6.12 show a different behavior for QSCQs encoding CDC constraints. First-off, the graphics on the right show that setting $sl = 3$ improves retrieval performance independently of the QSCQ under examination, although the gain in performance is nearly null for QSCQs encoding non-clustering relations, i.e. $\mathcal{Q}_{\text{CDC},2} : \{\text{N}(x_1, x_2)\}$. Again, the graphics on the left of Figure 6.12(a) indicates that, when dealing with non-clustering relations, R*-tree SC-QSAM outperforms QSL-QSAM independently of the parameter setup.

Finally, more interesting results are those represented in the graphics on the left of Figures 6.12(b) and 6.12(c), i.e. QSCQs containing clustering relations. In these cases, the majority of the tested parameter setups produced response times significantly longer than QSL-QSAM. The motivation is related to the reconstruction process that has to be undertaken to rebuild the clustering relations.

Best performance is obtained for parameter setups with having $M = 6$ and $sl = 3$. This behavior is probably due to the parameter $M$: Allowing for a higher number of entries in each tree node yields an R*-tree with a smaller number of levels and tiles. Consequently the retriever function performs a reduced number of loops in the reconstruction phase.

**Summary** Dataset qualification experiments demonstrate that, for both the tested calculi, R*-tree SC-QSAM performs as much better as the parameter $sl$ is higher. In our tests best results have been obtained for $sl = 3$. The other parameters do not affect the performance significantly. Note, however, that R*-tree SC-QSAM performs CDC-qualification in time slightly longer than QSL-QSAM, obtaining more compact qualified datasets though.

QSCQ execution experiments showed that the best performance is obtained for $sl = 3$. For queries encoding RCC constraints R*-tree SC-QSAM always outperforms QSL-QSAM, independently of the other parameters.

For queries encoding CDC spatial clustering relations best performance is obtained with parameter setups with $M = 6$.

## 6.6 QSR-based QSAM

QSR-based QSAM (QSR-QSAM) has been introduced in Section 4.4. It is grounded in a novel data structure named *inference graph* (cf. Section 4.4.1) that aims at highlighting the inference patterns existing among the qualitative relations holding on a spatial dataset $O$. Each node of an inference graph $\mathcal{IG}$ represents a relation in the qualified dataset $\mathcal{R}_{\boldsymbol{\mathcal{P}}}(O)$ where $\boldsymbol{\mathcal{P}}$ is the pool of calculi under consideration. A hyperarc $a$ in $\mathcal{IG}$ is called *inference path*; it always originates in a set of nodes $\mathcal{T}$ (the tail of the path) and heads to one node $h$ (the head of the path). An inference path of length $l$ represents a series of $l$ reasoning operations that have to be undertaken to infer $h$ from $\mathcal{T}$. An inference path is an instantiation of an *inference template* which, in turn, is generated from the reasoning tables (cf. Section 2.2.3) of the calculus under consideration. An inference graph of length $l$ does not contain inference paths longer than $l$.

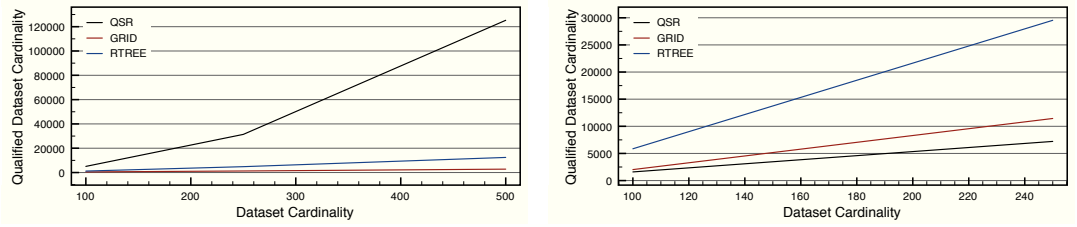
The length $l$ of the inference path is the only parameter of QSR-QSAM and is bounded above by the number of base relations in the calculus. Generating an inference graph of length $l$ for a calculus $\mathcal{M}$ requires to check every inference template of length smaller or equal to $l$ against the relations of the full qualified dataset $\mathcal{R}_{\mathcal{M}}(O)$.

To build the inference templates we implemented a template generator. For RCC the generation of inference templates of any length from the reasoning tables reported in Table 4.3 took approximately 5 seconds. The number of generated templates as the length $l$ varies is reported in Table 4.5.

For CDC, using the reasoning tables provided in (Skiadopoulos & Koubarakis, 2004; Wang & Hao, 2010), the template generator produced 1839 inference templates of length 1 in approximately 7 seconds. However, given the size of the reasoning tables, the generation of templates of length 2 took approximately 6 hours and yielded 32354339 inference templates. We did not go over $l = 2$.

Due to the computational cost of its algorithms, QSR-QSAM has been tested only on a subset of the *qualification testbed* described in Section 6.2. More specifically, RCC has been tested on spatial datasets of cardinality 100, 250, and 500, whereas CDC only on datasets of cardinality 100 and 250. Finally, still due to the computational costs and to the high number of inference templates, only experiments with $l = 1$ have been conducted for both calculi in order to retain comparable results.

**Qualification** The graphics in Figure 6.13 show the cardinality of the qualified datasets produced by QSR-QSAM in comparison with the best results obtained with *grid*-based and R*-tree-based Spatial Clustering QSAM. Qualification times for both calculi are several orders of magnitude bigger than the other solutions: the qualification of datasets of cardinality 250, for instance, took approximately 43 minutes for RCC and approximately 8.8 hours for CDC; whereas best performance of *grid* SC-QSAM was approximately 1 second and 4 seconds, respectively.




**(a)** Qualified Dataset Cardinality RCC. **(b)** Qualified Dataset Cardinality CDC.

**Figure 6.13:** Spatial dataset qualification by means of QSR-based QSAM. Cardinality of qualified datasets produced for RCC (a) and CDC (b) compared with the best results obtained with *grid*-based and R*-tree-based Spatial Clustering QSAM.

The experiments with RCC showed that QSR-QSAM produces qualified datasets of cardinality approximately equal to half of the full RCC-qualified $|\mathcal{R}_{\text{RCC}}(O)|$. This behavior can be explained by analyzing the statistical distribution of the relations in the qualified dataset and the RCC reasoning tables—cf. Table 4.3(a): On average, approximately 99.5% of RCC relations holding on the random datasets used for the experiments are DC. Since, the composition of DC with any other RCC relation yields a disjunctive relation, no inference path of length 1 having DC in its tail is generated from the composition table—cf. Table 4.3(b). Moreover, since DC permutes in DC, the inference graph of length 1 mostly consists of cycles of two nodes (and two simple arcs). Each such cycle is a relaxed strongly connected

component (cf. Definition 4.11) having only one source node that is the one stored by the qualification procedure.

The graphics in Figure 6.13(a) shows that both *grid* R*-tree SC-QSAM and R*-tree SC-QSAM outperform QSR-QSAM by producing RCC-qualified datasets of cardinalities much smaller than those generated by QSR-QSAM.

The experiments run on CDC show a completely different behavior: the graphics in Figure 6.13(b) indicates that QSR-QSAM produces very compact CDC-qualified datasets, outperforming both *grid* R*-tree SC-QSAM and R*-tree SC-QSAM. Again this is due to the statistical distribution of the relations in the qualified dataset and to the CDC reasoning tables (cf. Skiadopoulos & Koubarakis, 2004; Wang & Hao, 2010): The datasets used in the experiments gave rise to a well-variegated set of CDC relations and the constitution of the reasoning tables allow for producing an inference graph with a complex hyperarc structure. This, in turn, allowed for identifying strongly connected components consisting of several nodes, eventually yielding a higher reduction of the qualified datasets.

**QSCQ** The retriever function for QSR-QSAM has been implemented as follows: First it applies algebraic closure algorithm (cf. Section 2.2.5) to rebuild the full qualified dataset which is stored in the relation tables and queried via the retriever function of QSL-QSAM. Note that in case of multi-constraint QSCQs, given the computational cost of algebraic closure algorithm, the reconstruction is executed only when resolving the first constraint.

The results of the experiments showed that in the retrieval phase QSR-QSAM performs largely worse than all the other discussed solutions, F-QSAM included. On spatial datasets of cardinality $\mathsf{N} = 250$, for instance, all the QSCQs in the testbed encoding RCC constraints have been resolved in approximately 1 minute, against the nearly 7 seconds taken by F-QSAM. CDC performed even worse: all of the CDC QSCQs in the testbed have been answered in approximately 1 hour.

The huge time difference occurring between RCC and CDC is mainly related to the different number of base relations provided by the two models (8 vs 218) and thus to the size of the reasoning tables that are used in the algebraic closure to perform the reasoning.

These results yield the main conclusion that a plain application of QSR-QSAM on a full spatial dataset is definitely unsuitable in terms of query response times.

**Summary** Dataset qualification experiments showed that for RCC QSR-QSAM produces reduced qualified datasets of cardinality approximately equal to half of the cardinality of a full qualified dataset. For CDC the reduction was much greater. The obtained qualified datasets are extremely compact, outperforming all the QSAMs presented before. However, the qualification time is extremely high, making this QSAM unsuited for being used on complete datasets.

