MASTERS PROGRAM IN

GEOSPATIAL TECHNOLOGIES

A QUALITATIVE REASONING APPROACH FOR IMPROVING QUERY RESULTS FOR SKETCH-BASED QUERIES BY TOPOLOGICAL ANALYSIS OF SPATIAL AGGREGATION

Malumbo Chaka Chipofya

Dissertation submitted in partial fulfilment of the requirements
for the Degree of *Master of Science in Geospatial Technologies*

UNIVERSIDADE NOVA DE LISBOA
UNIVERSITAT JAUME·I
WESTFÄLISCHE WILHELMS-UNIVERSITÄT MÜNSTER

A QUALITATIVE REASONING APPROACH FOR IMPROVING QUERY RESULTS FOR SKETCH-BASED QUERIES BY TOPOLOGICAL ANALYSIS OF SPATIAL AGGREGATION

Dissertation supervised

by

PhD Prof. Joaquin Huerta

PhD Prof. Angela Schwering

PhD Prof. Marco Painho

February 2010

ACKNOWLEDGEMENTS

I would like to thank the Erasmus Mundus program for making it possible for me to pursue these studies through their prestigious grants. I would also like to thank all the professors and other staff working on the Master of Science in Geospatial Technologies for making my stay an enriching and enjoyable one. Thanks to my three supervisors for their comments and support. A special note of thanks goes to Professor Angela Schwering for her enormous help and guidance during the period of my research. Thanks to my colleagues on the master program for their friendship and support, especially my house mates who put up with unlimited questions and complaints over the past six months. Last but most important, I acknowledge the support and understanding of my wife and two children during this period that I have been away from home.

A QUALITATIVE REASONING APPROACH FOR IMPROVING QUERY RESULTS FOR SKETCH-BASED QUERIES BY TOPOLOGICAL ANALYSIS OF SPATIAL AGGREGATION

ABSTRACT

Sketch-based spatial query systems provide an intuitive method of user interaction for spatial databases. These systems must be capable of interpreting user sketches in a way that matches the information that the user intended to provide. One challenge that must be overcome is that humans always simplify the environments they have experienced and this is reflected in the sketches they draw. One such simplification is manifested as aggregation or combination of spatial objects into conceptually or spatially related groups.

In this thesis I develop a system that uses reasoning tools of the RCC-8 to evaluate sketch-based queries and provide a method for minimizing the effects of aggregation by determining whether a solution to a query can be expanded if some groups of regions are assumed to be parts of a larger aggregate region. If such a group of regions is found, then this group must be included in the solution. The solution is approximate because the approach taken only verifies that assumed parts of an aggregate are not inconsistent with the configuration of the whole solution. Only cases where the size of the solution equals the size of the query minus one are analysed.

It is observed that correctly identifying aggregated regions leads to solutions that are more similar to the original query sketch when the size of every other solution is smaller than the size of the query or when a lower limit is placed on the acceptable size of a solution because the new, expanded or refined solution becomes more complete with respect to the sketch of the query.

KEYWORDS

Region Connection Calculus

Spatial Aggregation

Spatial Reasoning

Spatial Queries

Spatial Query By Sketch

ACRONYMS

GIS – Geographic Information Systems

QSR – Qualitative Spatial Reasoning

QSKR – Qualitative Spatial Knowledge Represantion

RCC – Region Connection Calculus

SQBS – Spatial Query By Sketch

INDEX OF CONTENTS

INDEX OF TABLES

INDEX OF FIGURES

1. Introduction

1.1. Background and Motivation

1.1.1. Qualitative Spatial Reasoning and Sketch Mapping Applications

On going research efforts in Qualitative Spatial Knowledge Representation (QSKR) and Reasoning (QSR) since the mid 20th century have led to many innovations in Geographic Information Science (GI Science), e.g. spatial query evaluation techniques (e.g. Egenhofer 1997, and Bennett, Isli, and Cohn 1998). Interesting applications such as nuSketch BattleSpace (Forbus 2003) have been built to use combinations of quantitative and qualitative data applying spatial reasoning for some tasks. Such applications provide a basis for research into more general purely qualitative GI applications in areas such as Volunteered Geographic Information and Environmental Modelling.

Many users of geospatial applications may find it easier to work with spatial configurations of entities in the area of interest, using relative metrics as opposed to quantitative details about, for example absolute size, orientation, and location (Egenhofer and Mark, 1995). Tools to support users in this way have been researched as indicated above. These tools are designed primarily for querying spatial datasets in a spatial database with formal expressions of spatial relations specified in a query. The spatial relations in this case have to be computed from the geometric data of the spatial objects stored in the database.

An alternative approach would be to store qualitative representations of spatial datasets in the database, and retrieve the appropriate spatial objects and their relations when required. This would be similar to the approach is presented by Bennett, Isli, and Cohn (1998) where topological relations are pre-computed and stored in the database, and then they are used to evaluate queries. A qualitative model of this kind can allow the storage of spatial data supplied by users in the form of spatial descriptions. Qualitative analyses on this data could include process modelling/simulations (e.g. where people describe a physical process) as in (Forbus 2003), modelling of small spaces (e.g. a small scale farmer's partitioning of his/her field), way finding, etc.

One application of such a spatial data model could be collaborative map creation based on sketches and spatial descriptions of places on Earth. With the advent of Geographic Information Systems, their subsequent increased popularity, and the rapid proliferation of the Internet and its associated technologies, the ability to capture, document, and publish geospatial data has been made available to individual citizens at reasonable cost. Evidence of available opportunities for small players to participate in the Geographic Information market includes web-based services such as Google Maps, Google Earth, Yahoo! Maps, and similar services. '*Sites such as Wikimapia and OpenStreetMaps are empowering citizens to create a global patchwork of geographic information, while Google Earth and other virtual globes are encouraging volunteers to develop interesting applications using their own data*' (Goodchild 2007 p. 212). The significance of these Volunteered Geographic Information (VGI) services cannot be overemphasised. According to Goodchild, by 2007 Wikimapia had 4.8 million entries compared to Wikipedia's 7 million while roughly 2.8 million photographs were being contributed each month to the Flickr website. An intuitive method for contributing and finding spatial information that is close to the way perceive space, such as sketching, may increase the amount of collaboration and data shared through VGI services.

1.1.2. Motivation and Problem Overview

The above motivations for sketch-based map creation and sharing services not with standing, there are some challenges that have to be resolved even within current models that apply QSR on sketch maps for processing database queries. One major challenge is that most formal theories and models focus only on one aspect of space and their combination is not always easy (Liu, Li, and Renz 2009). Another main problem is that sketch maps are always more abstract than the reality they represent and may contain errors in any of several aspects of the spatial representation (e.g. topology, angle, or shape). Tversky (2002) specifically notes that "it is not trivial to say that people can extract from sketches what sketchers intended". In some cases, however, systematic errors can be dealt with in systematic ways as proposed by Wang (2009) for angles, curvature and direction, and by Egenhofer and Shariff (1998) for topological relations. Systematic errors in sketch maps may be manifested in several ways including, but not limited to, regularization of angles to right angels, straightening of road curves, exaggeration of size due to the relative significance of depicted features, and hierarchical organization of geographic features (Tversky 2002). Most of these are partly aimed at simplifying the representation in order to minimize the amount of memory and processing required for interpreting the information (Tversky 2003).

A particular form of simplification that may arise during the drawing of a sketch is generalization of information that leads to grouping of features that are conceptually similar or more spatially related. This type of generalization is called aggregation. When a query to a spatial database contains objects (aggregates) that represent aggregated groups of objects, valid solutions to the query may be excluded from the query results because the aggregate object does not match any objects in the database. In such a case, there needs to be a way of recognising and testing when aggregation may cause some results to be rejected, and then to find the objects in the database that correspond to the parts of the aggregate object of the query.

In this thesis, a formal model for topology, namely the RCC-8, is used as the basis for a database model for storing topological information. We use this model to develop a method for refining the solutions of a sketch-based query based on the assumption that some solutions to the query have been excluded as a result of aggregation. The queries in this work are performed against a database containing topological information extracted from other sketches.

1.2. Research Problem and Objectives

The main objective of this study is to develop a method for refining the solutions of a sketch-based query by searching for groups of objects in the database that together approximately match an object in the query for which no matches were found. The database model will not store any geometric information for the sketches to be analysed. The main objective is achieved by pursuing the following sub-objectives:

i. Develop a database model for topological relations between region objects in a sketch map. The implementation must have query processing algorithms based on the spatial query by sketch (SQBS) paradigm (Egenhofer 1997).
ii. Extend the initial model with a method for discovering a group of regions in the database that have been combined into a single whole in the query. The method sought must be as simple as possible but consistent.

1.3. Outline of the Thesis

Chapter 2 presents a review of the literature of the main background concepts used in the study. A brief background to qualitative spatial reasoning is given, details of the RCC model and spatial query by sketch methods are discussed, and a brief review of related work on spatial aggregation is given. The test model for the study is presented in Chapter 3 and the extension in Chapter 4. Chapter 5 discusses the results of the development of the model and experiments with selected sketch maps, and outlines some of the challenges met in the process. Finally, the overall results of the work are concluded in Chapter 6.

2. Literature Review

2.1. Formalisation of Space

There are many ways to model spatial data qualitatively. The specific elements of any model depend on the desired (or available) level of detail and the properties of the space being modelled, among other things. Surveys in QSKR and QSR have described formal approaches in terms of their theoretical foundations, the spatial aspects being formalised, and the complexity of reasoning over the representations of each approach (Cohn and Renz 2007; Renz and Nebel 2007). For example, logic based approaches such as RCC are distinguished from algebraically motivated ones such as the 9-intersection model, while different models can also be distinguished by the dimension of the entities that are considered as primitive spatial entities of their domains.

Aspects of space can be distinguished in terms of the mathematical theories used to model them. Papadias and Sellis (1993) note that (according to Buisson (1989)):

"*… the spaces of interest in spatial reasoning are topological spaces which include only concepts of connectedness and continuity, vector spaces which deal with vectorial dimensions and directions, metric spaces which deal with the concept of distance and Euclidean spaces which admit notions of scalar products, orthogonality, angle and norm.*" (Papadias and Sellis 1993, p.2)

In particular, topology, direction and orientation, distance, size, and shape have been investigated to some depth in QSR research and a brief discussion of approaches that have been studied for each of these aspects is give by Cohn and Renz (2008), Renz and Nebel (2007), as well as Bennett (2008).

On topology, the two most popular models are the RCC-8 and the 9-intersection. RCC-8 is a model of the Region Connection Calculus originally presented in (Randell, Cui and Cohn, 1994). The 9-intersection model is conceptually similar to the RCC-8 in terms of the topological distinctions made between different configurations of regions in space, but is based on the algebra of closed and open point-set topologies (Egenhofer and Franzosa, 1991). Therefore, a point is a primitive spatial entity in the 9-intersection model which also allows both, lines, and regions as spatial entities in its domain.

Direction and orientation calculi are more difficult to deal with in 2-dimensional space because they require a frame of reference and an order (clockwise, left-right) in which relations between objects are referred to. The double cross calculus provides a mechanism for computing relative position based on an extrinsic frame of reference. This calculus is more suited to points than extended spatial objects such as lines and regions. Star calculi are too numerical that it is possible to define a coordinate reference system on them (Cohn and Renz 2007). This is not well suited to sketch map aggregation (combination) because sketch data is defined at different levels of granularity and abstraction. Calculi with an extrinsic frame of reference such as ones that use cardinal directions (N, S, E, W) are also well suited to reasoning with points. A final category for consideration, are approaches designed to work mainly for 2-dimensional regions. The first uses Allen's interval algebra, treating the whole frame of reference as a combination of two orthogonal 1-dimensional reference frames. Each relation then becomes a pair of two interval relations. The other one is the direction-relation matrix (also referred to as the Cardinal Direction Calculus - CDC) in which directions are decided based on 9 sectors formed by the minimal bounding axes of a region (Goyal and Egenhofer 2001). Because the shape of a region determines its minimum bounding rectangle

this affects the direction relations that can be derived between any two regions and sometimes makes it difficult to derive definite direction information using a calculus such as CDC (Egenhofer 1997). While this is the case, direction relations are an important component of qualitative spatial representations used in spatial query evaluation. Section 2.3 below describes how direction relations are used in SQBS and in section 5.4 direction relations are discussed with respect to the number results returned in a query.

Reasoning in QSR is achieved using a variety of approaches of which the most popular are constraint based techniques (Renz and Nebel 2007). These approaches are based on the fact that the relationships between objects can be given in the form of constraints. A constraint over a set of variables V consists of a relation R and an n-tuple of variables from V. The constraint is said to be satisfiable if there exists an instantiation of the variables that is also a member of the relation R. Such an instantiation is said to satisfy the constraint R. A spatial reasoning problem can then be formalised as a constraint satisfaction problem (CSP) which consists of the set V, and a set of constraints over V. The desired solution is an instantiation of the variables such that all constraints are satisfied. CSPs are used for verifying the consistency of sets of relations and for comparing different constraint sets as graphs. A formal discussion on formulating and solving constraint satisfaction problems with unary and binary constraints is given in Kumar (1992). A note worthy point is that while binary constraints form edges between variables in the graph of a CSP, unary constraints are viewed as edges from variables to themselves (a loop on the same graph node). However, for simple applications such as the one presented in section 2.3, unary constraints may be excluded from the graph of the CSP because they would first be used to identify possible instances of the variables. This way the process is broken into two parts, the first being elimination of those instantiations of the variables that are not consistent with the unary constraints and then the evaluation of binary constraints.

2.2. RCC -8

The RCC theory is built on the concept of connectedness defined using one primitive dyadic relation, C, that determines for any two regions x and y whether the regions are connected. The theoretical foundations of the broader theory are rooted in earlier work by Clarke (1981, 1985) – see Randell Cui, and Cohn (1994). In broad terms $C(x, y)$, read 'x connects with y', is true if and only if the topological closures of x and y share a common point. In the original publication of the theory, no distinction is made between whether a set is considered as closed, open, or both. The theory provides two additional axioms which make it possible to define a basic set of binary relations. The axioms state that C is reflexive and symmetric:

$\forall x C(x, x)$ …. x is connected with itself (reflexivity)

$\forall xy[C(x, y) \rightarrow C(y, x)]$ …. If x connects with y then y connects with x (symmetry)

The whole set of basic relations derived from C can be embedded in a relational lattice with elements ordered in such a way that every higher relation subsumes all lower relations with which it is connected (Figure 1). This means that every relation higher in the lattice implies a disjunction of all lower relations connected to it while a lower relation implies a conjunction of all higher relations connected to it. The following basic relations were defined and presented in the original RCC paper of 1994:

- $\forall xy[DC(x, y) \leftrightarrow \neg C(x, y)]$ …. 'x is disconnected from y',

- $\forall xy\left[P(x,y) \leftrightarrow \forall z[C(z,x) \rightarrow C(z,y)]\right]$ … 'x is a part of y',
- $\forall xy\left[PP(x,y) \leftrightarrow [P(x,y) \wedge \neg P(y,x)]\right]$ … 'x is a proper part of y',
- $\forall xy\left[EQ(x,y) \leftrightarrow [P(x,y) \wedge P(y,x)]\right]$ … *'x is identical with y',*
- $\forall xy\left[O(x,y) \leftrightarrow \exists z[P(z,x) \wedge P(z,y)]\right]$ … *'x overlaps y',*
- $\forall xy\left[PO(x,y) \leftrightarrow [O(x,y) \wedge \neg P(x,y) \wedge \neg P(y,x)]\right]$ … *'x partially overlaps y',*
- $\forall xy\left[DR(x,y) \leftrightarrow [\neg O(x,y)]\right]$ … *'x is discreet from y',*
- $\forall xy\left[EC(x,y) \leftrightarrow [C(x,y) \wedge \neg O(x,y)]\right]$ … *'x is externally connected with y',*
- $\forall xy\left[TPP(x,y) \leftrightarrow \left[PP(x,y) \wedge \exists z[EC(z,x) \wedge EC(z,y)]\right]\right]$ … *'x* is a tangential proper part of y',
- $\forall xy\left[NTPP(x,y) \leftrightarrow \left[PP(x,y) \wedge \neg\exists z[EC(z,x) \wedge EC(z,y)]\right]\right]$ … *'x* is a non-tangential proper part of y'.

The relations P, PP, TPP, NTPP have inverse relations denoted PI, PPI, TPPI, and NTPPI respectively which means that these relations are non-symmetrical, while the remainder are symmetrical. The inverse of a relation is defined as the converse truth of the relation, so $\forall xy[PPI(x,y) \leftrightarrow PP(y,x)]$.

RCC-8 consists of subset of eight of these basic relations that are JEPD. In Figure 1 these are the ones closest to the lower bound of the lattice. Beyond the original interpretation of the primitive relation C, Bennett (2000) has given alternative interpretations with respect to different mathematical theories, notably the interpretations in point set topology that distinguish between open and closed sets. In the closed interpretation a region is identified with a regular closed set of points. Two regions are connected if they share at least one point and they overlap if their interiors share at least one point. In the open set interpretation regions are connected if their closures share at least one point, and they overlap if they share at least one point (Bennett, 2000). Both interpretations are consistent with propositions defining the relationships between interiors, boundaries, and closures of spatial regions given by Egenhofer and Franzosa (1991). This relationship between the two formalisms makes it possible to apply either model on the same definition of 2-dimensional spatial regions so that reasoning differs only in the operations employed and not in the formal specification of the input.

Some reasoning tasks with RCC-8 can be achieved using the RCC-8 composition table and the conceptual neighbourhood graph among other more complex approaches. Using the RCC-8 composition table inferences of the following nature can be made; given two relations $R_1(x,y)$ and $R_2(y,z)$, what is the set of relations that can possibly hold between x and z (Cohn, Bennett, Gooday, and Gotts 1997). Using the RCC-8 composition table, a set of RCC-8 relations among a set of objects can be tested for consistency by ensuring that relations among every 3 objects in the set are consistent (this is known as path-consistency which is contrasted from consistency per se, which requires all relations to be consistent with each other simultaneously). This is done by formalising the spatial scene as a CSP where the binary relations are the RCC-8 relations among objects in the scene.