QSCQ execution experiments showed that for both calculi the retrieval time is definitely too high. This is due to the high computational cost of algebraic closure algorithm. However it has to be considered that in this work we tested

a basic version of QSR-QSAM: We did not apply any heuristics for speeding up the algebraic closure. Moreover, the reconstruction process can be further sped up by using a more sophisticated version of this QSAM which in the qualification phase, beyond the qualitative relations, stores the inference paths used to perform the reduction. At retrieval time this information can be exploited to guide the reasoning performed with algebraic closure algorithm.

## 6.7 A Real-World Experiment

As a final experiment, we tested the most promising solutions presented above on a part of the OpenStreetMap[1] dataset of the city of Bremen, Germany. The tested dataset is depicted in Figure 6.14: it consists of polygons and lines, for a total of $\mathsf{N} = 6995$ objects. The purpose of this experiment, beyond a mere comparison of the tested QSAMs, is to demonstrate that the presented solution can work nicely also with datasets of higher cardinality, with spatial objects of different kinds (polygons and lines), and with dataset objects non-uniformly distributed.

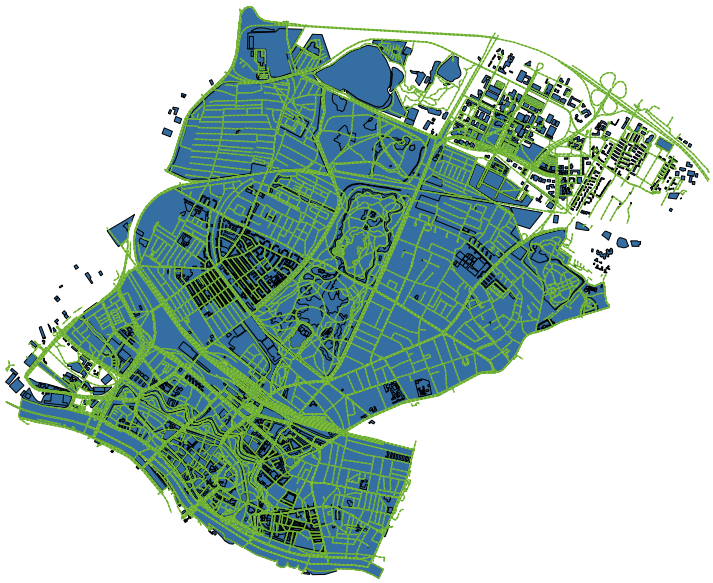


**Figure 6.14:** Bremen: test dataset.

We compared the performance of Qualitative Storage Layer QSAM (QSL-QSAM), *grid*-based Spatial Clustering QSAM (*grid* SC-QSAM), and R*-tree-based Spatial Clustering QSAM (R*-tree SC-QSAM). The parameters of the two SC-QSAM instances have been calibrated according to the results obtained in the experiments run on random datasets(cf. Sections 6.4 and 6.5, respectively).

[1] http://www.openstreetmap.org/

For *grid* SC-QSAM, a single scan of the spatial dataset allowed for the computation of the average object dimensions: The average object width is 141 and the average object height is 153. Accordingly we set the *grid* cell size $c = 147$.

For R*-tree SC-QSAM we opted for setting $M = 10$ in order to obtain an R*-tree of 5 levels and set the other parameters accordingly: $m = 4$, $rp = 30\%$, $cs = 4$ and $sl = 4$.

We tested both dataset qualification and QSCQ execution for both RCC and CDC. The results of the qualification experiments are reported in Table 6.2. QSL-QSAM produced worst results for both calculi and for both qualification time and qualified dataset cardinality. For RCC-qualification, *grid* SC-QSAM produced the most compact qualified dataset in time only slightly longer than R*-tree SC-QSAM. For CDC-qualification we obtained an opposite situation: R*-tree SC-QSAM produced the qualified dataset with lowest cardinality but *grid* SC-QSAM performed in shorter time.

| Calculus | QSAM | Qualif. Time ($hh : mm : ss$) | Qualif. Dataset Cardinality |
|---|---|---|---|
| RCC | QSL-QSAM | 01 : 48 : 55 | 48930025 |
| | *grid* SC-QSAM | 00 : 16 : 12 | 1594411 |
| | R*-tree SC-QSAM | 00 : 12 : 54 | 2151993 |
| CDC | QSL-QSAM | 03 : 38 : 32 | 48930025 |
| | *grid* SC-QSAM | 02 : 38 : 15 | 32247983 |
| | R*-tree SC-QSAM | 03 : 26 : 18 | 14222455 |

**Table 6.2:** Qualification outcomes for the OpenStreetMap dataset of the city of Bremen.

For QSCQ execution we tested one single-constraint and one multi-constraint QSCQ for each calculus. In order to be able to keep all the returned results in RAM, we opted for QSCQs with a high selectivity on the qualified spatial dataset. The main motivation beyond this choice is to avoid swapping to secondary memory which would affect the timing.

For RCC we tested the queries: $\{\text{PO}(x_1, x_2)\}$ and $\{\text{PO}(x_1, x_2), \text{NTPP}(x_2, x_3)\}$. For CDC we also tried a QSCQ with three constraints; the tested queries are: $\{\text{N}(x_1, x_2)\}$ and $\{\text{S}(x_1, x_2), \text{S}(x_3, x_2), \text{E}(x_3, x_1)\}$.

The outcomes of the experimentation are summarized in Table 6.3. Again, QSL-QSAM is outperformed by SC-QSAM. More precisely, *grid* SC-QSAM performed only slightly better than R*-tree SC-QSAM for resolving QSCQs encoding RCC constraints. In comparison with the retrieval time obtained with QSL-QSAM such a difference is negligible. For CDC, *grid* SC-QSAM performed almost as bad as QSL-QSAM whereas R*-tree SC-QSAM retained much better performance.

| Calculus | QSCQ | QSAM | Resp. Time ($hh:mm:ss$) | Retrieved Config. |
|---|---|---|---|---|
| RCC | $\{\text{PO}(x_1, x_2)\}$ | QSL-QSAM | 00 : 02 : 11 | 996 |
| | | *grid* SC-QSAM | 00 : 00 : 02 | |
| | | R*-tree SC-QSAM | 00 : 00 : 03 | |
| | $\left\{ \begin{array}{c} \text{PO}(x_1, x_2) \\ \text{NTPP}(x_2, x_3) \end{array} \right\}$ | QSL-QSAM | 00 : 02 : 10 | 961 |
| | | *grid* SC-QSAM | 00 : 00 : 02 | |
| | | R*-tree SC-QSAM | 00 : 00 : 03 | |
| CDC | $\{\text{N}(x_1, x_2)\}$ | QSL-QSAM | 00 : 01 : 59 | 268597 |
| | | *grid* SC-QSAM | 00 : 01 : 20 | |
| | | R*-tree SC-QSAM | 00 : 00 : 25 | |
| | $\left\{ \begin{array}{c} \text{S}(x_1, x_2) \\ \text{S}(x_3, x_2) \\ \text{E}(x_3, x_1) \end{array} \right\}$ | QSL-QSAM | 00 : 01 : 54 | 987970 |
| | | *grid* SC-QSAM | 00 : 01 : 06 | |
| | | R*-tree SC-QSAM | 00 : 00 : 24 | |

**Table 6.3:** Outcomes for QSCQ executions on the OpenStreetMap dataset of the city of Bremen.

## 6.8 Summary and Comparison

This chapter presented an empirical evaluation aimed at analyzing space-time tradeoff provided by the QSAMs introduced in Chapter 4. The criteria we evaluated are (*i*) dataset qualification time, (*ii*) space occupancy of the qualified dataset, and (*iii*) QSCQ execution time.

We ran two kinds of experiments—spatial dataset qualification and QSCQ execution—on two qualitative spatial calculi: RCC and CDC. Dataset qualification experiments have been carried out on a *qualification testbed* consisting of randomly generated datasets of different cardinalities. For QSCQ experimentation, we designed an *execution testbed* consisting of a set of 6 QSCQs—3 for each tested calculus. Such queries have been selected on the base of some preliminary experiments. They are sample QSCQs representing classes of execution behaviors, i.e. all the other QSCQs behave similarly to one of the selected queries.

In this section we provide and discuss an overall performance comparison of the presented solutions. For more specific details on the evaluation, refer to the previous sections.

In Section 6.3 we discussed Functional QSAM (F-QSAM) and QSL-QSAM. We argued that, since such QSAMs embody the two basic dataset qualification strategies presented in Section 3.5—runtime qualification and full pre-qualification—they provide goodness thresholds for the evaluation of the other QSAMs. F-QSAM provides an upper bound for QSCQ execution that should not be exceeded: A QSAM providing longer QSCQ execution times is not a good solution for query answering. Similarly QSL-QSAM provides a goodness threshold for dataset qualification:

A QSAM requiring more storage space than that required by QSL-QSAM does not provide a good solution for space occupancy. The experimental results, however, showed that most of the presented QSAMs behave better or nearly the same as QSL-QSAM in both dataset qualification and QSCQ execution. Accordingly, in the remainder of this section we use such an access method as a reference point for the comparison.

*Grid*-based and R*-tree-based SC-QSAM have been evaluated in Sections 6.4 and 6.5, respectively. The results obtained in the qualification experiments confirmed the reduction analysis carried out in Section 4.3.3: in order to allow for a good reduction, SC-QSAMs require a careful tuning of the spatial index used for doing the clustering. Our experiments demonstrate that, for both tested calculi, *grid* SC-QSAM performs optimally when the grid cell size $c$ (approximately) equals the average dataset object size $d$. In our experiments R*-tree SC-QSAM showed to perform as better as the parameter $sl$ is higher: For the tested parameter setups, $sl = 3$ provides the best performance.

In Section 6.6 we discussed QSR-based QSAM (QSR-QSAM). We pointed out that, given the big amount of inference templates and the computational cost of its qualification and retrieval algorithms, this QSAM has been only tested on a subset of the *qualification testbed* and only for one parameter value: $l = 1$.

Tables 6.4 and 6.5 report a comparison of the average performance obtained for the best parameter values with respect to QSL-QSAM. In Table 6.4 we computed the percentage of storage space reduction, whereas Table 6.4 summarizes the corresponding dataset qualification times.

| Calculus | QSAM | N = 100 | N = 250 | N = 500 | N = 750 | N = 1000 |
|---|---|---|---|---|---|---|
| RCC | *grid* SC-QSAM | 96.20% | 98.08% | 98.72% | 98.94% | 99.04% |
| | R*-tree SC-QSAM | 87.78% | 92.94% | 94.94% | 96.14% | 96.52% |
| | QSR-QSAM | 49.50% | 49.80% | 49.90% | – | – |
| CDC | *grid* SC-QSAM | 66.96% | 70.90% | 71.74% | 72.31% | 72.35% |
| | R*-tree SC-QSAM | 42.21% | 54.73% | 61.05% | 65.90% | 67.50% |
| | QSR-QSAM | 84.07% | 88.45% | – | – | – |

**Table 6.4:** Average qualified dataset reduction (in percentage) with respect to Qualitative Storage Layer QSAM as the cardinality $N$ of the spatial dataset varies.

For RCC, *grid*-based SC-QSAM showed to be the best solution, providing a spatial reduction comprised approximately between 96% and 99% of the full qualified dataset. It also provides the best solution in terms of qualification times, running in no longer than 36.52% of a full pre-qualification. Moreover, the values shown in the tables seem to have an asymptotic behavior, with the space reduction and qualification time asymptotes being approximately 99% and 4.8%, respectively.

For CDC the best spatial reduction is provided by QSR-QSAM. However, the qualification time is excessively high, making this solution unsuitable to be used on full spatial datasets. At the second place we find again *grid* SC-QSAM with a reduction comprised between 66.96% and 72.35% of the full qualified dataset. It also provides best qualification times, running in no longer than 45.96% of a

| Calculus | QSAM | N = 100 | N = 250 | N = 500 | N = 750 | N = 1000 |
|---|---|---|---|---|---|---|
| RCC | *grid* SC-QSAM | 36.52% | 15.86% | 8.58% | 6.05% | 4.86% |
| | R*-tree SC-QSAM | 31.62% | 18.09% | 13.04% | 10.24% | 9.93% |
| | QSR-QSAM | 4718% | 29423% | 116553% | – | – |
| CDC | *grid* SC-QSAM | 45.96% | 36.61% | 33.79% | 33.24% | 33.39% |
| | R*-tree SC-QSAM | 165.91% | 126.51% | 111.71% | 99.63% | 95.85% |
| | QSR-QSAM | 17609% | 182645% | – | – | – |

**Table 6.5:** Average qualification time (in percentage) with respect to Qualitative Storage Layer QSAM as the cardinality $N$ of the spatial dataset varies.

full pre-qualification. Also in this case the space-time performance seem to have an asymptotical behavior, with the reduction tending approximately to 72% and the qualification time to 33% of the performance obtained with QSL-QSAM.

| Calculus | QSCQ | QSAM | N = 100 | N = 250 | N = 500 | N = 750 | N = 1000 |
|---|---|---|---|---|---|---|---|
| RCC | $\mathcal{Q}_{RCC,1}$ | *grid* SC-QSAM | 138.06% | 1.82% | 0.32% | 0.21% | 0.19% |
| | | R*-tree SC-QSAM | 64.81% | 1.67% | 0.74% | 0.58% | 0.57% |
| | | QSR-QSAM | 195099% | 78420% | 143870% | – | – |
| | $\mathcal{Q}_{RCC,2}$ | *grid* SC-QSAM | 78.53% | 21.63% | 15.84% | 15.62% | 17.01% |
| | | R*-tree SC-QSAM | 86.16% | 21.76% | 16.35% | 15.54% | 16.74% |
| | | QSR-QSAM | 15070% | 25848% | 46085% | – | – |
| | $\mathcal{Q}_{RCC,3}$ | *grid* SC-QSAM | 72.83% | 28.07% | 26.51% | 24.22% | 25.41% |
| | | R*-tree SC-QSAM | 78.38% | 28.77% | 26.80% | 24.45% | 25.52% |
| | | QSR-QSAM | 28194% | 25073% | 46223% | – | – |
| CDC | $\mathcal{Q}_{CDC,1}$ | *grid* SC-QSAM | 76.62% | 3.02% | 17.59% | 26.90% | 26.55% |
| | | R*-tree SC-QSAM | 102.98% | 4.82% | 34.48% | 35.52% | 27.78% |
| | | QSR-QSAM | 22278687% | 1730227% | – | – | – |
| | $\mathcal{Q}_{CDC,2}$ | *grid* SC-QSAM | 641.18% | 39.12% | 45.76% | 40.16% | 50.39% |
| | | R*-tree SC-QSAM | 424.60% | 79.91% | 128.34% | 121.80% | 134.88% |
| | | QSR-QSAM | 2234344% | 422093% | – | – | – |
| | $\mathcal{Q}_{CDC,3}$ | *grid* SC-QSAM | 456.00% | 40.24% | 54.51% | 44.37% | 51.98% |
| | | R*-tree SC-QSAM | 314.95% | 78.83% | 137.87% | 133.48% | 139.43% |
| | | QSR-QSAM | 3173486% | 405640% | – | – | – |

**Table 6.6:** Average QSCQ response time (in percentage) with respect to Qualitative Storage Layer QSAM as the cardinality $N$ of the spatial dataset varies.

Performance comparison for QSCQ execution is summarized in Table 6.6 which reports the average retrieval time with respect to the QSL-QSAM retrieval.

The results show that for QSCQs encoding RCC relations the performance of *grid* SC-QSAM and R*-tree QSAM are comparable. When dealing with non-clustering relations—i.e. $\mathcal{Q}_{RCC,1}$ in our tests—these QSAMs provide a notable boost in response time: *grid* SC-QSAM executes in up to 0.19% of the time required by QSL-QSAM. The boost worsens in the execution of QSCQs encoding clustering relations because of the relation reconstruction that takes place during the retrieval: $\mathcal{Q}_{RCC,2}$ and $\mathcal{Q}_{RCC,3}$ are resolved, in up to approximately 17% and 25.5% of the time required by QSL-QSAM, respectively.

For QSCQs with CDC constraints, the results obtained show that *grid* SC-QSAM always outperforms R*-tree QSAM. Similarly to the case with queries encoding

RCC relations, the performance boost provided by the QSAM is higher when dealing with non-clustering relations than with clustering relations.

In conclusion, the result of the conducted experiments show that *grid* SC-QSAM provides the best performance on random generated datasets for both RCC and CDC and for both dataset qualification and QSCQ execution. However, the experimentation conducted on the OpenStreetMap dataset of Bremen (cf. Section 6.7) showed that R*-tree QSAM behaves better in terms of spatial reduction (cf. Table 6.2) as well as in QSCQ retrievals (cf. Table 6.3). The motivation is that the higher the cardinality of the spatial dataset, the more relevant the role played by the hierarchical structure of R*-tree.

As explained before, QSR-QSAM, in the basic version presented in this work, does not provide a suitable solution for real applications. More sophisticated versions of this QSAM can be developed and tested which we expect to lead to better performance.

Chapter 7

# Summary and Outlook

This chapter concludes the thesis with a summary and a discussion of the results achieved. Additionally it outlines some possible directions for future investigations stemming from the presented work.

## 7.1 Summary of Results

Despite continuous improvements, Geographic Information Systems (GIS) keep lacking instruments for interpreting and coping with spatial queries expressed in a human-friendly format, i.e. qualitative spatial descriptions. They provide interfaces mainly tailored for experts, denying the casual user the possibility to exploit their potentials and, consequently, drastically reducing the number of potential spatial data contributors and consumers.

As of today the casual user can exploit GIS capabilities mainly by resorting to some pre-defined interfaces which allow for accessing spatial information in a limited number of ways. One typology of access to spatial information, today largely unaccounted for, concerns the cases in which one wants to retrieve spatial information on the basis of a set of spatial constraints. In this work we faced the challenge of filling this gap.

Human beings have a natural predisposition for acquiring, reasoning on, and communicating spatial information that allows them to deal with spatial matters in a quite effortless manner. The processes underlying spatial reasoning in humans, mainly draws upon qualitative spatial representation. Qualitative Spatial Representation and Reasoning (QSR) is a well established field that also finds applications in GIS in that most typologies of spatial queries encode some sort of qualitative spatial relation: a predicate that spatial entities have to satisfy. Typically, in the literature, queries encoding different spatial predicates are studied separately and specialized solving strategies are developed for each type. In

this thesis we stressed the fact that all such spatial queries can be regarded as a single query type that we named Qualitative Spatial Relation Query. We presented a classification and a nomenclature of these queries. The classification is based on the level of indeterminacy of the spatial predicates encoded in a query whereas the nomenclature reflects the type of spatial request. Moreover, Qualitative Spatial Relation Queries have been ranked according to the computational cost required to resolve any of their instances. We decided to focus our work on the type hardest to solve that we named Qualitative Spatial Configuration Queries (QSCQs).