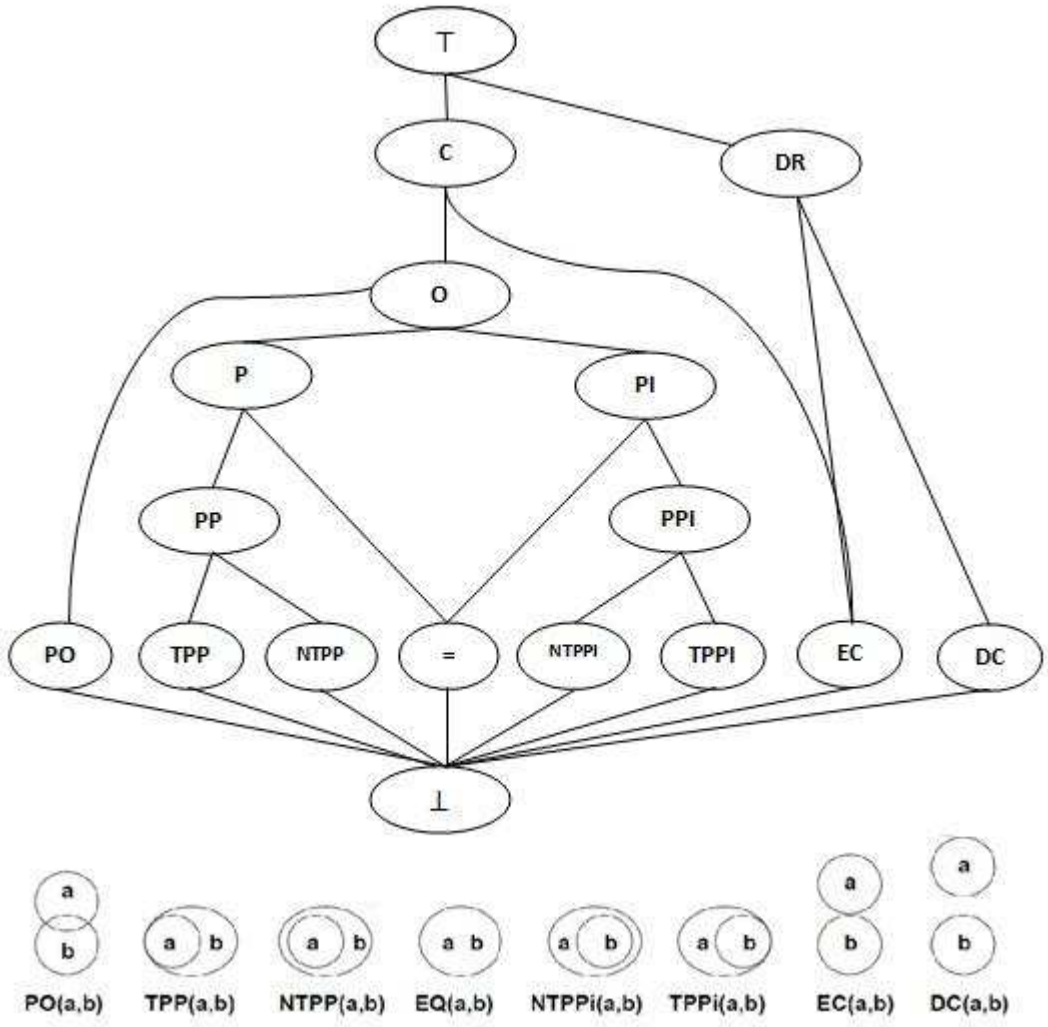


Figure 1: Lattice of the subsumption hierarchy of the basic binary RCC relations (reproduced from Randell, Cui, Cohn 1994).

2.3. Spatial Query by Sketch and Spatial Scene Queries

SQBS is a method for performing query operations in spatial databases based on a sketched representation of the desired spatial configuration. According to Egenhofer (1997), traditional query languages are tedious to use and their strict syntax and grammars together with the inherent nature of geographic information (that it is often vague, imprecise, or not standardized) impose a limitation on their usability. He argues that the fact that verbal spatial descriptions are usually ambiguous is a fundamental problem that may lead to misinterpretations.

In SQBS a sketch map drawn by a user is decomposed into individual objects represented in the sketch. The spatial and non-spatial properties of the individual objects, and the spatial relations among the objects form part of the formal representation of the spatial query. Five types of spatial information are used in SQBS: coarse topological relations, detailed topological relations, metrical refinements, coarse cardinal directions, and detailed cardinal directions. Coarse topological relations are relations of the 9-intersection model (Egenhofer and Herring 1990). Detailed topological relations consider details of the interaction between the boundaries of two non-disjoint spatial entities. Metrical refinements are measures used to quantify topological relations at the level of

interior and boundary interactions of spatial entities. Cardinal directions are exploited using CDC at two levels. The coarse cardinal directions provide a means for determining the relative direction of one object to another broadly. For example object u could be north (N) and north-west (NW) of object v. Detailed cardinal directions, on the other hand, say how much north or north-west the object is by calculating the proportion of the objects area (or length for lines) falls into each sector formed by the minimal bounding axes of the referent object.

In contrast with other visual query systems, which require the user to draw close approximations of the desired spatial configurations, SQBS primarily retrieves data based on coarse topological relations and eliminates undesirable results using the four other types of information described above. If no results remain at the end of the query processing, the query may be relaxed by substituting a relation from the query with the union of the relation with its conceptual neighbours (Egenhofer 1997).

A SQBS query is a form of spatial scene query where the query processor attempts to find a spatial configuration that is similar to that expressed in the query. The central question in spatial scene queries is how to establish the associations between the elements of one scene and the elements of another scene (Nedas and Egenhofer, 2008). A spatial scene query comprises a set of spatial objects and relations between the objects. A query is formulated as a spatial CSP. For each spatial object, the properties of the object such as its feature class, shape or size become unary constraints for the query, and the binary relations between the objects become binary constraints of the query.

The evaluation of the query then involves finding configurations in the database that satisfy all the constraints of the query. This is achieved by constructing an association graph which consists of a set pairs of variables (objects in the query) and database objects. The set of pairs are the nodes of the association graph, while the set of combined constraints become the edges of the graph. The construction of the association graph of the CSP involves first resolving the unary constraints by matching query variables to regions. For each variable v_i in the query, add a node $(v_i,\ u_j)$ to the association graph if object u_j in the database satisfies all the unary constraint of v_i. An edge is added between every pair of nodes $(v_i,\ u_j)$ and $(v_k,\ u_l)$ of the association graph if the binary constraints between objects u_j and u_l satisfy the binary constraints between variables v_i and v_k of the query. The final solutions to the query comprise all maximal complete subgraphs (maximal cliques) of the association graphs. Several algorithms for extracting maximal and maximum cliques of a graph have been developed (e.g. Bron and Kerbosch 1973, Tomita et al 2006, Koch 2001).

Solutions obtained from this type query evaluation are not always exact. Three types of solutions are distinguished, namely, complete solutions that are realised from maximum cliques of the association graph when all variables are part of the association graph, incomplete solutions that correspond to maximal cliques with only a subset of the query variables included, and the empty solution when no clique was found in the association graph.

Because there are many possible solutions (many possible association graphs and cliques per graph), to prioritise the results of a query, measures of similarity between the query scene and the database scene have been proposed that take into account three components:

i. An object similarity component measures the similarity between objects in the query scene and those in the database scene.

ii. A relation similarity component measures the similarity between the binary relations among objects in the query scene and those in the database scene.

iii. A scene completeness component that measures the similarity of two spatial scenes with respect to completeness (i.e. based on number of objects in the query scene, number of objects in the database scene, number of objects matched – or not matched).

The development of the proposed methods for scene similarity assessment was motivated by three psychological insights. First, that people start scene comparisons by locating possible object-matches, that they associate objects so that relations also correspond, and that with a gradual decrease in similarity, very different scenes become irrelevant (Nedas and Egenhofer 2008). A minimum value is imposed on each similarity component which restricts solutions included in the final results that are returned from the query. In this case we want to ensure that all spatial configurations in the database that are potential solutions to the query can be found. If there is an aggregated region in the query sketch then it will reduce the similarity measures of all three components. The present study attempts to reduce this effect by finding database regions that may correspond to an aggregated region in the query sketch.

2.4. Generalisation and Spatial Aggregation

One problem that has not been tackled by the SQBS and spatial scene query models discussed above is the tendency for people to generalise and schematise perceived environments (Tversky 1993, 2002) and how these generalisations and schematizations are manifested in the sketch maps that people draw. Wang (2009) has made proposals towards alleviating the effects of human schematization and systematic errors in sketch map formalisation. The methods proposed complement the detailed cardinal directions used in SQBS by considering among other things angles and curvature of objects in route maps.

Generalisation is a basic human activity in which unimportant specific aspects of reality are discarded and focus is given to the important ones. The concrete reality can be conversely viewed as a combination of the general and specific aspects that were separated during the generalisation. As a mental process, generalisation involves, among other functions, distinction, combination, and abstraction (Brassel and Weibel 1988) of details from the reality. For sketch maps, the structure captured by the sketches is the structure of the information being conveyed as opposed to the structure of the represented environment. Also, when sketching regions, people impose a hierarchical structure on the features depicted in the sketches emphasizing those with a larger environmental scale (Tversky 2002). From this it may be construed that the process of spatial generalisation as human activity is also impacted by the hierarchical organisation of information internalised by an individual or by at least by the process that generates this hierarchical representation.

It is therefore necessary that a model for sketch-based query should support the generalisation of spatial features. One method of spatial generalisation is spatial aggregation, in which process similar regions of space are combined to make one region based on certain attributes common to all regions affected. In a GIS application aggregation can be done on regions of the same feature type based on a real valued function (e.g. in Indulska and Orlowska, 2002). For sketch maps, however, where feature attributes may not be homogeneous and the accuracy of their values generally poor,

there is need for a non-numerical model for deciding how regions should be aggregated as spatial entities based on non-homogeneous but possibly related criteria.

From a topological perspective, the aggregation of spatial regions in a spatial scene presents some problems with respect to the consistency of the configuration obtained. This problem has been investigated by Tryfona and Egenhofer (1997) but no complete solution for the purely qualitative case has been found in the literature. Based on the 4-intersection model, a variation of the earlier cited 9-intersection model, this result is derived from an analysis of the interaction of the interiors and boundaries of the aggregate region and a third region on the one hand, and between the parts and the third region on the other.

The results reported by Tryfona and Egenhofer show that for an aggregate region with two parts, there are three groups of possible spatial configurations that require increasingly more information in order to determine their consistency. Their study distinguishes aggregate regions with connected parts (contiguous regions) and aggregate regions with disconnected parts. Aggregate regions with connected parts were analysed and of the 64 possible relations between an aggregate with two parts and another region, only 27 were found to be consistent. The second case was analysed based on the first case and a total of 31 relations were found to be consistent. In this study we assume that aggregate regions are one piece (with connected parts) so the result will be comparable only to the first case.

3. Test Model Design

The model was implemented in the Postgresql Database Management System. All functionality is written in the built-in procedural language PL/PGSQL which presented a few challenges as it is limited in terms of data structures as well as being a purely procedural language without support for session (global) variables. However, the implementation was sufficient for testing the query system with a variety of input queries on different topological configurations extracted from sketch maps. Objects in the sketches, which are interpreted as 2-dimensional regions in the plane, are represented by identifier labels or 'region names'. Region names are stored as text fields of variable size no more than 48 characters in length. All operations on data in the model are thus performed using references to the region names.

3.1. Database Design

Internally, each region name is associated with a unique region id (`reg_id`). User tables can be created and then associated with the internal structure by declaring one of their columns as a column of region names. This is achieved by invoking the function, `set_rcc_table(table_name, column_name)`, which alters the column to a 48-character variable length text field with unique values and sets a trigger to execute before each one of the three row-level operations insert, update, and delete. `set_rcc_table` also changes the name of the column to `the_rcc_region`. Once the link has been established, a user can manipulate his/her user tables using Postgresql's standard SQL statements and other functions. All changes to values in the column of region names will be reflected in the internal tables of the model. Other functionality of the model is accessible through several user interface functions discussed in the following sections. Figure 2 shows an overview of the overall system design for the implementation.

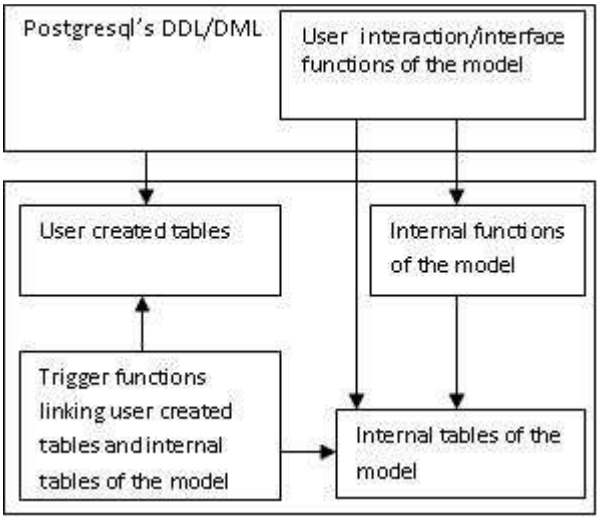


Figure 2: Overview of the system design showing the main elements of the system and their interactions

The database model has three basic tables (Figure 3). The first table called `rcc_spatial_regions` relates the unique region ids with user given region names and tables. From a user's perspective, this allows the design of the database to be independent of the how binary topological constraints are represented internally since the database design is decoupled from internal model design. Decoupling of the two components is also important for easy maintenance of the software because

later changes made to the internal program code did not require changing the user interaction methods or the functions that implement them.

The table `c_clust` keeps track of all regions that are altogether contiguous. Each record of `c_clust` consists of a cluster identifier number and the number of region names in the cluster. Upon insertion into the database, a region is assigned to the default cluster (`cluster_id` is 1). The default cluster is associated with all isolated regions. A region is subsequently removed from a cluster and added to another cluster when a constraint has been imposed on it with other regions in the database such that the region is no longer connected to any member of the original cluster. Two clusters are combined into one if any two regions, one from each cluster, become connected.

Finally, the table `rcc_constraint_network` stores records of basic RCC-8 constraints over the set of regions corresponding to the region names. Each constraint is stored as an ordered pair of region identifiers and a RCC 8 relation name.

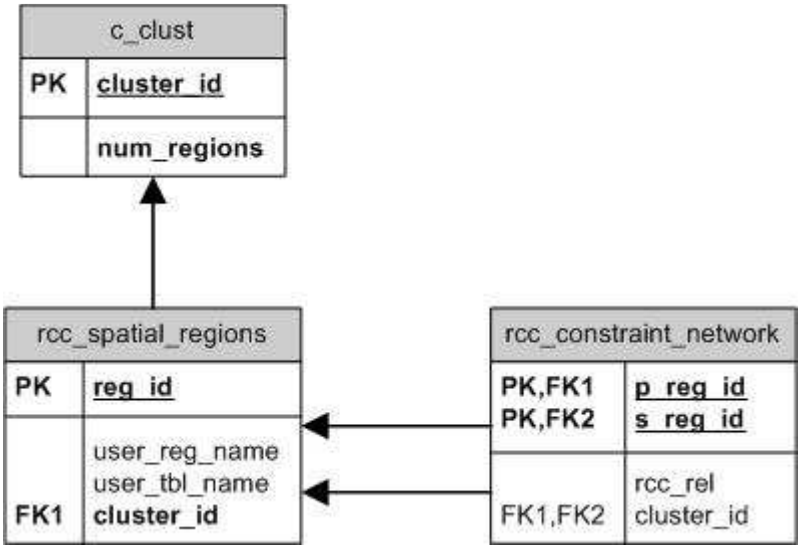


Figure 3: Model diagram for main relations

There are several other tables that support data processing functionality. The most important of these are `rcc_rel_comp`, `rcc_rel_compb8`, `rcc_u_constraints`, `rcc_b_constraints`, `rcc_qryreg_assoc`, `qry_soln_graphs`, and `qry_usr_soln`.

`rcc_rel_comp`, `rcc_rel_compb8` are the RCC-8 composition table for the eight basic relations and the composition table for all possible unions of RCC-8 relations respectively. The latter is computed from the former using the algorithm implemented by Fehling, Nebel, and Renz (1998). `rcc_u_constraints` and `rcc_b_constraints` are used to store unary and binary constraints, respectively, for each query. `rcc_qryreg_assoc` and `qry_soln_graphs` are used for storage of intermediate results during query processing, while `qry_usr_soln` is used as a template for presenting the final results.

3.2. Data Input Methods

All region names added to the database are associated with a particular sketch map (identified by the sketch's name). Data are inserted, updated, or deleted using normal relational DML operations and two functions: one used to indicate the name of a sketch map to which any subsequently inserted regions belong (`use_sketch(sketch_name)`), and the other to indicate that any

subsequently inserted regions should not be added to the currently selected sketch map. An insertion without a prior selection of a sketch map using `use_sketch(sketch_name)` will fail.

Three functions that take as arguments a binary constraint over two region names in the database enable the input and manipulation of constraints imposed on regions in a sketch. The function `related_regions(primary_region, secondary_region, relation)` adds the following constraint to the database: `primary_region { relation} secondary_region`. For example, if region X is a tangential proper part of region Y, then `related_regions(X, Y, TPP)` inserts `X {TPP} Y` into the table `rcc_constraint_network`.

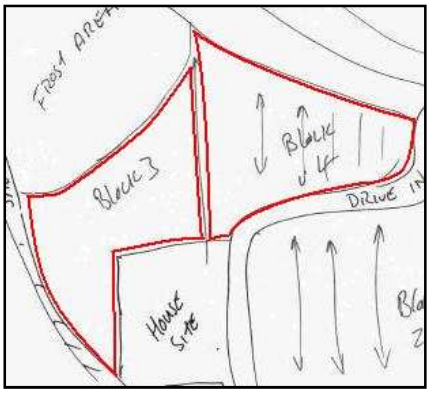


Figure 4: Exctract from a sketch map added to the database model

The function `unrelated_regions(primary_region, secondary_region)` removes the constraint imposed by `related_regions` on the given pair of regions. Since the database does not explicitly store constraints for which the corresponding relation is DC, calls to the function `related_regions` with the relation DC simply trigger a call to `unrelated_regions`. Specifically, these two functions are responsible for maintaining the cluster index by updating it whenever regions move between clusters.

```
SELECT use_sketch('KiwiFarm 1');

INSERT INTO farms(name, description, the_rcc_region)
VALUES ('Block 3', 'Farm division block 3', 'S1Block3');
INSERT INTO farm_properties(feature, name, value)
VALUES ('S1Block3', 'Category', 'spatial organisation');
INSERT INTO farm_properties(feature, name, value)
VALUES ('S1Block3', 'Feature', 'blocks');

INSERT INTO farms(name, description, the_rcc_region)
VALUES ('Block 4', 'Farm division block 4', 'S1Block4');
INSERT INTO farm_properties(feature, name, value)
VALUES ('S1Block4', 'Category', 'spatial organisation');
INSERT INTO farm_properties(feature, name, value)
VALUES ('S1Block4', 'Feature', 'blocks');

SELECT end_sketch();


SELECT related_regions(get_region_id('S1Block3', 'farms'),
get_region_id('S1Block4', 'farms'), 'EC');
```

Listing 1: Script for data input corresponding highlighted regions in Figure 4

Listing 1 above shows the SQL script that adds data about the highlighted regions in Figure 4 to the database. The first line instructs the database to associate all data about regions inserted from that point with the sketch named 'Kiwi Farm 1'. The reference to the sketch does not have to exist prior to the call, but if the name was already added to the database, then the existing reference is used. The last but one call indicates to the system that the input has finished. Any data added after this will not be associated the current sketch and if no other sketch is specified, input will fail for any value of the column `the_rcc_region`. The final line instructs the system to add a constraint (EC) between the two recently added regions. The table `farm_properties` here illustrates that other tables can be linked to a table with region names and data in those tables may be used during a query as seen in the coming sections.

The last function `update_region_constraint(primary_region, secondary_region, relation)` allows constraints to be edited in a consistent way by ensuring that no constraint is previously imposed on the given pair of regions prior to an update. Basically, an update is a sequence of two function calls, first to `unrelated_regions` and then `related_regions`.

3.3. Consistency and Path-consistency

Consistency checking is used to maintain the database in a consistent state every time a constraint is added or removed. Because path-consistency implies consistency in a basic RCC constraint network, we only apply a path-consistency algorithm when testing for consistency. Path-consistency also plays the role of ensuring that a set of constraints is complete by adding to the set the inverses of all constraints in the set and for every constrained object the constraint that the object equals itself.

In the implementation of the model, path-consistency is checked using a variation of Van Beek's path-consistency algorithm for CSPs (van Beek 1992). The algorithm used was proposed by (Stocker and Sirin 2009) and uses the recursive procedure $ADD(R_{ij}, Q)$ to verify the path-consistency at the node (i, j) of the CSP but allows for the empty and the universal relations to be ignored.

The path consistency algorithm takes a set of RCC-8 constraints over a set of variable say, V and for every pair of variables i and j in V it ensures that the relation R_{ij} is a subset of the composition of relations R_{ik} and R_{kj} for all k in V. Each time a constraint is checked, the converse constraint is also checked ensuring that the composition is also consistent with the inverse relation. Initially a constraint is placed in a queue. For each composition R_{ik} o R_{kj} the constraint for R_{ij} is compared with composition and if need be it is revised taking the intersection of the sets of relations of the two constraints. R_{ij} is placed back in the queue and reprocessed later until a fixed point at which it does not need to be refined to be consistent with any of the constraints derived from composition has been reached. The result is a set of binary constraints equivalent to the original set or the empty set which is indicated by a NULL value in the implementation. The result is an equivalent of the original in the sense that every triple of constraints that are consistent in the original set are also consistent in the returned set.

Function:	PATHCONSISTENCY(Θ)
Input:	Set Θ of binary RCC-8 constraints over a set of variables V.
Local:	Set G of binary RCC-8 constraints over V.
Output:	Path-consistent set equivalent to Θ, or NULL if inconsistency is detected.

```
1.    if Θ = Ø then
2.        return Θ
3.    end if
4.    G ← complete(Θ)
5.    if G ≠ Ø then
6.        Q ← {R_ij | i, j ∈ V and Ø ≠ R_ij ∈ G}
7.        while Q ≠ Ø do
8.            G ← ISCONSISTENT(G, Q, POP(Q))
9.            if G = Ø then
10.               Q ← Ø
11.           end if
12.       end while
13.   end if
14.   return G
```

Function:	ISCONSISTENCT(G, Q, R_{ab})
Input:	Set G of binary RCC-8 constraints over a set of variables V, FIFO set Q of binary RCC-8 constraints over V, Binary constraint R_{ab} on variables a, b $\in$ V.
Output:	Set equivalent to G that is path-consistent at R_{ab}, or NULL if inconsistency is detected.

```
1.    for each R_bc ∈ G, c ≠ Ø, a, b do
2.        G ← RELADD(G, Q, R_ab o R_bc)
3.        if G = Ø then
4.            return G
5.        end if
6.    end for
7.    return G
```

Function:	RELADD (G, Q, T_{ac})
Input:	A set G of binary RCC-8 constraints over a set of variables V, FIFO set Q of binary RCC-8 constraints over V, Binary constraint T_{ac} on variables a, c ϵ V derived from the composition over (a, b) and (b, c).
Local:	U_{ac} ,V_{ac} orginal and refined constraint on pair (a, c).
Output:	Path-consistent set equivalent to G, or NULL if inconsistency is detected.

```
1.    if T = Ø then
2.        Return G
3.    end if
4.    U_ac ← {R_ij | i = a, j = c, R_ij ϵ G}
5.    if U_ac = Ø then
6.        V_ac ← T_ac
7.    Else
8.        V_ac ← T_ac ∩ U_ac
9.        if V_ac = Ø then
10.           G ← Ø
11.           return G
12.       end if
13.       if U_ac = V_ac then
14.           return G
15.       end if
16.       G ← G - {U_ac}
17.   end if
18.   G ← G U {V_ac}
19.   Q ← Q U {V_ac}
20.   RELADD (G, Q, inverse(V_ac))
21.   return G
```

Algorithm 1: PATH-CONSISTENCY algorithm (from Stocker and Sirin, 2009)

3.4. Database Query

Queries on the data are formulated in two ways. A simple query may be an SQL select query to retrieve the set of region pairs that are constrained by a certain relation or to view the relation that is constraining a pair of regions. In addition, a simple query may be combined with other SQL queries to produce more complex but perhaps more useful results.

The second type of query is based on the query by sketch paradigm. As in Nedas and Egenhofer (2008), a sketch-based query is composed of two parts: A set of unary constraints on each variable from a set of variables V, and a set of binary constraints on members of the Cartesian product $V \times V$. Such a query may have several solutions and a solution to the query may be complete, incomplete, or empty.

3.4.1. Relational Queries

Relational database queries can be used to answer to basic questions with respect to the sketch maps topology: Which regions are constrained by a given relation? And, which relation constrains the given pair of regions? In addition, any other queries maybe performed on both the internal and user defined database tables.

3.4.2. Sketch-based Queries

The sketch-based query is modelled after Nedas and Egenhofer's spatial query with a few differences. According to their paper a spatial-scene query "has two major components: objects and relations among the objects" (Nedas and Egenhofer 2008). The analysis and evaluation of the query involves matching properties of objects expressed in the query and their binary constraints with those of objects in the database. These correspond to unary and binary constraints for the query respectively.

Each sketch that will be used to query the database must first be formally analysed to extract the topological relations among the objects depicted in the sketch. A manual process for achieving this is used in this study.

3.4.2.1. Sketch-based Query Presentation

Queries to the database are constructed by a series of calls to two functions, one for setting unary constraints and another for setting binary constraints for the query. Unary constraints are given as any SQL statement returning sets of region names.

The function `unary_qry_constraint()` must be called once for each object (variable) in the query sketch. Any variable not passed explicitly to the query system will not be processed. The function `unary_qry_constraint()` takes three arguments, namely, a variable name, the table from which regions must be fetched, an SQL statement as indicated above. As each variable is added, it is assigned a position in natural order starting from 1 for the first variable. Apart from the requirement that the SQL query must return a set of region names, the manner in which it is constructed and/or processed by the database backend is not influenced by the design of the query processing procedures. Consequently, there is no straight forward way to isolate the individual components or attributes of the unary constraints.

The function `binary_qry_constraint(string_of_binary_constraints)` must be called exactly once before executing the query. This sets the binary constraints placed upon the query variables. Binary constraints are given as a string of the following format:

```
primary_region secondary_region relation [AND primary_region secondary_region relation […] …]
```

3.4.2.2. Sketch-based Query Processing

Once the query variables and constraints have been set, the query is executed by calling the function `rcc_eval_qry()`. This function takes a Boolean argument specifying whether constraint relaxation for binary constraints should be attempted. The main processes that occur include query validation, variable to region name matching, creation of the association graph from each set of matches, generation and storage of viable solutions from the association graph.

3.4.2.2.1. Query Validation

The process of query validation involves creating a matrix structure from the binary constraints and ensuring that the set of constraints is inverse complete, equals complete, and path-consistent. This ensures that all implicit constraints are made explicit during validation.

The function returns a matrix representing a set of path-consistent constraints, or NULL if the constraints specified are inconsistent. But as a result of the way the path-consistency function processes the CSP, if the constraint between a pair of regions could not be discovered, this particular constraint is simply made NULL as opposed to removing the concerned regions from further analysis or failing the validation. This may contrast however with the Egenhofer approach because in that case constraints are not atomic but elements of Cartesian products several constraint sets.

3.4.2.2.2. Variable Region Matching

For each variable execute the associated SQL statement to retrieve the region names that belong to tuples returned from executing the SQL statement. This results in the creation of a list of matches between variables and region names. This match-list is then used to create a set of possible solutions to the query. Each potential solution set is a subset of the Cartesian product $V \times R$ in which each variable appears at most once, and each region appears at most once. This removes unnecessary steps when checking binary constraints. The condition set by Nedas and Egenhofer, however, only stipulates that each variable must appear at most once in any given association graph since in their approach only a single association graph is created for all possible matches.

As shown in Algorithm 1 the procedure for constructing solution match sets is iterative. Starting with one variable, each variable-region match is placed in its own solution match set. Then for each set, new sets are created by including every match of the subsequently selected variable into a new solution set. This process is repeated while ensuring that a match is not added to a solution set if another match with the same region was previously added. Additionally, the algorithm used restricts the regions that can be added to be from the same sketch. This introduces a bias in the content of results that can be obtained since only sketches for regions that matched the first variable will be included in any solution set. For purposes of our experiment this wasn't a problem because the number of sketch maps was small and the types of features used was limited so that every sketch map had at least one feature of the popular feature types.

Algorithm:	MAKE-SOLUTION-SETS
Input:	Set L of variable-region pairs (v, u) s.t. v ∈ V and u ∈ D the set of region names in the database.
Local:	newLSS lists of members of L.
Output:	Set S of lists of members of L.

```
1.  S ← {Ø}
2.  for each v ∈ V do
3.      for each LSS ∈ S do
4.          for each (v, u) ∈ L do
5.              if (u is not already in LSS and LSS does NOT contain a
                    region from DIFFERENT sketch as u)
6.                  newLSS ← LSS U {(v, u)}
7.                  S ← S U {newLSS}
8.              end if
9.          end for
10.         if newLSS ≠ Ø then
11.             S ← S - {LSS}
12.         end if
13.     end for
14. end for
```

Algorithm 2: Construction of solution sets

The order in which variables are processed in the procedure determines the number and content of the output solution match sets and therefore of the final solution. This is because of the restrictions on inclusion stated above, and the possible variations in the number of regions matched to each variable and vice-versa. The algorithm mitigates this effect slightly by processing variables with the highest number of matches first. This ensures that for each variable, there is a higher possibility to find at least one solution set to which it can be included. A bad case is when all variables match exactly one region and some regions are repeated, in which case only the first encountered variable matching the repeated region is included in the solution.

3.4.2.2.3. Construction of the Association Graph

The association graph is constructed in a similar manner as Nedas and Egenhofer's although our approach is based on building smaller sets of potential solutions. Whenever a constraint is NULL, it is excluded from the final solution. So, for any pair of variable-region matches, if the corresponding constraints are NULL, then they cannot be in the same solution. The resulting association graph is represented as a matrix of the same dimension as the query constraint graph. Whenever the constraint between regions in the database did not satisfy constraint between variables matched to those regions, the association graph entry corresponding to the joint constraint between the matches is set to NULL.

3.4.2.2.4. Generating Solutions

The final query solutions are created from the solution sets using a clique enumeration procedure. The maximal clique algorithm version 2 of Bron and Kerbosch (1973) is applied on each association graph extracting cliques that are either maximal or maximum (Algorithm 3). Each clique is a solution to the original database query. The solutions are stored in the table qry_soln_graphs, each with a reference to the solution set that generated it and a unique integer to identify the solution.

The solutions of the query are however not unique because of several obvious reasons. Firstly, the construction of solution sets allows for redundancies since two solution sets may have exactly the same variable-region matchings for a proper subset of the total number of variables. This is certainly the case where there are no unary constraints on any of the variables and each variable is matched with every other region in the database. Secondly, because we enumerate all maximal cliques of every solution graph, even those solution sets that do not have exactly the same set of regions, may end up giving the same solution. The alternative approach to find only the maximum clique is not anymore effective since it may instead reject valid (incomplete) solutions with regions that more closely match the regions given in the query in favour of a poorly matching complete solution.

3.4.2.2.5. Presentation of Results

Results are obtained solution by solution by calling `rcc_return_next()` which returns subsequent solutions in lexicographic order of the id of the solution itself and the id of the solution match set. Each solution is a database relation instance of the type `qry_usr_soln`. The table `qry_usr_soln` lists pairs of region names with the corresponding relation between the pair of regions.

Algorithm:	MAXIMAL-CLIQUES
Input:	Graph G=(V, E).
Local:	FIFO set Q of binary RCC-8 constraints over V.
Output:	Path-consistent set equivalent to Θ, or NULL if inconsistency is detected.
Function:	extract_maximal_cliques(C, P, X)
Input:	clique to be extended C, set P of candidates vertices that are all connected to vertices in C, set X of vertices already processed and now excluded from the current extension.
Local:	Pivot point u_p used by the branch and bound method to truncate the search tree.
Output:	Set S of maximal complete subgraphs of G.

```
1.  if P = Ø then
2.      Report C as maximal clique
3.  end if
4.  for each v ∈ P U X do
5.      if |NEIGHBOURS(v)| < MIN({|NEIGHBOURS(w)|; w ∈ P U X }) then
6.          u_p ← v
7.      end if
8.  end for
9.  for each u ∈ P do
10.     if NOT u ∈ NEIGHBOURS(u_p) then
11.         P ← P-{u}
12.         Cnew ← C U {u}
13.         Pnew ← P ∩ NEIGHBOURS (u)
14.         Xnew ← X ∩ NEIGHBOURS (u)
15.         extract_maximal_cliques(Cnew,Pnew,Xnew)
16.         X ← X U {u}
17.     end if
18.  end for
19.  Return
```

Algorithm 3: Bron-Kerbosch maximal clique algorithm version 2 (from a note on clique enumeration algorithms, Cazals and Karande 2008)

Because redundant solutions always come up, there is another function `rcc_return_unique()`, which returns only unique solutions by grouping all equivalent solutions together and presenting them only once. Both functions return solutions in the same format including a solution number, and sketch id. After all solutions have been returned, a subsequent call to either of the functions loops back to the first solution and so on.

3.4.2.3. Illustration of Sketch-based Querying with an Example

Listing 2 shows the script for a query corresponding to the sketch shown in Figure 4. The query assumes that regions for the 'House Site' and 'Drive In' are already in the database. The first line calls a function that clears all variables and tables used during query evaluation. This is necessary to have a clean result because otherwise the solutions may contain unwanted information. This function applies only to the current session. A global version of it clears the data structures for all sessions, whether finished or active. Next is a series of calls to `unary_qry_constraint()` passing a different SQL statement for each variable. After this binary constraints are added to the query and then finally the query is executed by calling `rcc_eval_qry(false)`. The value `false` tells the

model not to apply constraint relaxation during execution. The statement `SELECT * FROM rcc_return_unique(2)` fetches next available unique solution of size greater than or equal to two from the query results. The output for the query is shown in Table 1.

```
SELECT rcc_clear_qry();

SELECT unary_qry_constraint('A', 'farms ', 'select distinct a.the_rcc_region as
the_rcc_region from farms a, farm_properties b
            where a.the_rcc_region = b.feature and (b.name = ''Feature'' and
b.value = ''blocks'')
');

SELECT unary_qry_constraint('B', 'farms ', 'select distinct a.the_rcc_region as
the_rcc_region from farms a, farm_properties b
            where a.the_rcc_region = b.feature and (b.name = ''Feature'' and
b.value = ''blocks'')
');


SELECT unary_qry_constraint('C', 'farms ', 'select distinct a.the_rcc_region as
the_rcc_region from farms a, farm_properties b
            where a.the_rcc_region = b.feature and (b.name = ''Feature'' and
b.value = ''driveways'')
');

SELECT unary_qry_constraint('D', 'farms ', 'select distinct a.the_rcc_region as
the_rcc_region from farms a, farm_properties b
            where a.the_rcc_region = b.feature and ((b.name = ''Feature'' and
b.value = ''houses'') and (a.description ~* ''farm house''))
');

SELECT binary_qry_constraints('A B EC AND A C DC AND A D EC AND B C EC AND B D
EC AND C D EC');

SELECT rcc_eval_qry(false);

SELECT * FROM rcc_return_unique(2);
```

Listing 2: Script for a sketch-based query corresponding to Figure 4

Session ID	Sketch ID	Solution Number	Solution Match set	Primary Variable	Primary Region	Secondary Variable	Secondary Region	Relation
7616	2	1	284	A	S1Block3	B	S1Block4	EC
7616	2	1	284	B	S1Block4	A	S1Block3	EC
7616	2	1	284	A	S1Block3	C	S1DriveIn	DC
7616	2	1	284	C	S1DriveIn	A	S1Block3	DC
7616	2	1	284	A	S1Block3	D	S1HouseSite	EC
7616	2	1	284	D	S1HouseSite	A	S1Block3	EC
7616	2	1	284	B	S1Block4	C	S1DriveIn	EC
7616	2	1	284	C	S1DriveIn	B	S1Block4	EC
7616	2	1	284	B	S1Block4	D	S1HouseSite	EC
7616	2	1	284	D	S1HouseSite	B	S1Block4	EC
7616	2	1	284	C	S1DriveIn	D	S1HouseSite	EC
7616	2	1	284	D	S1HouseSite	C	S1DriveIn	EC

Table 1: Unique solution returned from query executed by script in Listing 2

3.5. Summary

The design of the model presented here is simple in comparison with the original SQBS model of Egenhofer on which it is based. It does not take into account direction information and assumes that topological relations are simple and known exactly. This simplicity results in a larger set of results being returned for any given query. The inclusion of direction relations and detailed topological relations would narrow down the results because the constraints would be stricter.

Our model also differs from the SQBS in the way topological relations are treated by using a different formalism, namely the RCC-8. The formalism employed allows us to directly import some algorithms like the one for path-consistency, and to also use some paradigms such as the hierarchy of relations expressed in the RCC subsumption lattice to make inferences about different combinations of relations under different functions such as composition, inverse, etc. The latter point is used in the next chapter to develop our model for refining solutions by analysing the database further to find out if a group of regions in the database may constitute parts of a single region in a query.

4. Refining Solutions

As observed in some sketch maps obtained from an experiment done in this study and with the insights from psychology and GI Science, it is evident that spatial aggregation is a process that may be performed by people drawing sketch maps. The aggregation may be intentional, motivated by the application of the map, or it may be a consequence of an attempt to abstract from the reality only those aspects that are of interest, thereby simplifying the representation. Although it was not possible, in this study, to establish the extent to which aggregation may be used in the map sketching process, its occurrence presents challenges during query processing with sketch maps, since it is not possible to know, beforehand, whether constraints on a sketched region in a query can be satisfied by the combined constraints of a group of regions in the database.

In this chapter, an extension to the test model described in chapter 3 is presented. The purpose of the model is to identify possible situations where a query variable may represent an aggregation of several regions in the database and then discovering a set of spatial regions in the database that satisfy the topological constraints of the aggregate with other regions in the query result. The regions so found become the parts of the aggregate and can be included in the solution in its place. As observed by Tryfona and Egenhofer (1997), the task of determining the consistency between the relations of an aggregate region and a third region, and the relations between its part and the third region is not trivial, especially for regions with disconnected parts. In fact, for some configurations it is impossible to determine consistency based only on course topological relations. As such, the solution proposed does not provide a definitive answer to the question, whether a group of regions in the database maybe aggregated to satisfy a query, but rather an approximate answer. The result is approximate in the sense that, while all regions identified may not introduce inconsistency in the overall configuration, there are relations for which we cannot determine, based on topology alone, whether the given combination of regions contains regions that facilitate the satisfaction of a particular query constraint. For example, if the aggregate contains another region, then if all parts of the aggregate partially overlap this region it is not possible to know whether the parts together cover this region completely or only partially (Figure 5).

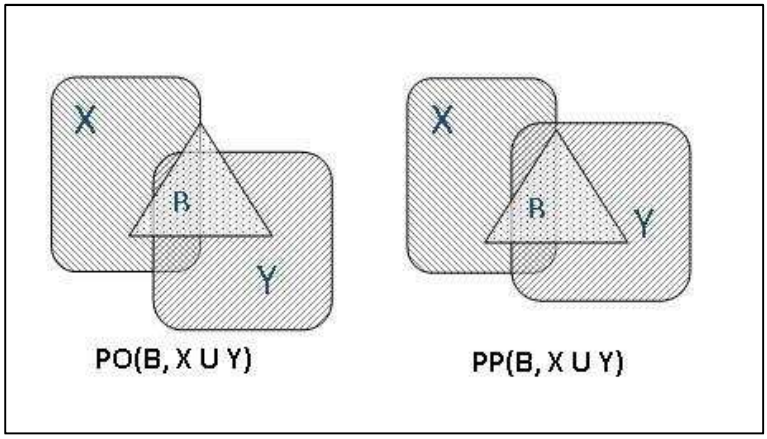


Figure 5: Two different relations realised between an aggregate (X U Y) and a third region (B) but with the same relations between the parts and B - PO(B, X) and PO(B, Y).