QSCQs can encode constraints from different qualitative spatial calculi, therefore they can be interpreted as a multi-calculus Qualitative Constraint Network (QCN), which, in turn can be represented as a directed hypergraph. The qualitative spatial relations among the objects in a spatial dataset can be computed into a so-called qualified dataset which can also be interpreted as a *multi-calculus* QCN (cf. Example 3.4). Accordingly, the problem of solving a QSCQ is equivalent to finding all the isomorphisms between the hypergraph representing the QSCQ and that representing the qualified dataset.

Initially, we assumed that the hypergraph representation of both the QSCQ and the spatial dataset were given and we derived from Ullmann's famous algorithm a basic matching procedure (cf. Section 3.4.1) suited for the special type of hypergraphs we treat.

Later on, we pointed out that, given the typical size of a real spatial dataset, the qualification operation required to generate a qualified dataset is computationally too onerous. Therefore, assuming that the hypergraph representation of the spatial dataset is given is not a reasonable hypothesis. Rather, dataset qualification has to be carefully accounted for in the QSCQ resolution process.

We presented two basic dataset qualification strategies (cf. Section 3.5): runtime qualification and pre-qualification. Runtime qualification is a purely functional approach that executes the qualification at retrieval time. It does not require any extra storage space but is highly inefficient for what concerns QSCQ retrieval time. Pre-qualification requires to extend the schema of the underlying spatial database with a set of Lookup Tables (LUTs) designed to accommodate the qualified dataset. We called such LUTs *relation tables* and the schema extension *qualitative storage layer*. Pre-qualification allows for significantly speeding up the retrievals but requires an enormous amount of space to store the qualified dataset. Moreover the elevated time required by qualification operation makes hard to deal with the dynamism of the real world: by the time the pre-qualification is completed the world has changed as well as its representation in the spatial database, calling for a new qualification.

In Chapter 4 we introduced the concept of Qualitative Spatial Access Method (QSAM). A QSAM (cf. Definition 4.1) is a special class of spatial access methods which applies a qualitative relation reduction/reconstruction paradigm to optimize the space-time tradeoff inevitably adduced by QSCQ solving strategies. The main purpose of a QSAM is to provide a solving strategy that takes into account

the dataset qualification problem by extending the spatial database with a qualitative storage layer and by only pre-computing a targeted subset of the whole qualified dataset. Non-stored relations have to be rebuildable at QSCQ execution time.

More specifically, a QSAM is composed of three elements: a qualifier function, a set of data structures, and a retriever function. The qualifier function takes care of qualifying the spatial dataset into a set of relation tables and, possibly, of storing some additional bits of information summarizing the strategy adopted to perform the reduction (if any). The retriever function exploits the information stored in the set of data structures to resolve a given QSCQ, possibly rebuilding missing qualitative spatial relations.

Four kinds of QSAMs have been proposed: Functional QSAM (F-QSAM), Qualitative Storage Layer QSAM (QSL-QSAM), Spatial Clustering QSAM (SC-QSAM), and QSR-based QSAM (QSR-QSAM). They all resort to hypergraph matching to solve a QSCQ but tackle differently the dataset qualification problem.

F-QSAM and QSL-QSAM (cf Sections 4.1 and 4.2, respectively) encode runtime qualification and full pre-qualification, respectively. As such, they provide goodness reference values to assess other QSAMs: A QSCQ retrieval should not take longer than the time employed by F-QSAM. Similarly, a QSAM producing a qualified dataset larger than that produced by QSL-QSAM does not provide a good approach for qualification.

SC-QSAM (cf. Section 4.3) is a family of QSAMs that resorts to a tile&cluster approach: It jointly exploits a special type of qualitative spatial relations called *clustering relations* (cf. Definition 4.2) together with a *spatial clustering index* (cf. Definition 4.5) to produce a reduced qualified dataset. If opportunely tuned, the spatial clustering index produces a subdivision of the spatial dataset that allows the qualifier function to compute and store a reduced number of qualitative relations. This approach provides a valuable compromise between the previous solutions: it shorten the retrieval time with respect to F-QSAM and produces more compact qualified datasets in a shorter time with respect to QSL-QSAM. At QSCQ execution time, the retriever function exploits the structure of the spatial clustering index to rebuild qualitative relations missing in the relation tables. The main shortcoming of this approach is that it requires a careful calibration of the spatial clustering index. Otherwise SC-QSAM might lead worse qualification and retrieval performance than those obtained with QSL-QSAM and F-QSAM, respectively. In this work we presented two special instances of SC-QSAM: (*i*) *grid*-based SC-QSAM (cf. Section 5.2.1) employs a uniform *grid* as a spatial clustering index. (*ii*) R*-tree-based SC-QSAM (cf. Section 5.2.2) resorts to R*-tree.

QSR-QSAM (cf. Section 4.4) exploits a novel data structure named inference graph (cf. Definition 4.9) to obtain a reduced qualified dataset from which it is possible to infer back all the missing relations by only resorting to symbolic reasoning. The advantages of this QSAM are that it is independent of the spatial calculus and it might be used to produce the minimum set of qualitative relations needed to describe a spatial dataset (cf. discussion on the reduction at the end

of Section 4.4.3). However, the temporal costs required by the qualifier and retriever functions makes QSR-QSAM unsuitable for being applied on a whole spatial dataset. Note that, in this work we have been mainly interested in drawing the foundations of this QSAM by defining the inference graph and a basic version of the reduction and reconstruction strategies. As will be discussed in Section 7.2.2, such basic strategies leave ample room for improvement.

In Chapter 5 we presented MyQual: a software framework for developing and benchmarking QSAMs. MyQual has been used to implement and test all the QSAMs introduced in this thesis, providing an empirical demonstration of its usefulness.

MyQual Toolbox is one of the four components provided by MyQual: it is an extensible repository for qualitative spatial calculi and QSAMs. As such, it provides a practical realization of a fundamental assumption of this work: We assumed the existence of a pool of qualitative relations rich enough to contain all the relations necessary to encode any qualitative spatial description produced by humans into a qualitative spatial relation query. Given the variety of expressions typical of spatial descriptions produced by humans, it is impossible to predict which relations will be needed. Contrarily, having an extensible repository of spatial calculi allows for enriching the pool at will when necessary.

Beyond being a development environment, MyQual aims at providing a distributed tool for bringing at the casual user reach the possibility of querying a GIS in a more intuitive way, i.e. via Qualitative Spatial Relation Queries.

In conclusion, the presented theoretical findings and the empirical results, prove that an accurate interplay of spatial access methods and a qualitative reduction/reconstruction paradigm is fundamental for enabling and efficiently solving Qualitative Spatial Configuration Queries in Geographic Information Systems. In particular, the empirical results indicated the Spatial Clustering QSAM family to be the most promising solution.

## 7.2 Future Work

The work presented in this thesis can be extended in several ways. In the following sections we outline some of the most attractive perspectives.

### 7.2.1 Minimum Qualitative Spatial Representation

The inference graph has been introduced in Section 4.4.1. We showed how it can be generated from the reasoning tables of a given qualitative spatial calculus and how it can be used to detect a reduced number of qualitative relations describing a spatial scene. We also provided a qualitative analysis of the reduction properties provided by this data structure and argued that it can be used to produce the *minimum* set of qualitative relations necessary to describe a spatial scene. A further investigation is required to formally verify the validity of this argumentation and to detect under which conditions a maximum reduction—i.e. a minimum description—is possible.

### 7.2.2 Empowering QSR-based QSAM

As demonstrated by the empirical evaluation carried out in Section 6.6 the presented version of QSR-QSAM is highly inefficient for what concern dataset qualification and QSCQ retrieval times. However, the presented solution can be empowered in several ways:

- The definition of Relaxed Strongly Connected Component (cf. Definition 4.11) can be modified to allow for the specification of a maximal length that the spanning hyperpaths connecting the nodes of each relaxed component do not have to exceed. This provides a certain control on the time required for the computation of the components. Defining such a maximal length allows for avoiding the exploration of long paths that, at the end, might not be relevant for computing a component. Of course, fixing the maximal allowed length at a too small value will produce components consisting of a reduced number of nodes, and consequently affect the overall reduction. Therefore, a more careful investigation has to be undertaken in order to define the best value for this new parameter.

- The inference paths used for doing the reduction, can be stored in an apposite data structure. Such information can be used in the retrieval phase as a sort of "reasoning map" to guide the reasoning operations performed by the algebraic closure algorithm: When iterating over a certain relation $\mathrm{R}(o_i, o_j)$ the algorithm can access the stored inference paths to immediately detect the relations which $\mathrm{R}(o_i, o_j)$ has to be composed with in order to re-generate one of the missing relations. Of course, also in this case there is a space-time tradeoff that has to be accounted for. The storage of inference paths allows for speeding up the relation reconstruction phase, and, thus, the QSCQ response time, but on the other hand requires an increased amount of storage space. This tradeoff has to be investigated to detect what is the percentage of inference path that can be stored without leading to an excessive worsening of the storage performance.