The model assumes that there is only one aggregate region in the query, if any, and that the parts aggregated to the region form a contiguous group of regions (no isolated, disconnected parts).

Because of these assumptions, we only attempt to find such aggregates if the number of matches in the solution's match set equals the size of the query or the size of the query minus 1. The first two assumptions are imposed to reduce the number of conditions that must be tested to determine whether the binary constraints of an aggregate's parts are consistent with the binary constraints of the aggregate (section 4.2.2 below). Additionally, if we assume that there is more than one aggregate region then there must be appropriate heuristics for discovering them, deciding how to combine database regions, and testing the consistency of the binary constraints between the groups of database regions with constraints of the aggregated query regions. These problems are beyond the scope of this study.

4.1. Analysis of the Problem

To determine whether a variable in the sketch query may be referring to a group of contiguous regions, we analyse the solutions generated during the initial query processing. The building of solution sets provides a pattern in which the results can be studied further. Consider a query with n variables. For each set of solutions with m common variable-region matches, the m common variable-region match subset will be called a basis for that set of solutions. Every basis in the set of solutions is itself an incomplete solution. The question is, given a set of incomplete solutions, whether we can generate a complete solution from the basis and a certain combination of variable-region matches or by introducing new variables to the query that correspond to parts of an aggregate region.

If there are $n - m$ variable-region matches that, if included in the basis, would lead to a complete solution, then we should already have the complete solution. So, in this case, there aren't any such matches and of the $n - m$ matches in each of the incomplete solutions there are some that fail to satisfy one or more binary constraints of the query.

For the case $m = n - 1$, we want a query variable A that satisfies the following conditions:

i. For each solution match set under consideration, A is matched with a different region or it is not matched at all;
ii. For each solution, the exclusion of A leads to the largest possible solution for the corresponding solution match set (a solution of size $n - 1$ for this case or m in general);
iii. For each solution, the inclusion of A in the solution leads to the exclusion of at least one other variable in the corresponding solution match set.

Condition (i) is a consequence of the choice of a basis. Since all solution match sets have the same basis of size $n - 1$, the only different match is the one for variable A. The second condition (ii) ensures that we do not have a complete solution using this basis and makes a point that any matching of A does not lead to a complete solution. Similar to condition (ii), condition (iii) provides a stronger test of the constraints of the variable A with respect to the basis – that is, every match for A is inconsistent with the constraints of at least one region in a consistent solution.

The process is based on the assumption that if the binary constraints on A can be satisfied by an instantiation of A to an aggregated region then there must be regions $d_1, \ldots, d_k$ in the database, with $k \geq 2$, such that the union $d_1 \cup \ldots \cup d_k$ is consistent with the constraints of A as specified in the query. The task thus becomes that of finding the regions d_j and testing the consistency of the configuration realised by replacing A with the d_js.

In the RCC theory, the sum function can be used to define the aggregate z of two regions x and y. The function is defined as (Bennett 2000, Randell, Cui, and Cohn 1992):

$$\forall xyz\big[sum(x,y) = z \leftrightarrow \forall w[C(z,w) \leftrightarrow C(x,w) \vee C(y,w)]\big]$$

To which is added the constraint that the parts making the aggregate must be contiguous proper parts of it:

$$\forall xyz[sum(x,y) = z \rightarrow PP(x,z) \wedge PP(y,z) \wedge C(x,y)\,] \qquad \text{(i)}$$

And finally the aggregate with more than two parts is constructed by summing a two part aggregate with another aggregate or other region making sure that condition (i) above holds. The constraint above is consistent with the definitions of the basic RCC relations and the sum function as given in the original paper on RCC since $PP(x,z) \rightarrow \forall w[C(x,w) \rightarrow C(z,w)\,]$.

4.2. Model Components for Processing Database Region Names

The first condition set on the regions to consider is that they come from the same sketch map as the regions in the original solutions. The second condition is that they must be contiguous. This is necessary because dealing with disconnected regions in spatial aggregation is more complex and beyond the scope of this thesis. The cluster structure of the current model is used as a first elimination strategy by retrieving only regions in the same cluster. Regions that are already in the basis are excluded, while regions which were previously matched to A and were included in at least one solution are given priority since they satisfy at least one constraint of A.

4.2.1. Region Name Sets

The model makes use of two groups of sets of region names, namely, constraint-local sets and global sets. There are two constraint-local sets for each binary constraint i of the variable A: PC_i, which is a set of regions that together or individually satisfy the constraint i, and N_{L_i}, the set of regions necessary for constraint i to be satisfiable by the regions in PC_i so that N_{L_i} is a subset of PC_i. Since the set P_G is processed all at once, regions are never removed from N_{L_i} but can be removed from PC_i if they have been identified as regions that cause inconsistency elsewhere.

Global sets facilitate the decision of success or failure for the process. The set P_G contains all regions selected for inclusion in the solution which must be a contiguous (connected) set; the set N_G contains members of every N_{L_i} in the model; and the set I_G is the set of inadmissible regions – which have been determined to be inconsistent with some constraint of A. The sets P_G and I_G are disjoint while N_G is a subset of P_G. Additionally, there is a variable i_c that is only instantiated when a constraint is found to be inconsistent, so that if i_c is not empty, then the process has failed. Formally, for l constraints we have the following restrictions on the sets:

$PC_i \subseteq P_G, 1 \leq i \leq l$ (i)

$N_{L_i} \subseteq PC_i, 1 \leq i \leq l$ (ii)

$N_G \subseteq P_G$ (iii)

$N_G = \bigcup_{i=1}^{l} N_{L_i}$ (iv)

$P_G \cap I_G = \emptyset$ (v)

As indicated above, regions are selected from the same cluster and added to the set P_G. For each constraint of A, each region in P_G is then processed against a set of rules and the information in the constraint-local and global sets. The result is that a region is either consistent or inconsistent with the constraint. An inconsistent region will be added to I_G and a consistent region to PC_i. The process fails when either a region in N_G is chosen to be added to I_G or when PC_i fails to satisfy the corresponding constraint i.

4.2.2. Rule Sets

The processing of the regions in the sets is done by evaluating a set of rules on a constraint level and on a global level. Constraint-local rules apply to each constraint differently depending on the relation implied by the constraint and determine the consistency of PC_i with respect to corresponding constraint i. The approximate nature of the result is inherent in the local rule sets because it is here that local consistency is determined. Global rules are applied the same for every constraint and are used to determine the consistency of the whole configuration.

Rules are grouped into rule sets which must all be executed whenever the rule set is invoked. Rule sets are a combination of rules and actions, and some contain only actions but they are all treated the same way during execution. The actions are operations on some given data structure of the model, like the constraint-local sets described in the previous section, with a specific data input such as a query constraint and database constraint. In contrast, rules are conditional statements that cause a specific action or group of actions to be invoked based on the evaluation of the condition.

4.2.2.1. Constraint-local Rules

There are eight constraint-local rule sets which correspond to the eight basic relations of the RCC-8 model. These rules are used to evaluate whether a region in P_G is necessary for constraint i, or it is consistent, inconsistent, or irrelevant to the constraint. In the following, B is a variable in the query with constraint R_{AB} with variable A which we want to verify for a group of database regions and $d_k \in P_G$ is any region in the database that is a potential part of A.

4.2.2.1.1. Rule Set 1: $DC = R_{AB}$

i. If $R_{d_kB} = DC$ then add d_k to PC_i,
ii. If $R_{d_kB} \in C$ then add d_k to I_G,
iii. If $|PC_i| < 2$ then label constraint R_{AB} as inconsistent and instantiate i_c to R_{AB}.

4.2.2.1.2. Rule Set 2: $EC = R_{AB}$

i. If $R_{d_kB} \in DR$ then add d_k to PC_i,
ii. If $R_{d_kB} \in O$ then add d_k to I_G,
iii. If $\forall d_l \in P_G, R_{d_lB} \neq EC$ then label constraint R_{AB} as inconsistent and instantiate i_c to R_{AB},
iv. If $\forall d_l \in PC_i - \{d_k\},\ R_{d_kB} = EC \wedge R_{d_lB} = DC$ then add d_k to N_{L_i} and update N_G.

4.2.2.1.3. Rule Set 3: $PO = R_{AB}$

i. If $R_{d_kB} \in \{DC, EC, PO, TPP, NTPP\}$ then add d_k to PC_i,
ii. If $R_{d_kB} \in \{EQ, TPPI, NTPPI\}$ then add d_k to I_G,

iii. If $\forall d_l \in PC_i,\ R_{d_lB} \notin \{PO, TPP, NTPP\}$ then label constraint R_{AB} as inconsistent and instantiate i_c to R_{AB},
iv. If $\forall d_l \in PC_i,\ R_{d_lB} \in PP$ then label constraint R_{AB} as inconsistent and instantiate i_c to R_{AB},
v. If $\forall d_l \in PC_i - \{d_k\},\ R_{d_kB} = PO \wedge R_{d_lB} \notin \{PO, TPP\}$ then add d_k to N_{L_i} and update N_G,
vi. If $\forall d_l \in PC_i - \{d_k\},\ R_{d_kB} = TPP \wedge R_{d_lB} \notin \{PO, TPP\}$ then add d_k to N_{L_i} and update N_G.

4.2.2.1.4. Rule Set 4: $EQ = R_{AB}$

i. If $R_{d_kB} \in \{TPP, NTPP\}$ then add d_k to PC_i,
ii. If $R_{d_kB} \in \{DC, EC, PO, EQ, TPPI, NTPPI\}$ then add d_k to I_G,
iii. If $\forall d_l \in P_G,\ R_{d_lB} \neq TPP$ then label constraint R_{AB} as inconsistent and instantiate i_c to R_{AB},
iv. If $PC_i \subsetneq \bigcup_{j=1}^{i-1} PC_j$ then label constraint R_{AB} as inconsistent and instantiate i_c to R_{AB}. This rule is not implemented, but serves as guiding rule during implementation.

4.2.2.1.5. Rule Set 5: $TPP = R_{AB}$

i. If $R_{d_kB} \in \{TPP, NTPP\}$ then add d_k to PC_i,
ii. If $R_{d_kB} \in \{DC, EC, PO, EQ, TPPI, NTPPI\}$ then add d_k to I_G,
iii. If $\forall d_l \in P_G,\ R_{d_lB} \neq TPP$ then label constraint R_{AB} as inconsistent and instantiate i_c to R_{AB},
iv. If $\forall d_l \in PC_i - \{d_k\},\ R_{d_kB} = TPP \wedge R_{d_lB} = NTPP$ then add d_k to N_{L_i} and update N_G.

4.2.2.1.6. Rule Set 6: $NTPP = R_{AB}$

i. If $R_{d_kB} = NTPP$ then add d_k to PC_i,
ii. If $R_{d_kB} \in \{DC, EC, PO, EQ, TPP, TPPI, NTPPI\}$ then add d_k to I_G,
iii. If $|PC_i| < 2$ (i.e. $PC_i = \{d_k\}$) then label constraint R_{AB} as inconsistent and instantiate i_c to R_{AB}.

4.2.2.1.7. Rule Set 7: $TPPI = R_{AB}$

i. If $R_{d_kB} \in \{DC, EC, PO, EQ, TPP, NTPP, TPPI\}$ then add d_k to PC_i,
ii. If $R_{d_kB} = NTPPI$ then add d_k to I_G,
iii. If $\forall d_l \in PC_i,\ R_{d_lB} \in \{DC, EC, NTPPI\}$ then label constraint R_{AB} as inconsistent and instantiate i_c to R_{AB}
iv. If $\forall d_l \in PC_i - \{d_k\},\ R_{d_kB} \in \{EQ, TPPI\} \wedge R_{d_lB} \in \{DC, EC\}$ then add d_k to N_{L_i} and update N_G.

4.2.2.1.8. Rule Set 8: $NTPPI = R_{AB}$

i. $\forall d_k \in P_G$ add d_k to PC_i.
ii. If $\forall d_l \in P_G,\ R_{d_lB} \in \{EC, DC\}$ then label constraint R_{AB} as inconsistent and instantiate i_c to R_{AB}.

4.2.2.1.9. Derivation of Constraint-Local Rules

The derivation of these rules is made using the RCC8 composition table as well as the relational lattice of the RCC and its corresponding theorems. Members of the sets PC_i are taken from the set of regions such that for any d_k and every possible B, $(R_{AB} \circ R_{Bd_k}) \cap \{TPPI, NTPPI\} \neq \emptyset$, from which follows directly that $R_{d_kA} \in \{TPP, NTPP\}$. Members of the set I_G are selected in such a way

that $\left(R_{AB} \circ R_{Bd_k}\right) \cap \{TPPI, NTPPI\} = \emptyset$ which ensures that they cannot be proper-parts of A. The sets N_{L_i} contain all such regions from PC_i that if removed from the spatial configuration in which constraint i is realised, the whole set would become inconsistent with the constraint.

The rational for using each of the conditions for adding a region to N_{L_i} and determining the local inconsistency at R_{AB} are given in the following paragraphs.

Rule 1-iii: We take $|PC_i| < 2$ to ensure that we have at least two regions that satisfy the given constraints. Otherwise then only one or no region satisfies the given constraint which implies inconsistency of the selected set with the constraint or a violation of the condition that there must be two or more parts of an aggregate region.

Rule 2-iii: If the aggregate is externally connected with the third region B, then at least one of its parts must be externally connected with B.

Rule 2-iv: If among the potential parts of the aggregate region A there is only one region that is externally connected to B then this region is necessary for the constraint to hold and removal of the region from the set will lead to the whole being inconsistent with the constraint between A and B.

Rules 3-iii, 3-iv: If the aggregate partially overlaps with the third region B, then at least one of its parts must overlap with B. Since B is not part (P) of A, the parts of A overlaping with B must either partially overlap or be properly contained by B. Similarly, there must part be parts of A that are not properly contained in B. Otherwise, the set is inconsistent with the constraint.

Rules 3-v, 3-vi: If no other part overlaps with B then the only partially overlaping region is necessary for the constraint to hold. Similarly if no other region is contained in B and no region overlaps B then the region that is properly contained by B is necessary for the satisfaction of the constraint. In this latter case, rule 3-iv guarantees that there is a region that is not properly contained by B and the contiguity requirement guarantees that the two parts (regions) are connected.

Rule 4-iii: There must be a tangential proper part of A that is also a tangential proper part of B. Otherwise A is a proper part of B.

Rule 4-iv: All regions that satisfied other constraints must also satisfy this one and they must be proper parts of the region B of this constraint. This observation is sufficient but not necessary for the model to determine the inconsistency of the constraint since all regions that individually give rise to inconsistency are removed from P_G and the corresponding PC_i anyway.

Rule 5-iii: As a tangential proper part of B, A has to have a tangential proper part that is also a tangential proper part of B. Otherwise if all tangential proper parts of A are non-tangential with B then A is non-tangentially connected with B since A is a sum of discreet proper parts as restricted by conditions set in section 4.1 above.

Rule 5-iv: If there is only one tangential proper part, then it must be necessary and its exclusion would lead to the set being inconsistent with the constraint.

Rule 6-iii: Since A is a non-tangential proper part of B then all its parts must also be non-tangential proper parts of B.

Rule 7-iii: If no individual part of A is equal to or properly contains B then either B is not properly contained in A or the containment is achieved by different parts of A together exhaustively covering every part of B. With the information available from the topological relations among the regions and

the method employed in the proposed model, it is not possible to determine whether the constraint is satisfied or not but we only eliminate obvious inconsistent cases.

Rule 7-iv: Here the assumption is that if there is only one region of which B is equal to or is a tangential proper-part of, then B and that region have a shared tangential boundary section or point that is not connected with any other parts of A. This is of course not sufficient to determine the consistency but it facilitates the determination of inconsistency when an attempt is made to move the region to the set I_G.

Rule 8-ii: If B is a non-tangential proper part of A then there is a region that is part of A and overlaps with B. Otherwise B cannot be a proper part of A and not overlap with any of A's parts.

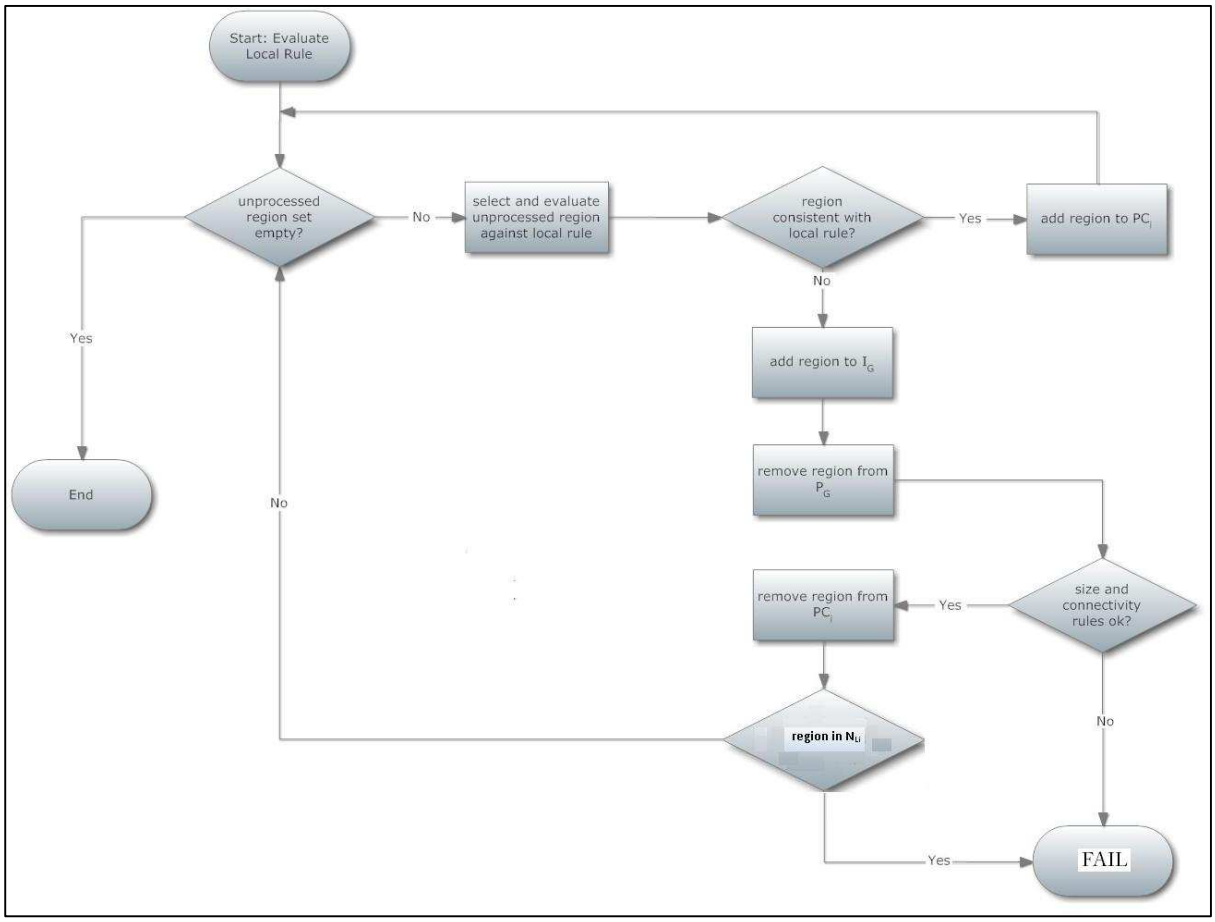


Figure 6: General procedure for evaluating constraint local rules.