- The actual reduction and reconstruction phases allow for using inter-calculi inference templates. That is, new inference rules connecting relations from different calculi can be used in the dataset qualification and in the QSCQ execution phases. For example one might note that, if an object $o_i$ is to the *north* of and object $o_j$ then it follows that $o_i$ is also *disconnected* from $o_j$. This inference rule can be encoded as an inter-calculus inference template. Some work in this direction has been done, for example, by Sharma (1996) who developed a series of inter-calculus reasoning tables. Inter-calculus templates allow for instantiating inference paths that connect inference graphs otherwise disconnected—i.e. generated from different calculi. It is of special interest to investigate how such a connection influences the reduction and the reconstruction performed by QSR-QSAM.

### 7.2.3 Other Qualitative Spatial Calculi and Inter-calculus QSCQs

One main objective of this work was to develop a theoretical framework general enough to deal with qualitative spatial calculi of any arity. Moreover, the framework also allows for dealing with qualitative relation queries encoding simultaneously relations concerning multiple spatial aspects—i.e. belonging to different calculi.

In this work, we limited ourselves to implementations of basic cases: we only dealt with two binary calculi—the Region Connection Calculus (RCC) and the Cardinal Direction Calculus (CDC)—and with QSCQs encoding relations from the same calculus. Further investigations may concern the implementation of other calculi, possibly of higher arity, and the experimentation of the presented techniques on queries encoding relations from multiple calculi.

### 7.2.4 Further Investigation on Spatial Clustering QSAM

The Spatial Clustering QSAM (SC-QSAM) family has been introduced in Section 4.3. In this work we presented and implemented two possible instances of such a family: one based on a *grid* index, the other on an R*-tree. Further investigations can be carried out that aim at analyzing other SC-QSAM instances, that is, the employment of other data structures and space partitioning methods to be used as a spatial clustering index.

Moreover, of special interest is the development of techniques to automatically detect spatial clustering relations among those provided by a qualitative spatial calculus. For this investigation, the results of Theorem 4.1 can be used as a starting point.

### 7.2.5 Mixed SC–QSR QSAM

It was pointed out in the discussion of QSR-based QSAM (QSR-QSAM) (and confirmed by the empirical evaluation) that the high temporal cost required by the qualification and retrieval functions, make QSR-QSAM not suited for being applied to whole spatial datasets. A possible solution to overcome this inconvenience is that of developing a mixed Spatial Clustering–QSR QSAM. Dataset qualification consists in applying first Spatial Clustering QSAM (SC-QSAM) to exploit clustering relations for computing a reduced number of inter-cluster object relations. Then, applying QSR-QSAM on each object cluster allows for reducing the number of in-cluster relations. QSCQ execution can be done by applying algebraic closure on each cluster and subsequently calling the SC-QSAM retriever function.

### 7.2.6 Dynamic and Parallel QSAMs

In Section 3.1 we introduced a generalization for queries encoding spatial predicates that allow for looking at them as a single query type. We called such a generalization *qualitative spatial relation query* and presented Qualitative Spatial Access Methods (QSAMs) in Chapter 4: a class of Spatial Access Methods (SAMs) tailored for solving the hardest variant of qualitative spatial relation query. In this thesis we investigated how to efficiently access qualitative spatial information from a spatial database, restricting ourselves to the consideration of static datasets, i.e. we only focused on insert and search operations. In order to make the presented solutions practically usable, they have to be provided with algorithms for dealing with dynamic datasets, that is functions for taking into account deletion and update operations have to be developed.

A further possible investigation direction concerns the parallelization of the algorithmic framework we presented. In Chapter 5 we discussed MyQual, a software framework for the development of QSAMs. MyQual has been designed to be a distributed framework, in the sense that a local installation can use tools located on a remote bounded machine and simultaneously provide tools to be used by remote installations of MyQual. So far, the hardware and computational resources required for the execution of a certain operation, are provided by the machine hosting the tool being used. The parallelization of the algorithmic framework would allow to further enhance the distributed fashion of MyQual, allowing for moving a step towards cloud computing: the computation required by a certain method can be split and distributed over the network of MyQual installations.

### 7.2.7 Treating Disjunctive Relations and Inconsistency

This work focused on the definition of a theoretical framework and on the development of basic methods and techniques for the resolution of Qualitative Spatial Configuration Queries (QSCQs). To allow for a lean analysis we set the assumption that the QSCQs under consideration were consistent—i.e. they do not encode spatial relations conflicting with each other. Moreover, we narrowed down our investigation to the case in which the relations encoded in a QSCQ were base relations—i.e. no disjunction of relations appears in a query.

A further extension of the presented solutions consists in the development of techniques that can deal with uncertainty and inconsistency. Some work have already been conducted on these issues (cf. Clementini *et al.*, 1994; Wallgrün *et al.*, 2010, for instance). A possible research direction, then, concerns the integration of previous work with the results presented in this thesis in order to develop QSCQ solving techniques capable of dealing with queries encoding disjunctive relations and of opportunely handling inconsistency.

### 7.2.8 Contributing Qualitative Spatial Information in VGI Projects

Volunteered Geographic Information (VGI) is a particular form of user-generated content in that it assumes the involvement of volunteers to collect and disseminate spatial information. Probably, the most known form of VGI, is represented by web-based projects like OpenStreetMap[1], which aims at collaboratively producing a free editable map of the world.

There are several ways a volunteer can contribute spatial information into a web GIS. One of the easiest consists in surveying spatial data by means of a Global Positioning System (GPS) device and uploading it to the server hosting the GIS.

However, the number of real contributors is strongly affected by the technological barriers risen by such a method: (*i*) Although GPS technology is spreading very fast, not everyone posseses a device with a GPS antenna. (*ii*) Not everyone having a GPS device knows how to use it. (*iii*) In order to be uploaded, the surveyed data has to be cleaned and tagged. This requires a certain level of expertise in GIS.

As discussed in (Fogliaroni *et al.*, 2010), providing the casual user with the possibility of contributing geographic information via natural spatial descriptions (cf. Sections 1.2.1 and 2.2.1) would allow for overcome such technological barriers.

MyQual, the software framework we developed for enabling QSCQs, can serve a fundamental role in this perspective: It provides the basic components to allow the underlying spatial database to store and retrieve qualitative spatial information. As a consequence, it also provides basic methods for inputting spatial information given in a qualitative form. That is, a volunteer can give a textual or pictorial description of the spatial entities he wants to contribute on, relating them to a set of entities in the spatial database underlying a GIS. The given spatial description can be encoded into a qualitative spatial relation query whose single relations can be stored in the qualitative storage layer. Although not having geometries associated, the new spatial entities are now present in the database and can be retrieved via a qualitative spatial relation query.

Therefore, the ideas presented in this thesis can be used as a base for developing a new contribution form of spatial information that can be used to empower already established VGI projects like OpenStreetMap.

[1] http://www.openstreetmap.org/

# References


Allen, J. (1983). Maintaining knowledge about temporal intervals. *Communications of the ACM*, **26**(11), 832–843. (17)

Ausiello, G., D'Atri, A. & Saccà, D. (1983). Graph algorithms for functional dependency manipulation. *Journal of the ACM (JACM)*, **30**(4), 752–766. (13)

Ausiello, G., D'Atri, A. & Saccà, D. (1986). Minimal representation of directed hypergraphs. *SIAM Journal on Computing*, **15**(2), 418–431. (13)

Ausiello, G., Franciosa, P. & Frigioni, D. (2001). Directed hypergraphs: Problems, algorithmic results, and a novel decremental approach. In A. Restivo, S. Della Rocca & L. Roversi, eds., *Theoretical Computer Science*, vol. 2202 of *Lecture Notes in Computer Science*, 312–328, Springer-Verlag. (12, 13, 14)

Balbiani, P., Condotta, J. & Del Cerro, L. (1998). A model for reasoning about bidimensional temporal relations. In A.G. Cohn, L. Schubert & S.S. C., eds., *Proceedings of the Sixth International Conference on Principles of Knowledge Representation and Reasoning*, 124–130, Morgan Kaufmann. (27)

Bartie, P., Reitsma, F., Clementini, E. & Kingham, S. (2011). Referring expressions in location based services: The case of the 'opposite' relation. In O. De Troyer, C. Bauzer Medeiros, R. Billen, P. Hallot, A. Simitsis & H. Van Mingroot, eds., *Advances in Conceptual Modeling. Recent Developments and New Directions*, vol. 6999 of *Lecture Notes in Computer Science*, 231–240, Springer-Verlag. (25)

Beckmann, N., Kriegel, H.P., Schneider, R. & Seeger, B. (1990). The R*-tree: An efficient and robust access method for points and rectangles. In *Proceedings of the 1990 ACM SIGMOD International Conference on Management of Data*, SIGMOD '90, 322–331, ACM. (39, 76, 123, 124)

Bentley, J.L. & Friedman, J.H. (1979). Data structures for range searching. *ACM Computing Surveys (CSUR)*, **11**(4), 397–409. (38)

Berge, C. (1976). *Graphs and hypergraphs*, vol. 6. Elsevier. (12)

Berge, C. (1989). *Hypergraphs: Combinatorics of finite sets*, vol. 45. North-Holland. (12)

Billen, R. & Clementini, E. (2004a). Introducing a reasoning system based on ternary projective relations. In P. Fisher, ed., *Developments in Spatial Data Handling*, 381–394, Springer-Verlag. (27)