The execution of constraint-local rules is uniform but differs only in the number of decisions to be made (Figure 6). A group of regions will be determined to be inconsistent with a given constraint when all regions in the group have failed to meet the minimum requirement for the constraint to be satisfied. This position is reached when the global variable i_c is instantiated. Otherwise, an individual region will cause the set to become inconsistent when it is determined to be in I_G and N_G at the same time.

4.2.2.2. Global Rules

Global rules enforce global constraints that we put on both constraint-local and global sets. Each constraint is enforced by one or more rules in one or more rule sets. The rule sets are grouped by the action that triggers them.

4.2.2.2.1. Rule Set 9: Remove a Region from P_G

i. If $|P_G - \{d_k\}| < 2$ then FAIL,
ii. If ¬IS-CONNECTED-SET($\bigcup_i(PC_i) - \{d_k\}$) then FAIL,
iii. If $|\bigcup_i(PC_i) - \{d_k\}| < 2$ then FAIL,
iv. If ¬IS-CONNECTED-SET($P_G - \{d_k\}$) then select a region-name say d_l from any PC_i and remove P_G −CONNECTED-SET-OF($P_G - \{d_k\}, d_l$) from P_G and PC_i
v. Remove d_k from PC_i,
vi. Remove d_k from P_G.

4.2.2.2.2. Rule Set 10: Add a Region to I_G

i. Add d_k to I_G,
ii. Remove d_k from P_G.

4.2.2.2.3. Rule Set 11: Remove a Region from PC_i

i. If $d_k \in N_{L_i}$ then FAIL,
ii. Remove d_k from PC_i.

4.2.2.2.4. Rule Set 12: Add constraint to i_c

i. FAIL.

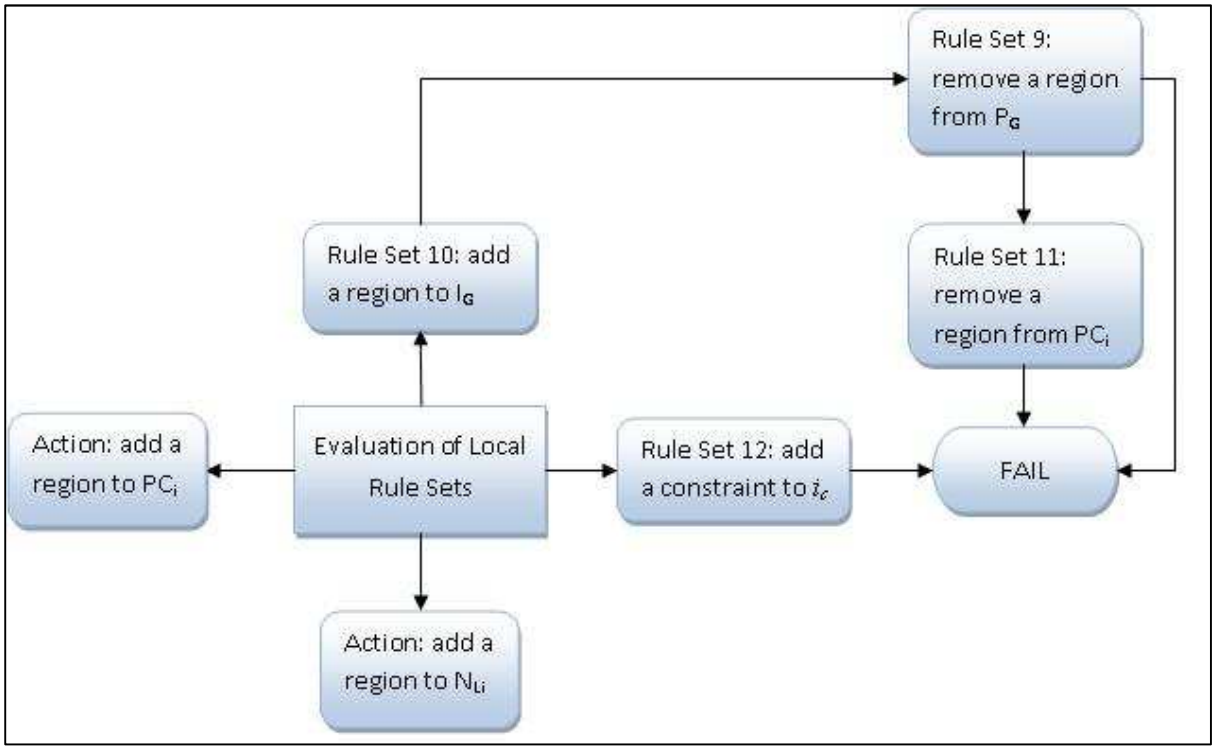


Figure 7: Call sequence of main global rule sets and actions

4.2.2.2.5. Executing Global Rules

Within each rule set, the rules are executed in order so that rules at the top of the list are executed first and those at the bottom last. Global rules are intended to be triggered automatically when the corresponding action (e.g. add a region-name to a set) is initiated by another rule or function that is external to the triggered rule (Figure 7). However due to limitations of the environment in which the experimental implementation was programmed, the rules are implemented inside functions that are called to execute the corresponding actions. The action ***remove*** attempts to remove a region name from a set but has no consequence if the region name is not found. The function IS-CONNECTED-SET(S) performs a depth first search on a graph whose set of vertices is S and whose set of edges is assumed to be the set of binary (RCC-8) relations $\{R_i\}_1^b$ on members of S; $b \leq \frac{n(n-1)}{2}$ if the set S has n vertices; and $R_i \neq DC, 1 \leq i \leq b$. In the implementation S are region names. The function returns true if all members of S were reachable in the depth first search. The function CONNECTED-SET-OF(S, *region-name*) returns the subset of contiguous regions in S containing *region-name*. FAIL is a function that flags failure of the selected configuration to satisfy the constraints of A, and causes the process to exit.

4.3. Overall Model Design and Implementation

The overall design of the model is made so that the execution of rules on the defined sets leads to inconsistency whenever possible. First, the original solutions are evaluated and from all incomplete solutions of size $n-1$, the process tries to find groups of solutions that have a common basis and a variable satisfying the three conditions of section 4.1 (Figure 8). In the following, references to A, B, and d_k have the same meaning as the previous section.

Once the solutions are identified, they are processed one after the other in no specific order. For each set of solutions identified, region names are selected from the corresponding sketch map and put into a set Q from which groups of contiguous variables are selected in order of the size of the largest solution in which they were included as matches for variable A. Regions in Q cannot already be in the basis of the set of solutions being processed. The selected regions are added to the set P_G but they are not immediately deleted from Q. Finally, for each binary constraint of A, P_G is processed against the corresponding rules (e.g. if $R_{AB} = EC$ evaluate rule set 2) as shown in Figure 6 above. If the selected set of regions fails, they are deleted from Q and new regions are added to P_G from Q and the process is repeated. The process keeps track of the contents of the global sets P_G and I_G because all regions removed from P_G must be left in Q for reprocessing just as long as they are not in I_G and the exit function FAIL has not been called.

The main function that implements the discovery of regions that are regarded as parts of an aggregate region that is in the query sketch loops through all groups of contiguous regions in Q until Q is empty (Figure 9). Although the set Q does not function as a queue per se, the name is used to highlight the fact that groups of regions that have been added to the set are processed in some predetermined order. The process exits when every basis identified in the first step has been processed. The solutions derived during the process, if any, are added to the original solutions as part of the final query results. The variable A is removed from the original association graph and a new variable is added for every additional region. The new variables are named by appending a number to the name of the original variable separated by an underscore. For example, if we find three regions to replace A, then they will be named A_1, A_2, and A_3.

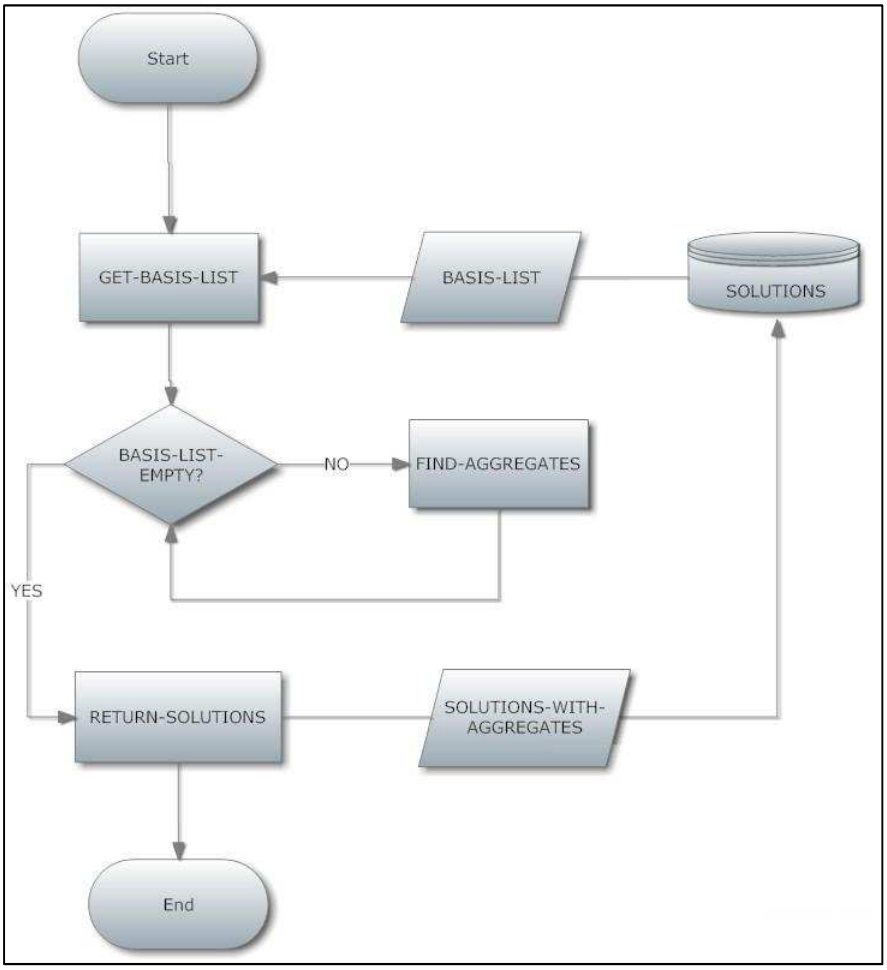


Figure 8: Flowchart of the main process for refining solutions

4.4. Summary

The procedure for refining the solutions for a query described in this section uses a dual approach for eliminating inconsistent candidates from being part of the final solution. On one hand, a group of regions can fail to satisfy the connectedness and part-hood requirements of the relation between the aggregate and a third region. On the other hand, a region may be deemed as being necessary for one constraint and inadmissible for another constraint simultaneously. The process of evaluating constraints of the chosen variable against a set of candidates is executed twice because some constraints that were evaluated earlier may not be satisfied by the regions remaining at the end of the process. This is because some regions will have been moved to I_G at a later time. The second time that the constraints are evaluated, no regions will be moved to I_G because all regions that are in P_G at this time already passed the test the first time around.

The procedure is approximate because there are scenarios that require more information to decide whether the given combination of regions satisfies the constraints of the aggregate (Rule sets 4, 7, 8). Rule set 8 is particularly weak since all regions are admissible regardless of their relation with the third region. A region is determined to be necessary for a specific constraint to be satisfied if it plays the role of a region bound by the existential quantifier in the definition of relation (in the RCC theory – see Randell, Cui, Cohn 1992). Such a region is selected by ensuring that no other region plays the

same role. While the results are approximate, they are always consistent with the configuration in the database since all constraints are tested and candidate regions are forced to be contiguous.

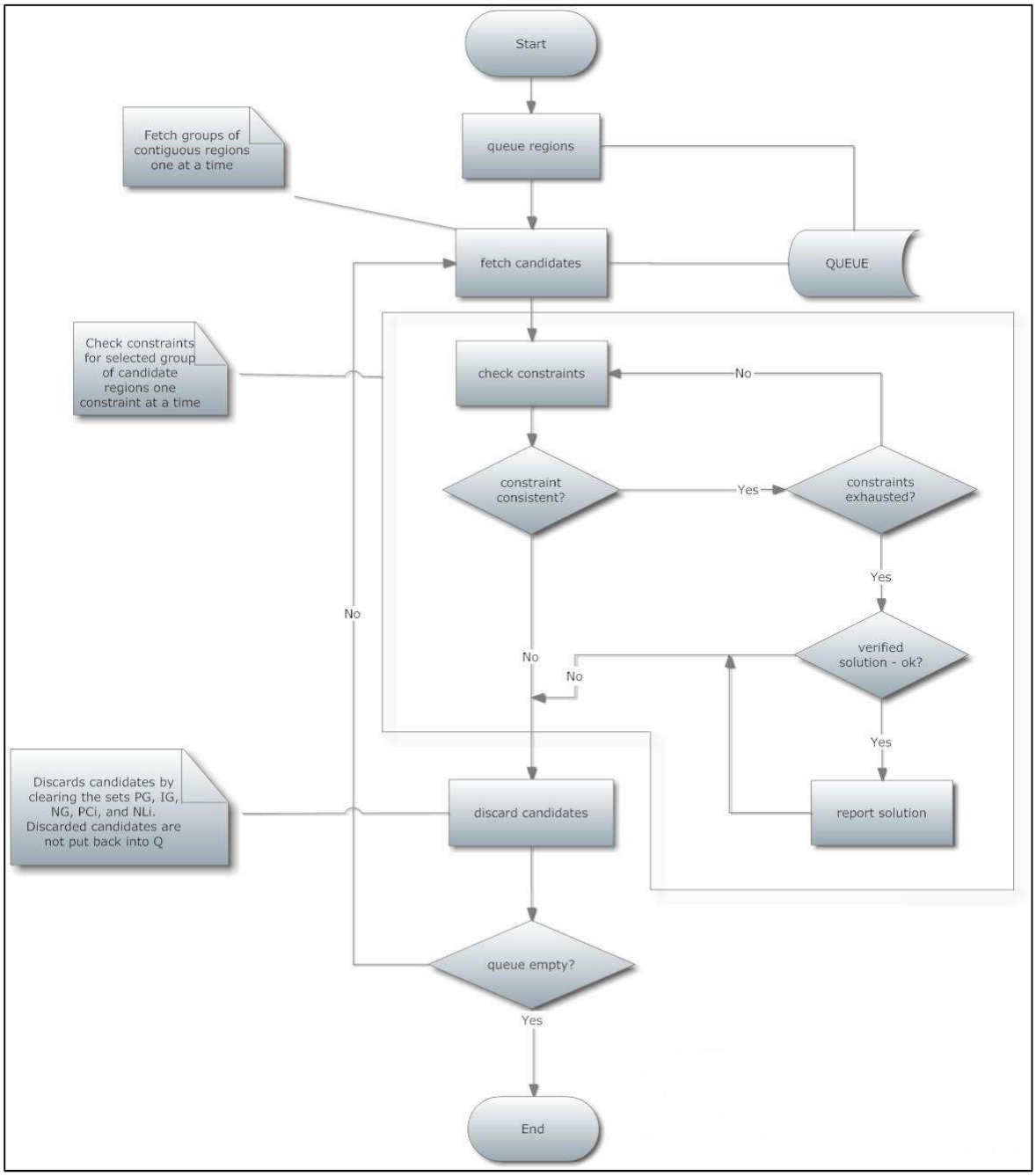


Figure 9: Procedure for finding potential parts of a region in the query sketch whose constraints are not satisfiable by the any individual region in the database.

5. Experimental Results and Discussion

Both the original model of Chapter 3 and the extension described in Chapter 4 were tested using sketch queries on a database populated with topological information of objects from other Sketch maps. Six sketch maps were employed in the evaluation of the model and the testing of the assumptions of the study. The experiment involved a collection of an initial six sketch maps with the same or similar themes. These were then shown to a group of subjects who identified certain features in a sketch and were then asked to draw a query sketch that represented some objects of interest from the original sketches.

5.1. Sketch Map Selection

Sketch maps for the experiment had to be chosen with consideration of the processing capabilities of the experimental database model. Within the limitations imposed by the database model, themes that would be easy for respondents to work with should be selected so that each query sketch should contain adequate information to pass on to the database. Lessons from research on spatial cognition also informed the search and selection of sketch maps.

5.1.1. Application of Sketch Map and Data

The sketch map has to have a meaningful application or purpose. Specifically, the sketch must express some phenomenon of which the information can be used to make a decision. Additional information about geographic features must also be present in the sketch either as annotations in the sketch or as symbols described outside the sketch. Sketch maps of agricultural farm lands will be used in this study. Sketch maps of agricultural farms may be simple but rich in information because a wide variety of physical features, phenomena and their relationships may be mapped.

5.1.2. Sketched Area and Mapped Features

Psychologists have also argued that recollection of spatial information is, in part, dependent on its purpose and the importance of the task being applied. Two main types of sketches are distinguished in this regard: survey maps and route maps. According to Tversky (2002), survey maps have to do with regions viewed from a bird's eye perspective while route maps are more linear and based on a ground exploration view. Survey maps contain spatial relations between several features while route maps concentrate on spatial relations between pairs of features. In particular, because of their nature, it would seem that survey maps tend to have a high level of connectedness in the sense that more objects in the sketch maps would be non-disjoint or contiguous.

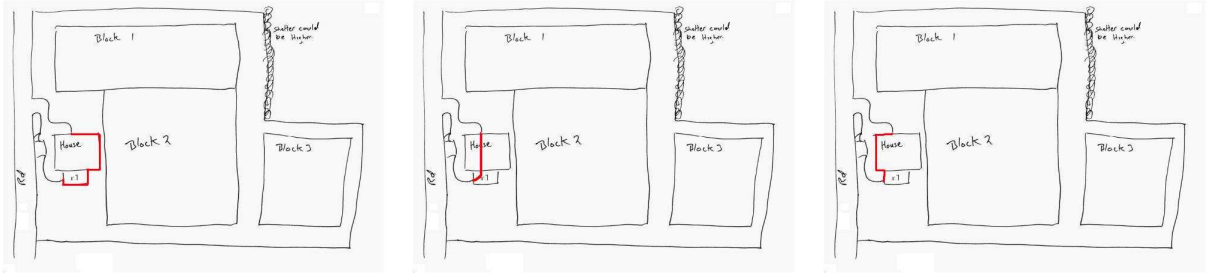
A main limitation of the model employed in this study is that it deals only with topological information thereby requiring a significantly high number of non-disjoint regions to be present in any given spatial scene that is processed. As such, contiguity of spatial features was a critical aspect when selecting the maps to use for the study. Consequently, sketch map types that are used for tasks in which topological interaction is generally not emphatic were avoided. These for example include route descriptions.

5.1.3. Selected Sketch Maps

Six sketches were selected from a corpus of more than 30 sketches. Four of the sketches were drawn by farmers in New Zealand who participated in a study on farm management practices conducted by the Agricultural Research Group on Sustainability (ARGOS) at University of Lincoln, New Zealand (Read, Hunt, and Fairweather 2005). One sketch was taken from a study on small-scale intensive farming by Aragó and Molin (2009). The last sketch is a map of a farming area in the city of Blantyre, Malawi and was provided by one respondent from the city who was asked to draw a sketch of his farm and a few features around it such as other farms, roads, and houses. Two of the ARGOS sketches are sketches of the same farm drawn by different members of the same family (see sketches 3 and 6 in Annex A: Input Sketches)

5.2. Graphical Analysis of Sketch Maps

The approach used for extracting topological relations from sketch maps is to identify all regions in the sketch-based on labelling. Labels include all names of specific objects in the sketch that have been directly written into the sketch. Labels can also provide feature type information such as 'shed' or 'shed 1' because people don't usually assign specific names to certain things such as residential houses or private parking lots. Annotations refer to all other text and symbols that provide context for the objects in the sketch. For example, items in a legend or descriptive text within the sketch are annotations. The next step is to decide which of the remaining unlabeled objects can be interpreted as regions and of those that have been identified as regions either a feature class or a unique name is assigned. However, all of the sketches used had at least eight out of ten features labelled with descriptive names (e.g. block 1 is an area of farm land with a specific crop or purpose). Elements (lines and other objects) in the maps for which their purpose was difficult to make out and lines that should be represented as regions (e.g. rivers) were analysed on a case by case basis. Special cases were identified in almost all the sketches (e.g. Figure 10) and some of them are discussed below.

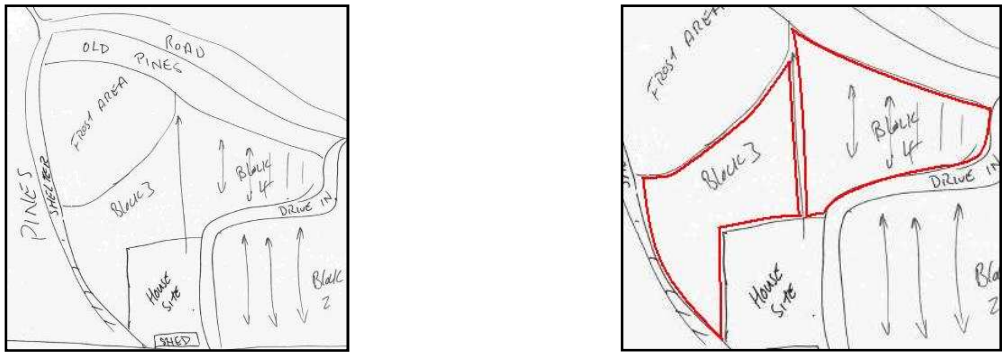


(a) House and r1 externally connected to area of farm	(b) House and r1 partially overlap area of farm	(c) House and r1 are tangential proper parts of area of farm

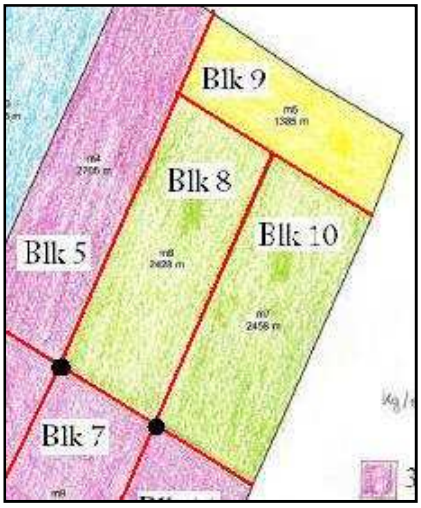
Figure 10: Three possible choices for the relation between a house and a farm, and between an attachment to the house and the farm. Option (b) is not acceptable since it entails the house overlapping the road (quite unusual). Consequently option (c) is chosen.

A region was taken to be the area enclosed by a closed loop that is not self intersecting including the loop itself (Figure 11 (a)). This reflects the closed interpretation of the RCC in point set topology. And, because we do not know the sequence in which the lines in the sketch map were drawn, we only look for loops that are closed or, almost closed and have been labelled appropriately. Adjacent regions both contain their shared boundary points (Figure 11 (b)). Where regions cannot be directly identified from the particular shapes and patterns drawn the label or annotations associated with

the drawn shapes or patterns are used to infer the type of feature that is being represented and then a decision is made on whether to consider the shapes/pattern as a region or to ignore them (e.g. patterns on upper-right boundary in the sketches of Figure 10). This approach is acceptable because in fact among all object features or properties, the class of an object has the highest diagnostic effect during object categorisation (Tversky 1977, Nedas and Egenhofer 2008).

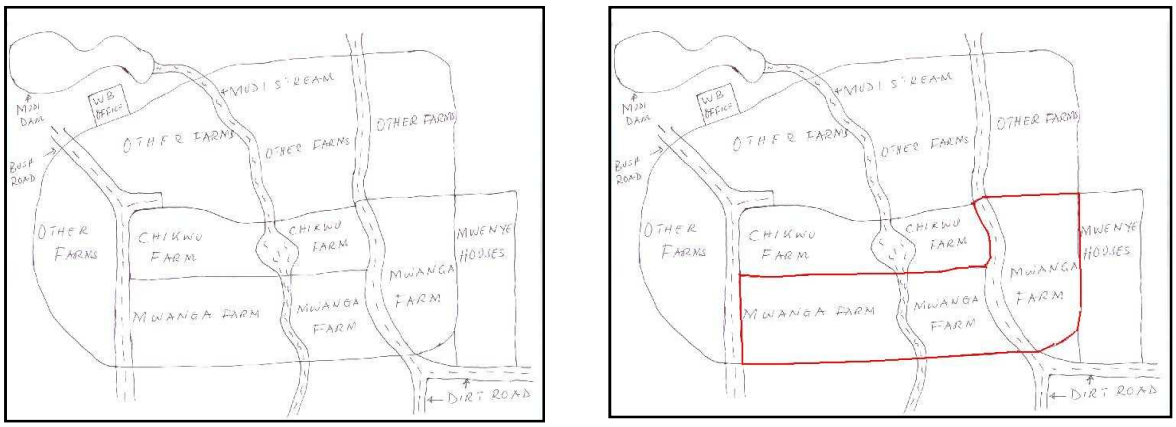


(a) Identifying regions in a sketch map. The original sketch is shown in the left and on the right are some identified regions (marked with red borders). Drawing errors have been corrected.



(b) Shared boundaries mean that in grid configuration regions are externally connected to both the regions adjacent (Blk 5, Blk 8) and opposite them (Blk 5, Blk 7). The following relations hold:

EC(Blk 5, Blk 8), EC(Blk 5, Blk 7), EC(Blk 7, Blk 8), EC(Blk 8, Blk 9), EC(Blk 8, Blk 10), EC(Blk 9, Blk 10), EC(Blk 7, Blk 10).



(c) Analysis of the boundary-boundary interactions between several regions help to decide the coverage of a region and relations between different regions.

Figure 11: Extracting regions and their relations from sketch maps

Regions divided by features such as roads or rivers are recognized as individual wholes as opposed to being two separate regions (Figure 11 (c)). Also, the interactions between the borders of the regions help to determine the relation between pairs of regions (Figure 11 (c)).

As regions are identified, they are classified into feature types. The feature classification used is directly adopted from the ARGOS study (Read, Hunt, and Fairweather 2005) and are later used to specify unary constraints when querying the database. The ARGOS study identified two sets of categorisations, one for kiwi fruit orchards and the other for sheep and cattle farms. While the two differ in the number of categories and number of features in each category, features common to both types of farms were categorised the same way for both cases. Because we do not use all the sketch maps and details of the ARGOS study, only some of the features and categories were used in this study as shown in (Table 2).

Categories	**Features**
Spatial organisation	Blocks, boundaries, *farms*[*]
Wind	Shelter, shelter trees, prevailing wind, wind damage
Buildings	Houses, sheds, hay barns , pack-houses
Transport	Roads, driveways, loading areas, airstrips, bridges
Social context	Neighbours, private businesses[*]
Other biota based activities	Other crops, pine trees, other trees, compost
Landscape morphology	Slope, aspect, gullies
Climate	Frost protection, frost areas, altitude, climate
Water	Water sources, streams and rivers, irrigation, Lakes, and ponds, water tanks, drainage
Biotic context for management	Soils, bush
Stock management	Animals, laneways, sheep, cattle and stock yards, silage pits

Table 2: Geographic features and their categorisations

To illustrate the procedure, Figure 12 shows the sketch map of a kiwi farm that was used as one of the input data sources. The regions marked by a red outline include two of those shown in Figure 11. The full list of regions extracted from the sketch including their assigned feature classes are shown in Table 4. Where it is not clear what categorisation to use, some regions are assigned more than one feature type or category. For example, in the sketch map in Figure 12, the house site can be a proposed site for a new house or it can be a part of the farm where the farmhouse is located. The list of binary topological relations identified for each pair of regions is shown in Table 3. The information in the two tables is used either for data input or for creating sketch-based queries to the database. The scripts for creating the tables used in this experiment and adding the information extracted from the sketch in Figure 12 to the database is given in Annex B: Sample scripts. The same procedure was used to analyse and input all the sketches in the study as well as to create the queries (Listing 3 of Annex B: Sample scripts).

[*] These were included as a feature only in this study but were not present with given names or meanings in the ARGOS study.

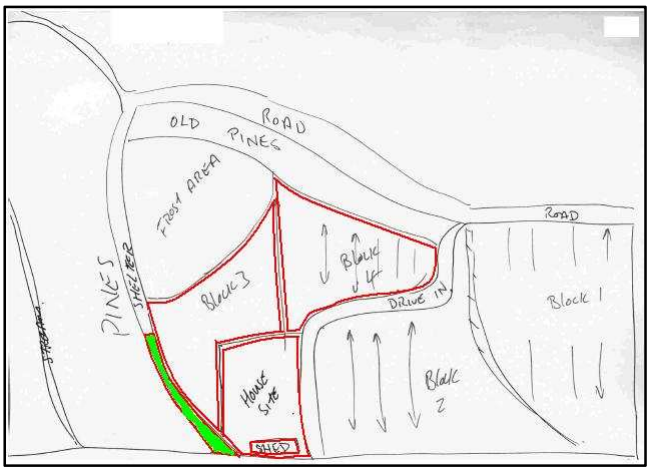


Figure 12: Example sketch map with a group of regions identified in the sketch (red borders).

Primary Region	Secondary Region	Relation
S1Road	S1OldPines	EC
S1Road	S1Pines	EC
S1Road	S1Block1	EC
S1Road	S1FrostArea	EC
S1Road	S1Shelter	EC
S1Road	S1Block3	EC
S1Road	S1DriveIn	EC
S1OldPines	S1FrostArea	EC
S1OldPines	S1Block4	EC
S1OldPines	S1DriveIn	EC
S1FrostArea	S1Block4	EC
S1FrostArea	S1Block3	EC
S1Block3	S1Block4	EC
S1Block3	S1HouseSite	EC
S1Block3	S1Shelter	EC
S1Block4	S1Shelter	DC
S1Block4	S1DriveIn	EC
S1HouseSite	S1Shelter	EC
S1HouseSite	S1DriveIn	EC
S1HouseSite	S1Shed	NTPPI
S1DriveIn	S1Block2	EC
S1Block2	S1Block21Boundary	EC
S1Block1	S1Block21Boundary	EC

Table 3: Topological relations between pairs of regions from the sketch in Figure 12. Inverse relations and relations where regions are disjoint are not included.

Name	Description	Region name (the_rcc_region)	Level	Value
Road	Main road passing near farm	S1Road	Category	transport
Road	Main road passing near farm	S1Road	Feature	roads
Old Pines	Old pine tree lot	S1OldPines	Category	other biota based activities
Old Pines	Old pine tree lot	S1OldPines	Feature	pine trees
Pines	Pine tree lot	S1Pines	Category	other biota based activities
Pines	Pine tree lot	S1Pines	Feature	pine trees
Frost Area	Frost Area	S1FrostArea	Category	climate
Frost Area	Frost Area	S1FrostArea	Feature	frost area
Block 1	Farm division block 1	S1Block1	Category	spatial organisation
Block 1	Farm division block 1	S1Block1	Feature	blocks
Block 2	Farm division block 2	S1Block2	Category	spatial organisation
Block 2	Farm division block 2	S1Block2	Feature	blocks
Block 3	Farm division block 3	S1Block3	Category	spatial organisation
Block 3	Farm division block 3	S1Block3	Feature	blocks
Block 4	Farm division block 4	S1Block4	Category	spatial organisation
Block 4	Farm division block 4	S1Block4	Feature	blocks
House Site	Farm house yard or lot	S1HouseSite	Category	spatial organisation
House Site	Farm house yard or lot	S1HouseSite	Category	buildings
House Site	Farm house yard or lot	S1HouseSite	Feature	blocks
House Site	Farm house yard or lot	S1HouseSite	Feature	houses
Shed	Farm storage and work shed	S1Shed	Category	buildings
Shed	Farm storage and work shed	S1Shed	Feature	sheds
Shelter	shelter	S1Shelter	Category	wind
Shelter	shelter	S1Shelter	Feature	shelters
Block 2-1 Boundary	Farm division boundary between block 2 and block 1	S1Block21Boundary	Category	spatial organisation
Block 2-1 Boundary	Farm division boundary between block 2 and block 1	S1Block21Boundary	Feature	blocks
Block 2-1 Boundary	Farm division boundary between block 2 and block 1	S1Block21Boundary	Feature	boundaries

Table 4: Data about spatial regions extracted from sketch map in Figure 12

5.3. Query Analysis

As discussed in section 3.4.2 above, querying the database is a three step process. First, the unary and binary constraint are specified, and then a command is given to evaluate the query. Finally, the results are retrieved using one of two functions (for all solutions or for unique solutions only). Optionally, the refinement procedure can be called to check if some incomplete solutions can be made complete by searching for groups of regions that satisfy the constraints of a query variable missing from a unique solution (Chapter 4).

Queries were performed on the database in three categories. The purpose of the exercise was to determine whether the implementation would search and retrieve the desired results given input queries for which the target sketch maps are known a priori. The first category (1) comprised queries taken from the original sketch maps as is. The second category (2) comprised queries created

Target Sketch	Query No.	Total Number of Solutions	Number of Unique Solutions	Unique Solutions With Correct Matching (Y/N)	Number of Complete Solutions	Complete Solutions With Correct Matching (Y/N)	Number of Regions in the Query	Number of Regions in Target Sketch	Number of Refined Solutions	Refined Solutions With Correct Matching (Y/N)	Max. no. of Database Regions that Matched Aggregated Region	Number of sketches in results
1	1_1_1	185	101	Y	1	Y	13	13	0	N	0	2
5	1_1_2	42	35	Y	1	Y	9	9	0	N	0	1
3 and 6	1_1_3	322	178	Y	1	Y	18	18	0	N	0	4
1	2_1_1	18	12	Y	0	N	7	13	1	Y	2	1
1	2_2_2	40	11	Y	1	Y	9	13	0	N	0	1
2	2_1_3	22	20	Y	2	Y	7	8	0	N	0	1
2	2_2_4	16	8	Y	2	Y	5	8	0	N	0	1
4	2_1_5	40	30	Y	2	Y	7	12	0	N	0	1
5	2_1_6	4	3	Y	1	Y	5	9	0	N	0	1
5	2_2_7	33	27	Y	1	Y	7	9	0	N	0	1
3 and 6	2_1_8	61	38	Y	1	Y	10	18	0	N	0	2
3 and 6	2_2_9	5	4	Y	1	Y	6	18	0	N	0	1
1	3_1_1	63	49	Y	0	N	10	13	1	Y	4	3
1	3_2_2	70	51	Y	0	N	11	13	1	Y	3	1
4	3_1_3	1644	350	Y	0	N	8	12	2	Y	5	1
4	3_2_4	328	126	Y	0	N	8	12	2	Y	5	1
5	3_1_5	12	9	Y	0	N	7	9	1	Y	3	1
5	3_2_6	38	32	Y	1	N	8	9	0	N	0	1
3 and 6	3_1_7	541	393	Y	0	N	16	18	1	Y	3	2
3 and 6	3_2_8	373	168	Y	0	N	17	18	1	Y	2	3

Table 5: A summary of query results. Query numbers 1_1_1 - 1_1_3 correspond to the first category of queries, 2_X_X to the second category and so on.

from sketches provided by subjects of the experiment. In the final category (3) were queries created from the original sketches by combining groups of connected regions into a single query variable. These were used to test the extended model. For all three categories, the results of the queries were tabulated as shown in Table 5. For the first category of queries, only three of the sketches were used. The second category includes queries aimed at all sketches while the third category does not include any queries targeted at sketch 2. The data in Table 5 have been arranged into the three categories with category 1 at the top and category 3 at the bottom.

5.4. Discussion of Results

In this section, along with a general discussion of the results in Table 5, we discuss the factors influencing the output of a query in the two models and give some suggestions for improving the quality of query results. A total of 20 queries were performed against the database. Each query was unique in the combination of features represented or the set of binary topological relations between them.

In Table 5, the column *Total Number of Solutions* lists the total number of cliques extracted from all the association graphs of the corresponding query while *Number of Unique Solutions* are those solutions that are unique in the whole query. *Unique Solutions With Correct Matching*, *Complete Solutions With Correct Matching*, and *Refined Solutions With Correct Matching* indicate whether there was a solution in which the regions of the query were matched with the intended database regions. *Number of Refined Solutions* is the total number of new solutions included in the results after applying the refinement procedure. Refined solutions are always unique since the basis from which each refined solution is generated is itself a unique incomplete solution. *Max. no. of Database Regions that Matched Aggregated Region* represents the number of database regions that were determined to be parts that together equal the aggregated region of the query.

5.4.1. Solutions to a Query

The solutions to a query vary widely depending on both the unary and binary constraints imposed on the regions in the query and the data in the database. As seen in Table 5, the total number of solutions returned from the queries is in most cases are so high that it may not be possible to determine whether a desired result has been given or not, let alone to find it. The high number is a result of the way in which solutions are created. As discussed in section 3.4.2 regions are matched with variables first on the basis of unary constraint satisfaction and then packaged into solution match sets that are evaluated later on in the process. The procedure used for creating solution match sets leads a very large number of those sets being created since all possible combinations of matches are used as long as every variable and every database region appear in each set at most once. So, there will be an equally high number of association graphs and therefore, an even higher number of solutions, since each graph has the potential to yield more than one solution. This effect may be reduced by considering only solutions with a high number of matches. An effective way to do this is by incorporating the scene completeness component of the similarity model developed by Nedas and Egenhofer (2008). A lower bound or lower threshold for acceptable values of component similarity can be imposed on the query results to limit the number of results returned to the user. This is an effective method because from the results obtained during testing, for all queries with more than 20 solutions, solutions with lower numbers of matches were more numerous than those with higher numbers of matches. For example, Figure 13 shows a plot of number of regions per

solution against the number of solutions for query 3_1_3 of Table 5, with a line of best fit generated by an exponential decay function. Lower bounds of 5, 6, and 7 regions per solution will reduce the total number of solutions returned for the query to 404, 80, and 10 respectively.