BILLEN, R. & CLEMENTINI, E. (2004b). A model for ternary projective relations between regions. In E. Bertino, S. Christodoulakis, D. Plexousakis, V. Christophides, M. Koubarakis, K. Böhm & E. Ferrari, eds., *Advances in Database Technology – EDBT 2004*, vol. 2992 of *Lecture Notes in Computer Science*, 537–538, Springer-Verlag. (27)

BILLEN, R. & ZLATANOVA, S. (2003). 3D spatial relationships model: A useful concept for 3D cadastre? *Computers, Environment and Urban Systems*, **27**(4), 411–425. (25)

BITTNER, T. & DONNELLY, M. (2007). A formal theory of qualitative size and distance relations between regions. In *Proceedings of the 21st International Workshop on Qualitative Reasoning*. (29)

BRAKATSOULAS, S., PFOSER, D. & THEODORIDIS, Y. (2002). Revisiting R-tree construction principles. In Y. Manolopoulos & P. Návrat, eds., *Advances in Databases and Information Systems*, vol. 2435 of *Lecture Notes in Computer Science*, 17–24, Springer-Verlag. (40)

CALCINELLI, D. & MAINGUENAUD, M. (1994). Cigales, a visual query language for geographical information system: The user interface. *Journal of Visual Languages and Computing*, **5**(2), 113–132. (41)

CHANG, N. & FU, K. (1980). Query-By-Pictorial-Example. *IEEE Transactions on Software Engineering*(6), 519–524. (40)

CLARK, K.L. (1978). Negation as failure. In *Logic and Databases*, vol. 1, 293–322, Plenum Publishing Corporation. (108)

CLEMENTINI, E. & DI FELICE, P. (1997). A global framework for qualitative shape description. *GeoInformatica*, **1**(1), 11–27. (22, 29, 31)

CLEMENTINI, E. & DI FELICE, P. (2000). Spatial operators. *SIGMOD Record*, **29**(3), 31–38. (26, 36)

CLEMENTINI, E., SHARMA, J. & EGENHOFER, M.J. (1994). Modelling topological spatial relations: Strategies for query processing. *Computers & Graphics*, **18**(6), 815–822. (41, 57, 143)

CLEMENTINI, E., DI FELICE, P. & HERNÁNDEZ, D. (1997). Qualitative representation of positional information. *Artificial Intelligence*, **95**(2), 317–356. (23, 28, 31)

COHN, A.G. (1995). A hierarchical representation of qualitative shape based on connection and convexity. In A. Frank & W. Kuhn, eds., *Spatial Information Theory A Theoretical Basis for GIS*, vol. 988 of *Lecture Notes in Computer Science*, 311–326, Springer-Verlag, 10.1007/3-540-60392-1_20. (29)

COHN, A.G. (1997). Qualitative spatial representation and reasoning techniques. In G. Brewka, C. Habel & B. Nebel, eds., *Proceedings of the 21st Annual German Conference on Artificial Intelligence: Advances in Artificial Intelligence*, vol. 1303 of *Lecture Notes in Computer Science*, 1–30, Springer-Verlag. (17)

Cohn, A.G. & Hazarika, S.M. (2001). Qualitative spatial representation and reasoning: An overview. *Fundamenta Informaticae*, **46**(1-2), 1–29. (17)

Cohn, A.G. & Renz, J. (2008). Qualitative spatial representation and reasoning. In V.L. Frank van Harmelen & B. Porter, eds., *Handbook of Knowledge Representation*, vol. 3 of *Foundations of Artificial Intelligence*, 551–596, Elsevier. (17, 19)

Cohn, A.G., Bennett, B., Gooday, J. & Gotts, N.M. (1997). Qualitative spatial representation and reasoning with the region connection calculus. *GeoInformatica*, **1**(3), 275–316. (27)

Comer, D. (1979). Ubiquitous b-tree. *ACM Computing Surveys (CSUR)*, **11**(2), 121–137. (38, 39, 57)

Condotta, J., Ligozat, G. & Saade, M. (2006). A generic toolkit for n-ary qualitative temporal and spatial calculi. In *TIME 2006. Proceedings of the Thirteenth International Symposium on Temporal Representation and Reasoning*, vol. 6, 78–86. (21)

Conte, D., Foggia, P., Sansone, C. & Vento, M. (2004). Thirty years of graph matching in pattern recognition. *International journal of pattern recognition and artificial intelligence*, **18**(3), 265–298. (51)

Cui, Z., Cohn, A.G. & Randell, D.A. (1993). Qualitative and topological relationships in spatial databases. In D. Abel & B. Chin Ooi, eds., *Proceedings of the Third International Symposium on Advances in Spatial Databases*, vol. 692 of *Lecture Notes in Computer Science*, 296–315, Springer-Verlag. (80)

Dechter, R. (1992). Constraint networks. In S.C. Shapiro, ed., *Encyclopedia of Artificial Intelligence (2nd Edition)*, vol. 1, 276–285, John Wiley & Sons, Inc. (29)

Dechter, R. (2003). *Constraint processing*. The Morgan Kaufmann Series in Artificial Intelligence, Morgan Kaufmann. (29)

Douglas, K. & Douglas, S. (2005). *PostgreSQL. A comprehensive guide to building, programming, and administering PostgreSQL databases (2nd Edition)*. Sams Publishing. (98)

Düntsch, I., Wang, H. & McCloskey, S. (2001). A relation–algebraic approach to the region connection calculus. *Theoretical Computer Science*, **255**(1-2), 63–83. (21)

Dylla, F. & Moratz, R. (2004). Empirical complexity issues of practical qualitative spatial reasoning about relative position. In *Workshop on Spatial and Temporal Reasoning at ECAI 2004*. (31)

Egenhofer, M. (1996). Spatial-Query-By-Sketch. In W. Citrin & M. Burnett, eds., *Proceedings of the 1996 IEEE Symposium on Visual Languages*, 60–67. (2, 41)

EGENHOFER, M. & SHARMA, J. (1992). Topological consistency. In P. Bresnahan, E. Corwin & D. Cowen, eds., *Proceeding of the Fifth International Symposium on Spatial Data Handling*, vol. 2, 335–343. (94)

EGENHOFER, M. & SHARMA, J. (1993). Assessing the consistency of complete and incomplete topological information. *Geographical Systems*, **1**(1), 47–68. (22, 31, 94)

EGENHOFER, M.J. (1989). A formal definition of binary topological relationships. In W. Litwin & H.J. Schek, eds., *3rd International Conference, FODO 1989 on Foundations of Data Organization and Algorithms*, vol. 367 of *Lecture Notes in Computer Science*, 457–472, Springer-Verlag. (26, 57)

EGENHOFER, M.J. (1991). Reasoning about binary topological relations. In O. Günther & H.J. Schek, eds., *Proceedings of the Second International Symposium on Advances in Spatial Databases*, vol. 525 of *Lecture Notes in Computer Science*, 143–160, Springer-Verlag. (26, 27, 57)

EGENHOFER, M.J. (1994). Spatial sql: A query and presentation language. *IEEE Transactions on Knowledge and Data Engineering*, **6**(1), 86–95. (40)

ELMASRI, R. & NAVATHE, S. (2008). *Fundamentals of database systems*, vol. 2. Pearson Education India. (13)

ESRI (1998). Shapefile technical description. Tech. rep., Environmental Systems Research Institute, Inc. (32)

FAGIN, R., NIEVERGELT, J., PIPPENGER, N. & STRONG, H.R. (1979). Extendible hashing: A fast access method for dynamic files. *ACM Transactions on Database Systems (TODS)*, **4**(3), 315–344. (57)

FALOMIR, Z., MARTÍ, I., VIANA, W., MUSEROS, L. & ESCRIG, M.T. (2010). A pragmatic approach for qualitative shape and qualitative colour similarity matching. In R. Alquézar, A. Moreno & J. Aguilar, eds., *Artificial Intelligence Research and Development – Proceedings of the 13th International Conference of the Catalan Association for Artificial Intelligence*, vol. 220 of *Frontiers in Artificial Intelligence and Applications*, 281–290, IOS Press. (29)

FINKEL, R.A. & BENTLEY, J.L. (1974). Quad-trees a data structure for retrieval on composite keys. *Acta Informatica*, **4**, 1–9. (76)

FOGLIARONI, P., WALLGRÜN, J.O., CLEMENTINI, E., TARQUINI, F. & WOLTER, D. (2009). A qualitative approach to localization and navigation based on visibility information. In K.S. Hornsby, C. Claramunt, M. Denis & G. Ligozat, eds., *Proceedings of the 9th International Conference on Spatial Information Theory*, COSIT'09, 312–329, Springer-Verlag. (29)

FOGLIARONI, P., DE FELICE, G. & WALLGRÜN, J.O. (2010). A qualitative perspective on volunteered geographic information. In *GIScience 2010 - Workshop on the Role of Volunteered Geographic Information in Advancing Science*. (144)

Fogliaroni, P., De Felice, G., Schmid, F. & Wallgrün, J.O. (2011). Managing qualitative spatial information to support Query-by-Sketch. In J. Wang, Bölemann, M. Chipofya, A. Schwering & J.O. Wallgrün, eds., *Understanding and Processing Sketch Maps (COSIT'11)*, vol. 42 of *ifgiPrints*, 21 – 32, IOS Press. (119)