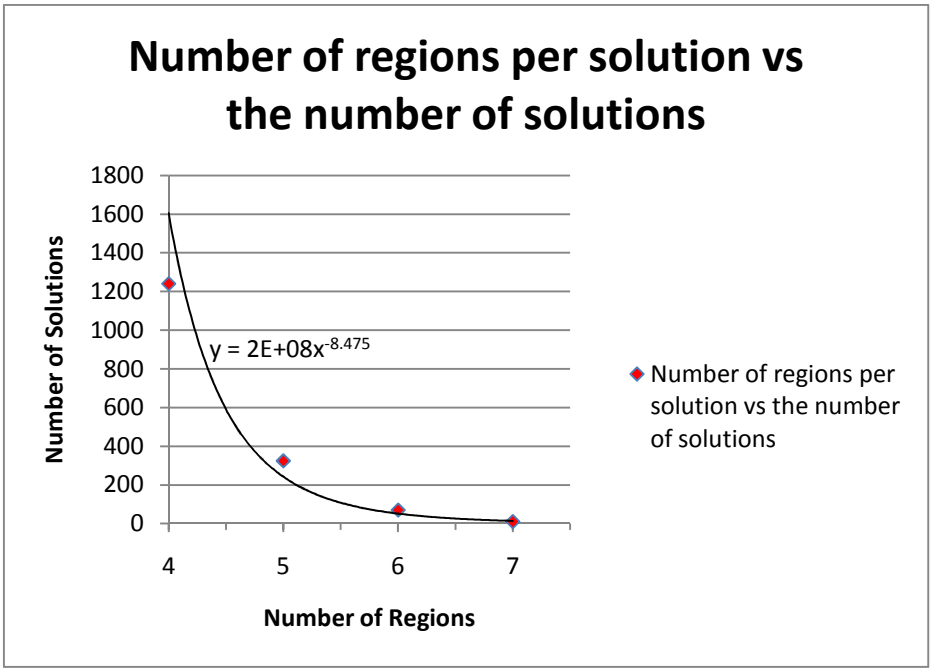


Figure 13: Plot of number of regions per solution against number of solutions returned for query 3_1_3

For all queries, unique solutions are much less than the total number of solutions but are equally high. This is partly because in the analysis subsets of larger solutions were considered as being different from the larger solution. While the desirable result is the largest solution containing all smaller solutions whose matches are subsets of its matches, identifying such a solution and eliminating the corresponding smaller solutions was difficult due to limitations of the implementation language (pl/pgsql) which are outlined in section 5.5. Figure 14 below shows a scatter plot of the total number of solutions against number of unique solutions from all results. The plot indicates a possible direct relationship between the two variables as expected but in this case with two possible outliers. This suggests that reducing the total number of solutions may also reduce the number of unique solutions in predictable way. However, further investigation may be required to ascertain this fact.

The number solutions may also have been affected by the query size (i.e. number regions in a query). The size of the query in part determines the number solution match sets created during query evaluation and so it affects the number of potential solutions that may be returned by the query. Figure 15 shows a plot of the number of unique solutions returned by the query against number of regions in the query. Two lines were fitted as shown in the plot. The exponential fit models the points more closely suggesting that the number of solutions may increase exponentially with query size. If this is the case then it may be appropriate to vary the query evaluation methods or parameters over different ranges of query sizes. Again, more data and further statistical analyses are required to verify this observation.

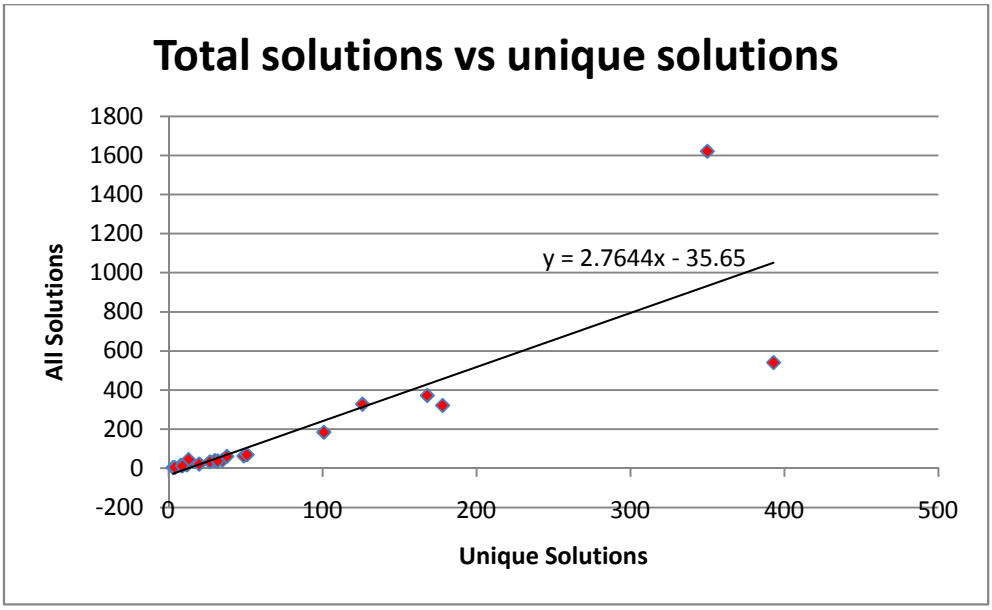


Figure 14: Plot of total number of solutions against the number of unique solutions

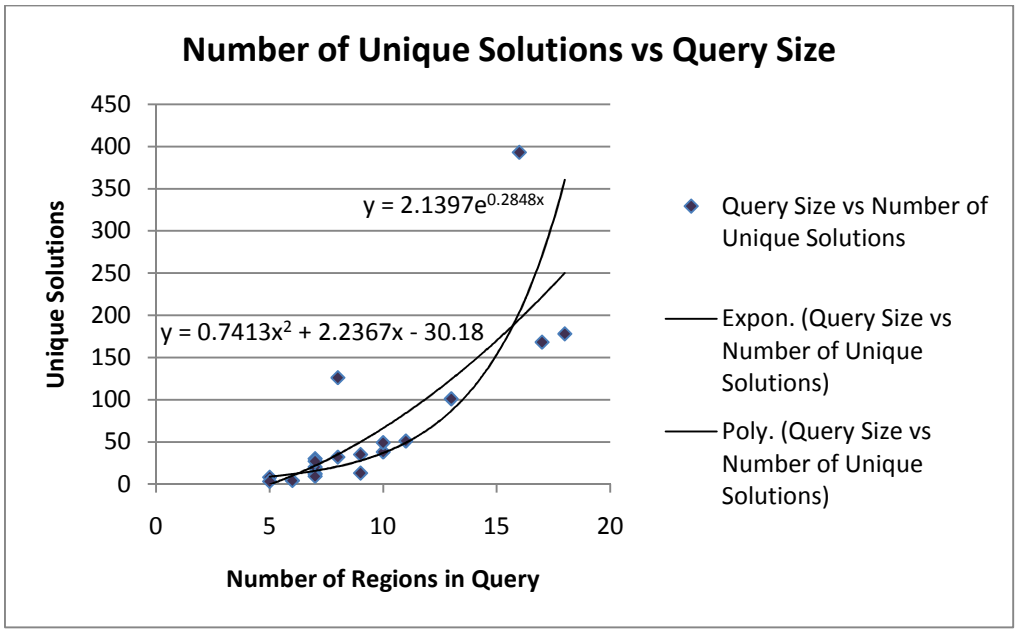


Figure 15: Plot of the number of unique solutions returned by the query against number of regions in the query.

Complete solutions are fewer for each query since they correspond to stricter levels of constraints. This corresponds to the problems of unacceptably large numbers of solutions discussed in the first paragraph. In particular, a scene completeness measure favours complete solutions most since the highest score is attained when all variables are included in the solution. Queries 2_1_3, 2_2_4, and 2_1_5 all returned with two complete solutions each. Each solution is unique because the matching of regions has changed.

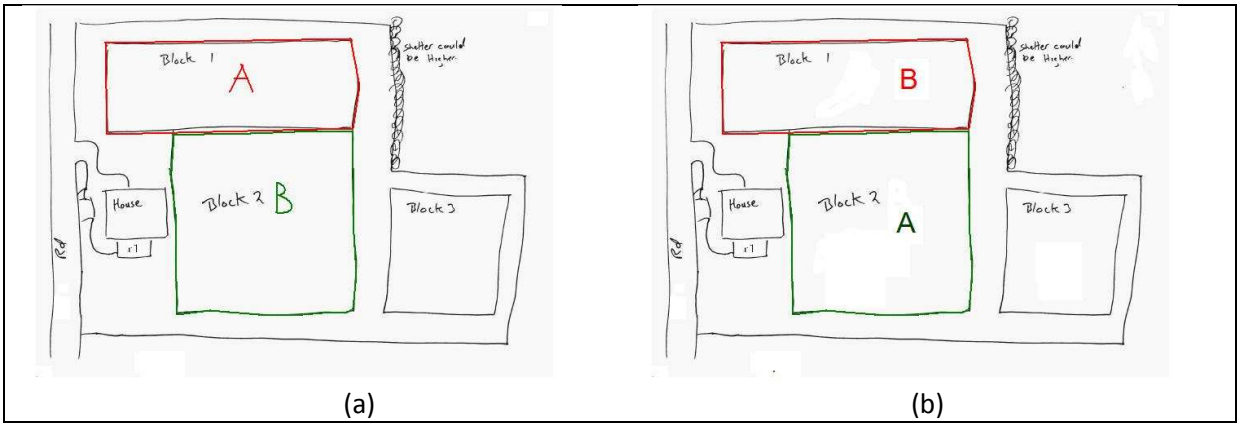


Figure 16: Sketch 2 with two equivalent labelling of regions that lead to ambiguity. Ambiguity of some topological relations is due to symmetry. A and B can both be satisfied by either of the regions Block 1 and Block 2.

In the three cases from the results above multiple complete solutions were a result of swapping of region variable matches due to the ambiguity in the available information. Figure 16 shows two matchings of the pair of regions that led to having two complete solutions in both query 2_1_3 and 2_2_4. By including direction information it is possible to decide with more confidence the intended region. Additional information about other regions in the sketch may also be needed to provide context. For example, in Figure 16, to recover a relation such as B {SouthOf} A from the database, you may need to establish other relations such as House {SouthOf} A since for qualitative direction information there will not be a fixed north and any frame of reference is either local to the referent region or it is local to a specific scenario or task.

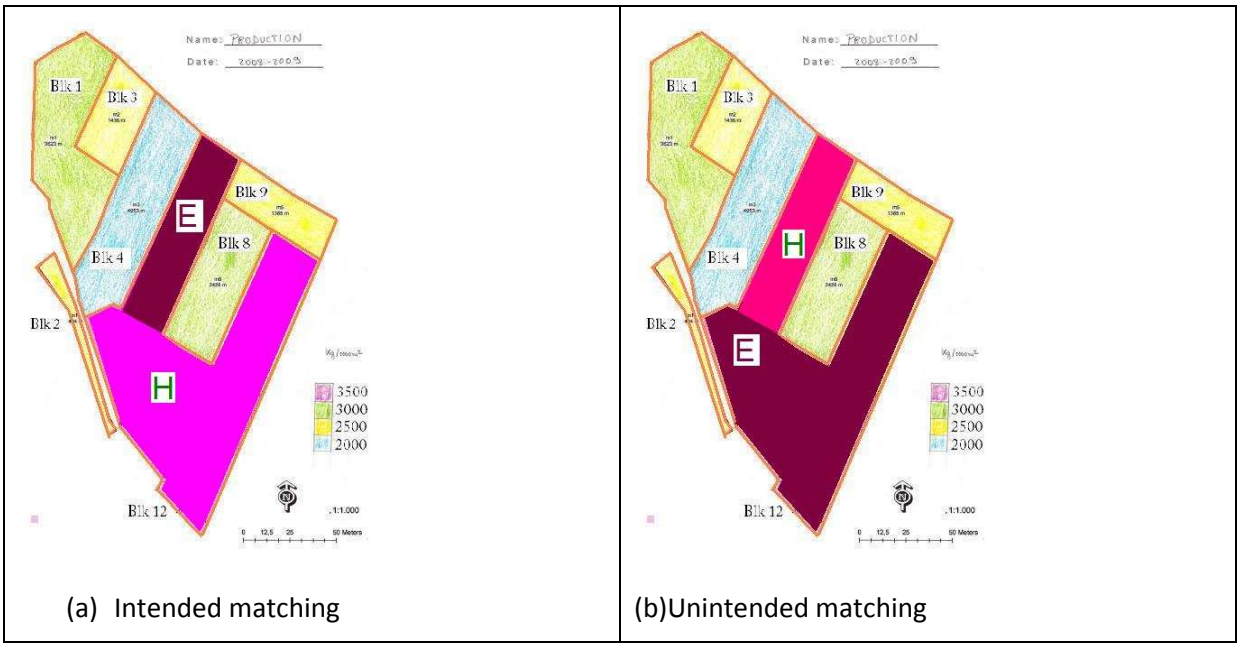
Finally, the results show that there were always solutions where the matches included were exactly those intended in the query. Similarly for complete solutions, at least one of the solutions returned corresponded exactly to the configuration intended in the query.

5.4.2. Aggregated Regions

Deciding when a refinement of the solutions should be attempted was particularly difficult because there is no way of knowing from the topological information alone whether the refinement process will yield a better solution. This is one of the rationales for restricting the scope of the solution to one aggregated region per query. For the single aggregate region case, the heuristic employed to determine when to perform a refinement for any group of solutions is simple (section 4.1) and the procedure used find and test potential parts of the aggregate is short with multiple exit conditions that facilitate early detection of failure. Another advantage of the model used is that candidates for the aggregation are contiguous regions from the same sketch as the aggregate which significantly cuts down the search space and already removes inadmissible solutions. Only one region must be located first and all other regions connected to it will be retrieved and processed when this region is retrieved.

The results from the tests show that in all but one case, if no complete solution was found, then refinement led to at least one new solution. This is true for all cases where regions from the original sketch map were actually aggregated in the query sketch. As in the case in Figure 16 above multiple solutions were a result of ambiguity in the available spatial. Figure 17 below shows two versions of Sketch 4 each marked to represent one of the two refined solutions of queries 3_1_3. For this case too, ambiguity leads to the inability to choose the best solution among two equivalent solutions

from a topological stand point. However, as observed in section 5.4.1 above, direction information was easily used to eliminate the less desirable match.

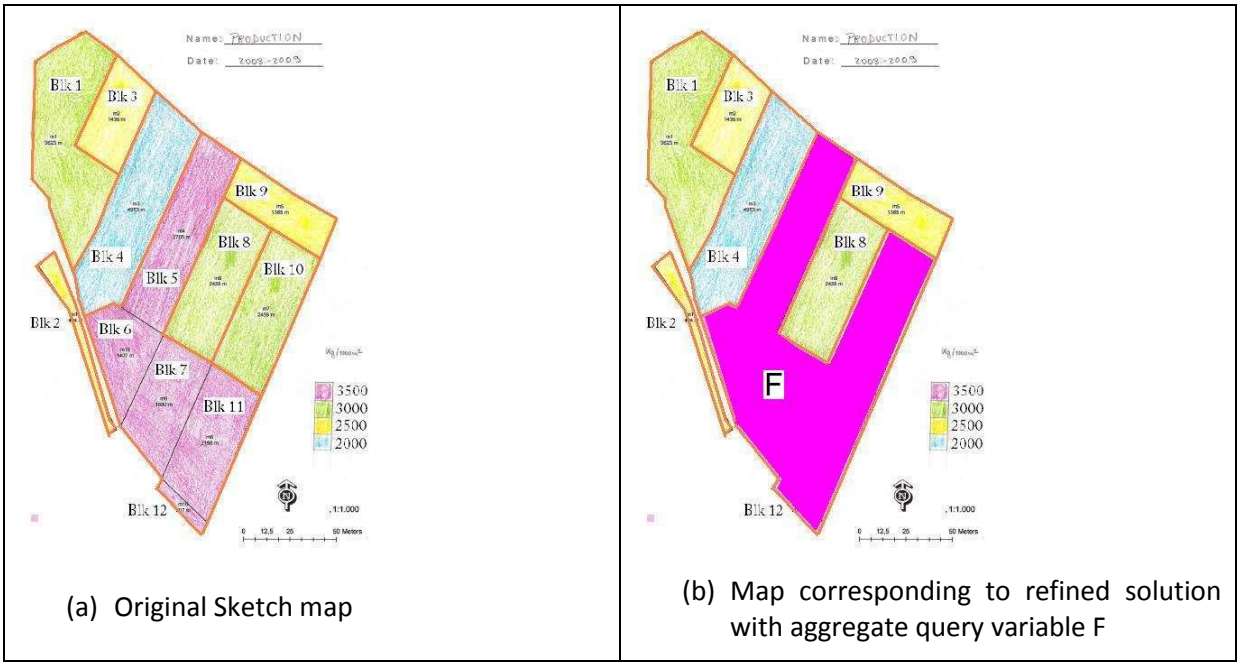


(a) Intended matching

(b)Unintended matching

Figure 17: Ambiguity in original sketches carried on to refined solutions

In Figure 18 the region Blk 10 of the original sketch map is included as part of the aggregate region. Its exclusion from this solution will not change the set of topological relations satisfied by the group as a whole. This signifies the importance of detailed topological relations and detailed direction relations. For example, knowing whether the aggregate touches Blk 8 on one, two, or three side can help to determine in more detail which regions to include as parts of the aggregate. If for instance the aggregate touches Blk 8 only on the same side as it touches Blk 9, then we could eliminate Blk 10 from the aggregate. However, the present model includes every region that does not lead to an inconsistent configuration with respect to the aggregate region.

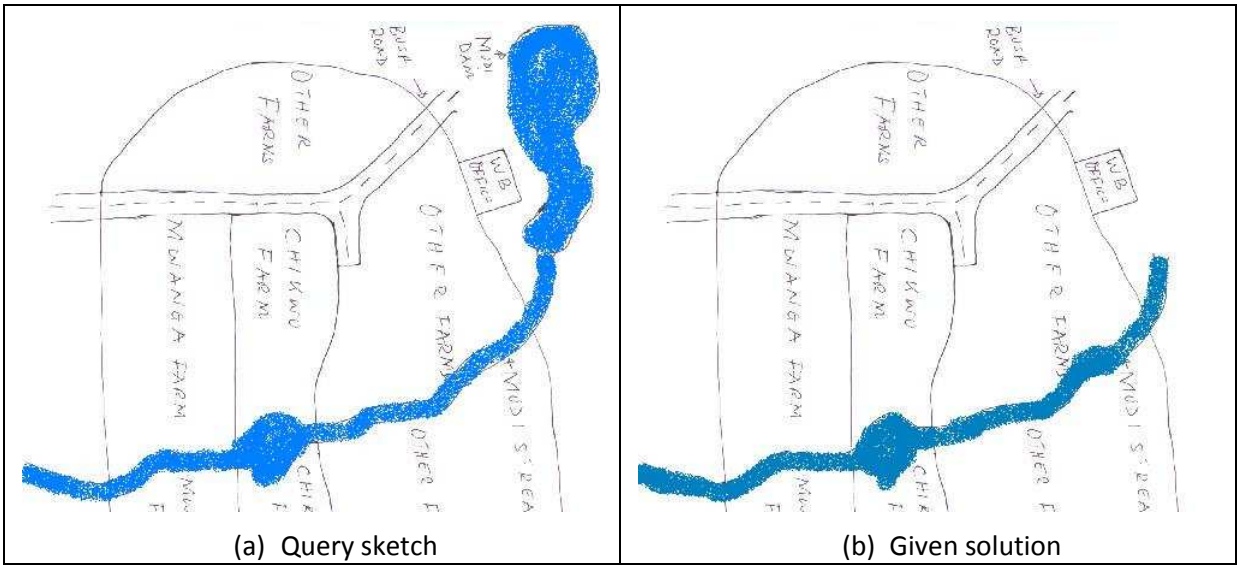


(a) Original Sketch map

(b) Map corresponding to refined solution with aggregate query variable F

Figure 18: Original sketch map of sketch 4 (a) and example of a solution with an aggregate region (b).

One notable drawback of using a simplistic model as the one employed is that there are subtle cases that it may not be able to deal with. An example is query 3_2_6 in which the river and lake are drawn as one feature and only the river is retrieved and the query yields a complete solution (Figure 19). Again, the problem here is context because there is no way of expressing the extents of the sketched regions or their relative positions in terms of distance or direction.