Forbus, K.D. (2008). Qualitative modeling. In V.L. Frank van Harmelen & B. Porter, eds., *Handbook of Knowledge Representation*, vol. 3 of *Foundations of Artificial Intelligence*, 361–393, Elsevier. (16)

Forbus, K.D., Nielsen, P. & Faltings, B. (1987). Qualitative kinematics: A framework. In *Proceedings of the 10th International Joint Conference on Artificial Intelligence - Volume 1*, IJCAI'87, 430–435, Morgan Kaufmann Publishers Inc. (31)

Forbus, K.D., Nielsen, P. & Faltings, B. (1991). Qualitative spatial reasoning: The CLOCK project. *Artificial Intelligence*, **51**(1-3), 417–471. (16, 31)

Frank, A.U. (1982). mapquery: Data base query language for retrieval of geometric data and their graphical representation. *ACM SIGGRAPH Computer Graphics*, **16**(3), 199–207. (40)

Frank, A.U. (1991). Qualitative spatial reasoning with cardinal directions. In H. Kaindl, ed., *Proceedings of the 7th Austrian Conference on Artificial Intelligence, ÖGAI*, vol. 287 of *Informatik-Fachberichte*, 157–167, Springer-Verlag. (27)

Franklin, W.R. (1978). *Combinatorics of hidden surface algorithms*. Ph.D. thesis, Center for research in computing technology, Harvard University. (38)

Franklin, W.R. (1984). Adaptive grids for geometric operations. *Cartographica: The International Journal for Geographic Information and Geovisualization*, **21**(2-3), 160–167. (38)

Franklin, W.R. (1989). Uniform grids: A technique for intersection detection on serial and parallel machines. In *Proceedings of the Ninth International Symposium on Computer-Assisted Cartography*, 100–109, American Congress on Surveying and Mapping. (38)

Freksa, C. (1991a). Conceptual neighborhood and its role in temporal and spatial reasoning. In M.G. Singh & L. Travé-Massuyès, eds., *Decision Support Systems and Qualitative Reasoning*, 181–187, North-Holland. (41)

Freksa, C. (1991b). Qualitative spatial reasoning. In D.M. Mark & A.U. Frank, eds., *Cognitive and linguistic aspects of geographic space*, 361–372. (16, 17)

Freksa, C. (1992). Using orientation information for qualitative spatial reasoning. In A. Frank, I. Campari & U. Formentini, eds., *Theories and Methods of Spatio-Temporal Reasoning in Geographic Space*, vol. 639 of *Lecture Notes in Computer Science*, 162–178, Springer-Verlag. (27, 29)

FREKSA, C. & RÖHRIG, R. (1993). Dimensions of qualitative spatial reasoning. In P. Carreté & M. Singh, eds., *Proceedings of Qualitative reasoning and decision technologies (QUARDET'93)*, 483–492. (22, 26)

FREKSA, C. & ZIMMERMANN, K. (1992). On the utilization of spatial structures for cognitively plausible and efficient reasoning. In *Proceedings of the IEEE International Conference on Systems, Man and Cybernetics*, 261–266, IEEE. (20, 27)

GAEDE, V. & GÜNTHER, O. (1998). Multidimensional access methods. *ACM Computing Surveys (CSUR)*, **30**(2), 170–231. (36, 37, 38)

GALLO, G., LONGO, G., PALLOTTINO, S. & NGUYEN, S. (1993). Directed hypergraphs and applications. *Discrete Applied Mathematics*, **42**(2-3), 177–201. (12, 13, 14)

GEREVINI, A. & RENZ, J. (1998). Combining topological and qualitative size constraints for spatial reasoning. In M. Maher & J.F. Peugeot, eds., *Proceedings of the 4th International Conference on Principles and Practice of Constraint Programming*, Lecture Notes in Computer Science, 220–234, Springer-Verlag. (31, 32)

GEREVINI, A. & RENZ, J. (2002). Combining topological and size information for spatial reasoning. *Artificial Intelligence*, **137**(1-2), 1–42. (31, 32)

GOODCHILD, M.F. (2007). Citizens as sensors: The world of volunteered geography. *GeoJournal*, **69**(4), 211–221. (1)

GOYAL, R. & EGENHOFER, M. (2000). Consistent queries over cardinal directions across different levels of detail. In *Proceedings of the 11th International Workshop on Database and Expert System Applications*, 876–880, IEEE Computer Society. (25)

GOYAL, R. & EGENHOFER, M. (in press). Cardinal directions between extended spatial objects. *IEEE Transactions on Knowledge and Data Engineering*, available at http://www.spatial.maine.edu/∼max/RJ36.html. (27, 41, 58)

GOYAL, R.K. & EGENHOFER, M.J. (1997). The direction-relation matrix: A representation for directions relations between extended spatial objects. In *The Annual Assembly and the Summer Retreat of University Consortium for Geographic Information Systems Science*. (28, 41, 58)

GUESGEN, H. (1989). Spatial reasoning based on Allen's temporal logic. Tech. Rep. TR-89-049, International Computer Science Institute at Berkley, California, 1989. (17, 27)

GÜTING, R. (1994). An introduction to spatial database systems. *The VLDB Journal*, **3**(4), 357–399. (34)

GUTTMAN, A. (1984). R-trees: A dynamic index structure for spatial searching. In *Proceedings of the 1984 ACM SIGMOD International Conference on Management of Data*, SIGMOD '84, 47–57, ACM. (39, 76, 123)

Hernandez, D. (1994). *Qualitative representation of spatial knowledge*, vol. 804 of *Lecture Notes in Computer Science*. Springer-Verlag. (31)

Hernandez, D., Clementini, E. & Di Felice, P. (1995). Qualitative distances. In A. Frank & W. Kuhn, eds., *Spatial Information Theory A Theoretical Basis for GIS*, vol. 988 of *Lecture Notes in Computer Science*, 45–57, Springer-Verlag. (28)

Ingram, K.J. & Phillips, W.W. (1987). Geographic information processing using a sql-based query language. In N.R. Chrisman, ed., *Proceedings of the 8th International Symposium on Computer-Assisted Cartography*, 326–335, American Congress on Surveying and Mapping. (40)

Joseph, T. & Cardenas, A. (1988). picquery: A high level query language for pictorial database management. *IEEE Transactions on Software Engineering*, **14**(5), 630–638. (40)

Klippel, A. (2003). Wayfinding choremes. In W. Kuhn, M. Worboys & S. Timpf, eds., *Spatial Information Theory. Foundations of Geographic Information Science*, vol. 2825 of *Lecture Notes in Computer Science*, 301–315, Springer-Verlag. (18)

Knauff, M., Rauh, R. & Renz, J. (1997). A cognitive assessment of topological spatial relations: Results from an empirical investigation. In S. Hirtle & A. Frank, eds., *Proceedings of the International Conference on Spatial Information Theory: A Theoretical Basis for GIS*, vol. 1329 of *Lecture Notes in Computer Science*, 193–206, Springer-Verlag. (32)

Lee, Y. & Chin, F. (1995). An iconic query language for topological relationships in gis. *International Journal of Geographical Information Systems*, **9**(1), 25–46. (41)

Levinson, S.C. (1996). Frames of reference and Molyneux's question: Crosslinguistic evidence. In P. Bloom, M.A. Peterson, L. Nadel & M.F. Garrett, eds., *Language and Space*, 109–169, MIT Press. (23)

Li, S. (2007). Combining topological and directional information for spatial reasoning. In *Proceedings of the 20th international joint conference on Artifical intelligence*, IJCAI'07, 435–440, Morgan Kaufmann Publishers Inc. (32)

Li, S. & Cohn, A.G. (2009). Reasoning with topological and directional spatial information. *CoRR*, **abs/0909.0122**. (32)

Ligozat, G. (1998). Reasoning about cardinal directions. *Journal of Visual Languages and Computing*, **9**(1), 23–44. (27)

Litwin, W. (1980). Linear hashing: A new tool for file and table addressing. In *Proceedings of the sixth international conference on Very Large Data Bases - Volume 6*, VLDB '80, 212–223, VLDB Endowment Inc. (57)

Liu, W., Li, S. & Renz, J. (2009). Combining rcc-8 with qualitative direction calculi: Algorithms and complexity. In *Proceedings of the 21st International Joint Conference on Artifical Intelligence*, 854–859, Morgan Kaufmann Publishers Inc. (32)

LLOYD, S. (1982). Least squares quantization in PCM. *IEEE Transactions on Information Theory*, **28**(2), 129–137. (40)

LONGLEY, P., GOODCHILD, M., MAGUIRE, D. & RHIND, D. (2005). *Geographic Information Systems and Science*. John Wiley & Sons, Ltd. (33)

MACKWORTH, A. (1977). Consistency in networks of relations. *Artificial Intelligence*, **8**(1), 99–118. (30)

MONTANARI, U. (1974). Networks of constraints: Fundamental properties and applications to picture processing. *Information Science*, **7**(2), 95–132. (29, 30)

MORATZ, R. (2006). Representing relative direction as a binary relation of oriented points. In *Proceedings of the 2006 conference on ECAI 2006: 17th European Conference on Artificial Intelligence August 29 – September 1, 2006, Riva del Garda, Italy*, 407–411, IOS Press. (27)

MORATZ, R. & RAGNI, M. (2008). Qualitative spatial reasoning about relative point position. *Journal of Visual Languages & Computing*, **19**(1), 75 – 98. (29)

MORATZ, R. & WALLGRÜN, J.O. (2003). Propagation of distance and orientation intervals. In *Proceedings of 2003 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, vol. 4, 3245–3250. (29)

MORATZ, R., RENZ, J. & WOLTER, D. (2000). Qualitative spatial reasoning about line segments. In W. Horn, ed., *Proceedings of the 14th European Conference on Artificial Intelligence, ECAI 2000*, 234–238, IOS Press. (27)

MORATZ, R., DYLLA, F. & FROMMBERGER, L. (2005). A relative orientation algebra with adjustable granularity. In *Proceedings of the Workshop on Agents in Real-Time and Dynamic Environments (IJCAI 05)*. (23, 27)

MUKERJEE, A. & JOE, G. (1990). A qualitative model for space. In *Proceedings of the eighth National conference on Artificial intelligence - Volume 1*, AAAI'90, 721–727, AAAI Press. (27)

NIEVERGELT, J., HINTERBERGER, H. & SEVCIK, K.C. (1984). The grid file: An adaptable, symmetric multikey file structure. *ACM Trans. Database Syst.*, **9**(1), 38–71. (39, 76)

OPENGIS CONSORTIUM (1998). OpenGIS simple features specification for sql. Tech. rep., OpenGIS Consortium (OGC). (34, 35, 40, 105)

PAPADIAS, D. (1994). *Relation-based representation of spatial knowledge*. Ph.D. thesis, National Technical University of Athens. (31)

PAPADIAS, D. & SELLIS, T. (1994a). A pictorial language for the retrieval of spatial relations from image databases. In *the Proceedings of the 6th International Symposium on Spatial Data Handling (SDH)*, Taylor Francis. (41)

Papadias, D. & Sellis, T. (1994b). Qualitative representation of spatial knowledge in two-dimensional space. *The VLDB Journal*, **3**, 479–516. (31)

Papadias, D. & Sellis, T. (1995). A pictorial Query-By-Example language. *Journal of Visual Languages & Computing*, **6**(1), 53–72. (41)

Papadias, D., Sellis, T., Theodoridis, Y. & Egenhofer, M.J. (1995). Topological relations in the world of minimum bounding rectangles: A study with r-trees. In *Proceedings of the 1995 ACM SIGMOD international conference on Management of data*, SIGMOD '95, 92–103, ACM. (31)

Pfoser, D. (2011). On user-generated geocontent. In D. Pfoser, Y. Tao, K. Mouratidis, M. Nascimento, M. Mokbel, S. Shekhar & Y. Huang, eds., *Advances in Spatial and Temporal Databases*, vol. 6849 of *Lecture Notes in Computer Science*, 458–461, Springer-Verlag. (2)

Piaget, J. & Inhelder, B. (1967). *The child's conception of space*. Routledge. (26)

Raiman, O. (1991). Order of magnitude reasoning. *Artificial Intelligence*, **51**(1-3), 11–38. (29)

Ramsey, P. (2005). *PostGIS manual*. Refractions Research Coroporation. (98)

Randell, D.A. & Cohn, A.G. (1989). Modelling topological and metrical properties of physical processes. In R.J. Brachman, H.J. Levesque & R. Reiter, eds., *Proceedings of the 1st International Conference on Principles of Knowledge Representation and Reasoning (KR'89)*, 357–368, Morgan Kaufmann. (26)

Randell, D.A., Cui, Z. & Cohn, A. (1992). A spatial logic based on regions and connection. In B. Nebel, C. Rich & W. Swartout, eds., *Principles of Knowledge Representation and Reasoning: Proceedings of the Third International Conference (KR'92)*, 165–176, Morgan Kaufmann. (26)

Renz, J. & Ligozat, G. (2005). Weak composition for qualitative spatial and temporal reasoning. In P. van Beek, ed., *Principles and Practice of Constraint Programming (CP 2005)*, vol. 3709 of *Lecture Notes in Computer Science*, 534–548, Springer-Verlag. (21)

Renz, J. & Mitra, D. (2004). Qualitative direction calculi with arbitrary granularity. In C. Zhang, H. W. Guesgen & W.K. Yeap, eds., *PRICAI 2004: Trends in Artificial Intelligence*, vol. 3157 of *Lecture Notes in Computer Science*, 65–74, Springer-Verlag. (27)

Renz, J. & Nebel, B. (2007). Qualitative spatial reasoning using constraint calculi. In M. Aiello, I. Pratt-Hartmann & J. Benthem, eds., *Handbook of Spatial Logics*, 161–215, Springer-Verlag. (22, 31)

Roussopoulos, N. & Leifker, D. (1985). Direct spatial search on pictorial databases using packed r-trees. In *Proceedings of the 1985 ACM SIGMOD international conference on Management of data*, SIGMOD '85, 17–31, ACM. (40)

RUSSELL, S.J. & NORVIG, P. (2003). *Artificial Intelligence: A modern approach*. Pearson Education. (16)

SAMET, H. (2006). *Foundations of multidimensional and metric data structures*. Morgan Kaufmann. (38)

SANTOS, P., DEE, H. & FENELON, V. (2009). Qualitative robot localisation using information from cast shadows. *Proceedings of the 2009 IEEE International Conference on Robotics and Automation*, 220–225. (29)

SCHLIEDER, C. (1995). Reasoning about ordering. In A.U. Frank & W. Kuhn, eds., *Proceedings of the 3rd International Conference on Spatial Information Theory (COSIT'95)*, 341–349. (27)

SELLIS, T.K., ROUSSOPOULOS, N. & FALOUTSOS, C. (1987). The R+tree: A dynamic index for multi-dimensional objects. In *Proceedings of the 13th International Conference on Very Large Data Bases*, VLDB '87, 507–518, Morgan Kaufmann Publishers Inc. (40)

SHARMA, J. (1996). *Integrated spatial reasoning in geographic information systems: Combining topology and direction*. Ph.D. thesis, University of Maine. (22, 31, 95, 141)

SKIADOPOULOS, S. & KOUBARAKIS, M. (2004). Composing cardinal direction relations. *Artificial Intelligence*, **152**(2), 143–171. (25, 28, 129, 130)

SMITH, B. (1995). On drawing lines on a map. *Spatial Information Theory: A Theoretical Basis for GIS*, 475–484. (33)

STEVENS, S.S. (1946). On the theory of scales of measurement. *Science*, **103**(2684), 677–680. (23)

TARJAN, R. (1971). Depth-first search and linear graph algorithms. In *Proceedings of 12th Annual Symposium on Switching and Automata Theory*, 114 –121. (88)

TARQUINI, F., DE FELICE, G., FOGLIARONI, P. & CLEMENTINI, E. (2007). A qualitative model for visibility relations. In J. Hertzberg, M. Beetz & R. Englert, eds., *KI 2007: Advances in Artificial Intelligence*, vol. 4667 of *Lecture Notes in Computer Science*, 510–513, Springer-Verlag. (29)

TASSONI, S., FOLIARONI, P., BHATT, M. & DE FELICE, G. (2011). Toward a qualitative model of 3D visibility. In *25th International Workshop on Qualitative Reasoning (IJCAI 2011)*, (position paper). (25)

TVERSKY, B. & LEE, P. (1998). How space structures language. In C. Freksa, C. Habel & K. Wender, eds., *Spatial Cognition*, vol. 1404 of *Lecture Notes in Computer Science*, 157–175, Springer-Verlag. (18)

Tversky, B. & Lee, P.U. (1999). Pictorial and verbal tools for conveying routes. In *Proceedings of the International Conference on Spatial Information Theory (COSIT'99)*, Lecture Notes in Computer Science, 51–64, Springer-Verlag. (18)

Ullmann, J. (1976). An algorithm for subgraph isomorphism. *Journal of the ACM (JACM)*, **23**(1), 31–42. (53, 138)

Wallgrün, J.O., Frommberger, L., Wolter, D., Dylla, F. & Freksa, C. (2007). Qualitative spatial representation and reasoning in the SparQ-toolbox. *Spatial Cognition V Reasoning, Action, Interaction*, 39–58. (20)

Wallgrün, J.O., Wolter, D. & Richter, K.F. (2010). Qualitative matching of spatial information. In *Proceedings of the 18th SIGSPATIAL International Conference on Advances in Geographic Information Systems*, 300–309, ACM. (53, 143)

Wang, M. & Hao, Z. (2010). Reasoning about the inverse of cardinal direction relation. *Information Technology Journal*, **9**(9), 935–941. (129, 130)

Worboys, M. & Duckham, M. (2004). gis*: A computing perspective (2nd edition)*. CRC Press, Inc. (33)

Yao, X. (1999). Qualitative geo-referencing for web-based gis. In B. Li, ed., *Geoinformatics and Socioinformatics*, vol. 19, 1–15. (2)

Yao, X. & Thill, J.C. (2006). Spatial queries with qualitative locations in spatial information systems. *Computers, Environment and Urban Systems*, **30**(4), 485–502. (29)

Zimmermann, K. (1995). Measuring without measures: The delta calculus. In W.K. A Frank, ed., *Spatial Information Theory: a theoretical basis for GIS*, vol. 988 of *Lecture Notes in Computer Science*, 59–68, Springer-Verlag. (29)

Zloof, M.M. (1977). Query-By-Example: A data base language. *IBM systems Journal*, **16**(4), 324–343. (40)