(a) Query sketch (b) Given solution

Figure 19: Situations requiring more information to retrieve better a solution

5.4.3. Returning Results

The purpose of the query system is not to retrieve a single exact match but to find all similar configurations for a given query. But, for sketch maps, each sketch represents specific information about selected objects in a specific environment from a particular perspective (Tversky 2003) and therefore different solutions to the same query should represent a different result for the user. One way of dealing with situations where a large number of solutions are generated from a single sketch is to use similarity measures described by Nedas and Egenhofer (2008) first to rank solutions within the same sketch and include the highest ranking solution only from each sketch in the final results presented to the user.

5.5. Challenges

The biggest challenge faced during this study was implementing the model. Implementation was hampered mostly by a failure to incorporate available tools for reasoning with the RCC model. As indicated at the beginning of Chapter 3, the selected DBMS was PostgreSQL because of its extensibility features (Douglas and Douglas 2003). Two tools that were tested with PostgreSQL are Renz and Nebel's consistency algorithms for the RCC-8 model, which were implemented in the C programming language (Renz and Nebel 2007), and SparQ which is a toolbox of programs (C-Libraries) that implement functions for qualitative spatial representation and reasoning (Wallgrün et al. 2006). For the first tool, one problem was that passing multiple values to a C function is not possible from PostgreSQL. The other problem was that PostgreSQL uses its own memory management functions which continually yielded errors whenever the C functions attempted to access an array on any call subsequent to the first call. In the end this option was dropped. The SparQ reasoner had only been tested on POSIX systems and on the Microsoft Windows Vista system used in this study compilation failed to complete.

The entire implementation has been written in pl/pgsql. A very big challenge that arose from this fact was finding an appropriate structure for organising the software code. Pl/pgsql is poor in data structures and implementing complex algorithms such as the clique enumeration algorithm was very difficult. The lack of data constructs such as structs and pointers in the C language or classes in object-oriented programming languages meant that values associated in any way have to always be accessed directly from a database table. This has led to a great number of SQL queries being written for the internal functions of the model. Because of the dynamic nature of the system, the database server has to make frequent disk input/output operations since the query planner cannot easily predict the data required for any subsequent queries even if they are in the same transaction (Douglas and Douglas 2003, pp. 167-208). Global variables are not supported so that all data required across several functions have to be passed explicitly. As a result, in the implementation, most global data is dumped into tables where other functions can find it. In a multiuser environment this is dangerous since one user may place data in a table and another user may alter that information in between transaction. However all data passed through tables are within transaction data and therefore protected by the DBMS transaction management system.

The way unary constraints are modelled and specified in a query was one of the most difficult design decisions to be made because there was a need to maintain flexibility in terms of what type of information can be modelled alongside the topological information in the database. As seen in Chapter 3, unary constraints of the query are normal relational attributes and the model does not impose any specific structure on how they should be modelled in the database. This leads to non-uniform treatment of unary constraints during query evaluation since each unary constraint is a user specified SQL statement. For binary constraints, only the basic RCC-8 relations are stored so that a relation to be stored in the database cannot be derived from composition since usually the result of composition is a disjunction of relations rather than a single basic relation. This reduces the flexibility of using the system as incomplete information cannot be used as input.

Finding test data, generating test cases, and executing the tests was also a challenge because data were not readily available and there was clear way to involve real users since no graphical user interface was available for sketching and presentation. In addition, manual extraction of topological information in the sketch and encoding of the information into a query was tedious and error-prone. This made the testing slow and difficult as every information had to be checked at least twice.

6. Conclusions and Future Work

6.1. Conclusions

It is well known that topology alone is not sufficient as a query evaluation criterion for spatial queries (Egenhofer and Mark 1995, Egenhofer 1997) but it is, nonetheless, essential. The model presented in this work is able to compare the topological relations among regions in one sketch map with those in a database and retrieve the all configurations from the database that match the sketch map completely or partially.

Symmetric topological relations (DC, EC, PO, and EQ) can introduce ambiguity if no other information about the spatial relations among the regions concerned is available. Considering the results from the tests described above, in all cases there were many solutions per query but whenever more than one complete solution was encountered it was possible to choose from them the exact match for the query by considering the direction relations among the regions. From this it is evident that the inclusion of more binary constraints on the pairs of regions (e.g. direction, distance, and shape constraints) in the query would reduce the number of admissible results with a positive effect. Limiting the size of the solutions by some threshold will also help to reduce the size of the set of solutions and remove some redundancies as subsets of larger solutions will be excluded. For databases with a large number of sketches, this must be done on a per sketch basis since sketches with fewer regions will always suffer from high thresholds.

Refining solutions by aggregating database regions and comparing them to an unmatched query variable may improve the results returned by the query if thresholds are imposed for acceptable sizes of a solution. In fact refinement has a small counter effect to the effects of a threshold because previously unacceptable solutions may be moved into the acceptable range by incorporating more regions. But the aggregation model will not work in cases where the query is larger than every sketch in the database. For this, new ways of deciding when and how to apply the aggregation procedure will be needed. The procedure is nevertheless an appropriate tool for processing queries to databases of sketch maps because in such cases solutions that match the query sketch completely are preferred to those that are not complete.

While the model was developed with sketch maps in mind, it can easily be extended for use with geospatial databases since most of them would be expected to support topological and directional relation operators (Güting 1994). The refinement model can also be employed independently of the entire system developed in this study because it only requires as input a set of regions each with some specified topological constraints with a fixed region and another distinct set of regions that may satisfy the constraints between the fixed region and regions of the first set.

6.2. Future Work

The refinement model developed in this thesis is limited in several respects, as presented in Chapters 4 and 5. These limitations were a result of the complexity associated with developing a general solution to the problem. The generalisation of the present solution to include any number aggregate regions and allowing disconnected regions to form parts of an aggregate presents an opportunity for future work at two levels. First, the analysis of the query needs to be such that the aggregation procedure is entered only if a certain level of certainty about the presence of an

aggregate region in the query is reached. Otherwise, the procedure may be executed unnecessarily many times without a positive result. The second level involves developing heuristics to select and compare database regions for inclusion in an aggregate region. A hierarchical process could be useful in this scenario since it allows comparisons to be made locally first and then expand to neighbouring regions. The inclusion of direction information in the decision model of the procedure may also become advantageous to at this level.

Much work has already been done about incorporating different aspects of spatial representation into query by sketch systems (Egenhofer 1997, Egenhofer and Shariff 1998, Forbus et al. 2003, Nedas and Egenhofer 2008). Although most of this work is aimed at query systems for spatial databases, it can be equally applicable to databases of sketch maps but this assertion needs to be tested. An implementation of a database model that allows the combination of different sketches at the abstract level of qualitative spatial representations still remain an open question as far as sketch-based query systems are concerned. In the context of the present model, this means that a complete solution to a single query could be created from partial solutions from two different sketches in the database. Present work in this direction includes that of Li et al. (2009) in which they apply consistency checking for a combined CSP of the Interval Algebra.

7. Bibliographic References

Aragó, P., Molin, P., Site-specific farming strategies for developing countries, Seminar Paper, GIS for Developing Countries, Institute for Geoinformatics, Münster, July 2009.

Bennett, B., Encyclopedia of Geographic Information Science, in: K. K. Kemp (ed.), pp 426-432, Sage, 2008.

Bennett, B., Logics for Topological Reasoning, Lecture Notes, ESSLLI Summer SChool, University of Birmingham, August 2000.

Bennett, B. Isli, A. Cohn, G. A System Handling RCC-8 Queries on 2D Regions Representable in the Closure Algebra of Half-Planes. *Proceedings of the 11th International Conference on Industrial and Engineering Applications of Artificial Intelligence and Expert Systems (IEA-AIE)*, LNCS, Springer-Verlag, 1998.

Brassel, K., Weibel, R., A review and framework of automated map generalization, International Journal of Geographical Information Systems, 2(3), pp. 229–244, 1988.

Bron, C., Kerbosch, J., Algorithm 457: finding all cliques of an undirected graph. Comm. ACM, 16(9), pp. 575-577, 1973.

Cazals , F., Karande , C., A note on the problem of reporting maximal cliques, January 30, 2008.

Cohn, A. G., Bennett, B., Gooday, J. and Gotts, N., Qualitative Spatial Representation and Reasoning with the Region Connection Calculus, *Geoinformatica*, 1, pp. 1-44, 1997.

Douglas, K., Douglas, S., PostgreSQL, Sam Publishing, Indianapolis, USA. 2003.

Cohn, A. G., Renz, J., Qualitative Spatial Representation and Reasoning, in: F. van Hermelen, V. Lifschitz, B. Porter (eds.), *Handbook of Knowledge Representation*, Elsevier, pp. 551-596, 2007.

Egenhofer, J. , Query Processing in Spatial-Query-by-Sketch, Journal of Visual Languages and Computing, 8(4), pp. 403-424, 1997.

Egenhofer, M., Mark, D., Naive Geography. In: A. Frank and W. Kuhn (eds.), Lecture Notes in Computer Science, Vol. 988, Springer-Verlag, pp. 1-15, September 1995.

Egenhofer, M.J., Shariff, R.B., Metric Details for Natural-Language Spatial Relations, ACM Transactions on Information Systems, Vol. 16, pp. 295-321, 1998.

Egenhofer, M.J., Franzosa, R.D., Point-Set Topological Spatial Relations, International Journal of Geographical Information Systems, 1991

Egenhofer, M., Herring, J., A mathematical framework for the definition of topological relationships, *Proceedings of Fourth International Symposium on Spatial Data Handling*, Zurich, pp. 803-813, 1990.

Fehling, R., Nebel, B., Renz, J., rcc8op.c, a module from the experimental software for testing consistency of sets rcc8 relations. Institut für Informatik, Albert-Ludwigs-Universität, Freiburg, Germany. 1998.

Forbus, K., Usher, J., Chapman, V., Qualitative Spatial Reasoning about Sketch Maps. *Proceedings of the Fifteenth Annual Conference on Innovative Applications of Artificial Intelligence*, Acapulco, Mexico, 2003.

Goodchild, M. F., Citizens as sensors: the world of volunteered geography, *GeoJournal*, 69, pp. 211–221, 2007.

Goyal, R., Egenhofer, M.J., Similarity of Cardinal Directions, Lecture Notes in Computer Science, 2001.

Güting, R.H., An Introduction to Spatial Database Systems, Special Issue on Spatial Database Systems, VLDB Journal, 3(4), 1994.

Indulska, M., Orlowska, M.E., On Aggregation Issues in Spatial Data Management, Australian Computer Society, 2002.

Koch, I., Fundamental study: Enumerating all connected maximal common subgraphs in two graphs. Theoretical Comp. Sc., 250(1-2):1–30, 2001.

Kumar, V., Algorithms for constraint-satisfaction problems: a survey, AI Magazine 13(1), pp. 32-44, 1992.

Li, J. J., Huang, J., Renz, J., A Divide-and-Conquer Approach for Solving Interval Algebra Networks. *Proceedings of the Twenty-first International Joint Conference on Artificial Intelligence (IJCAI'09)*, Pasadena, CA, July 2009

Liu, W., Li, S., Renz, J., Combining RCC-8 with Qualitative Direction Calculi: Algorithms and Complexity. *Proceedings of the Twenty-first International Joint Conference on Artificial Intelligence (IJCAI'09)*, Pasadena, CA, July 2009.

Nedas K.A., Egenhofer M.J., Spatial Scene Similarity Queries, Transactions in GIS 12(6), 2008.

Papadias, D., Sellis, T., The Semantics Of Relations In 2D Space Using Representative Points: Spatial Indexes. European Conference on Spatial Information Theory (COSIT '93), Elba, Italy. Springer-Verlag, 1993.

Randell, D. A., Cui, Z., Cohn, A. G., A Spatial Logic based on Regions and Connections. *Conference on Knowledge Representation and Reasoning*, 1992.

Read, M., Hunt, L., Fairweather, J., Sketch Maps: Features and Issues Important for the Management of ARGOS Orchards and Farms, ARGOS Research Report 05/10, August 2005.

Renz, J., Nebel, B., Qualitative Spatial Reasoning using Constraint Calculi, in: M. Aiello, I. Pratt-Hartmann, J. van Benthem, eds., *Handbook of Spatial Logics*, Springer Verlag, Berlin, 161-215, 2007.

Stocker, M., Sirin, E., PelletSpatial: A Hybrid RCC-8 and RDF/OWL Reasoning and Query Engine. *Proceedings of the 5th International Workshop on OWL: Experiences and Directions (OWLED)*, 2009.

Tomita, E. , Tanaka, A. , Takahashi, H., The worst-case time complexity for generating all maximal cliques and computational experiments. Theoretical Computer Science, 363(1), pp. 28–42, 2006

Tryfona, N., Egenhofer, M.J., Consistency among Parts and Aggregates: A Computational Model Transactions in GIS 1(3), 1997.

Tversky, B., Navigating by mind and by body. In C. Freksa, W. Brauer, C. Habel, K. F. Wender (Eds), Spatial Cognition III: Routes and Navigation, Human Memory and Learning, Spatial Representation and Spatial Reasoning, pp.1-10, Berlin: Springer Verlag, 2003.

Tversky, B., What do Sketches say about Thinking?, 2002.

Tversky, B., Cognitive Maps, Cognitive Collages, and Spatial Mental Models. In Frank, A.U., and Campari, I. (Eds.) Spatial Information Theory: A Theoretical Basis for GIS, Proceedings COSIT '93. Lecture Notes in Computer Science, 716, pp.14-24, Springer: Berlin, 1993.

Tversky, A., Features of similarity, Psychological Review, 84(4), pp. 327-352, 1977.

van Beek, P., Reasoning about Qualitative Temporal Information, Artificial Intelligence, 58(1-3), pp. 271-321, 1992.

Wang, J. How Human Schematization and Systematic Errors Take Effect on Sketch Map Formalizations, Master Thesis, Institute for Geoinformatics, University of Munster, 2009.

Wallgrün, J. O., Frommberger, L., Dylla, F., Wolter, D., SparQ User Manual V0.7. SFB/TR 8 Spatial Cognition - Project R3-[Q-Shape], 2009.

8. Annexes

Annex A: Input Sketches

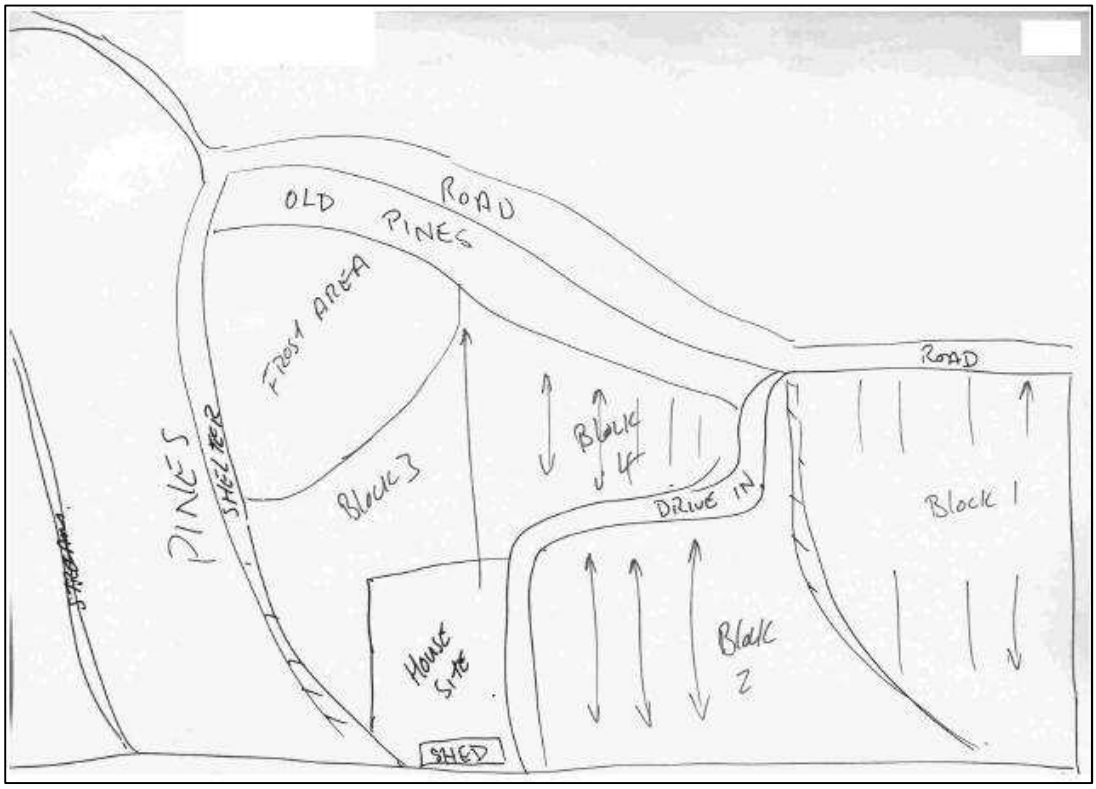


Figure A1: Sketch 1

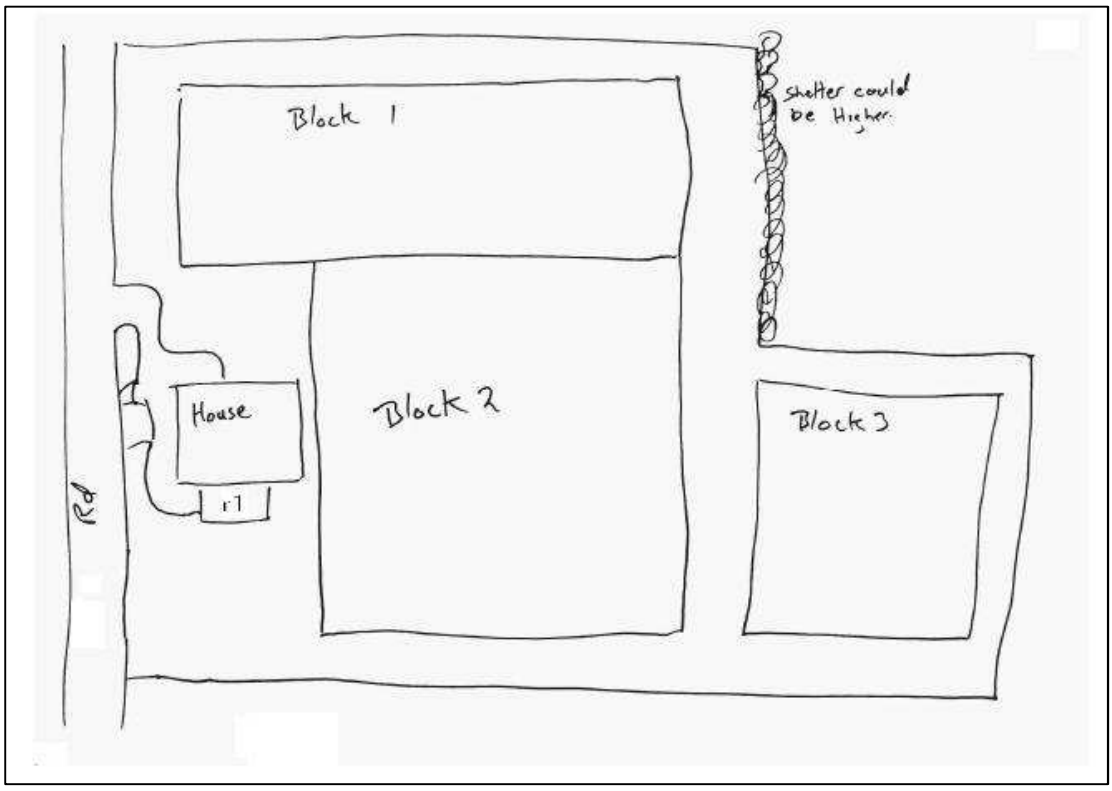


Figure A2: Sketch 2

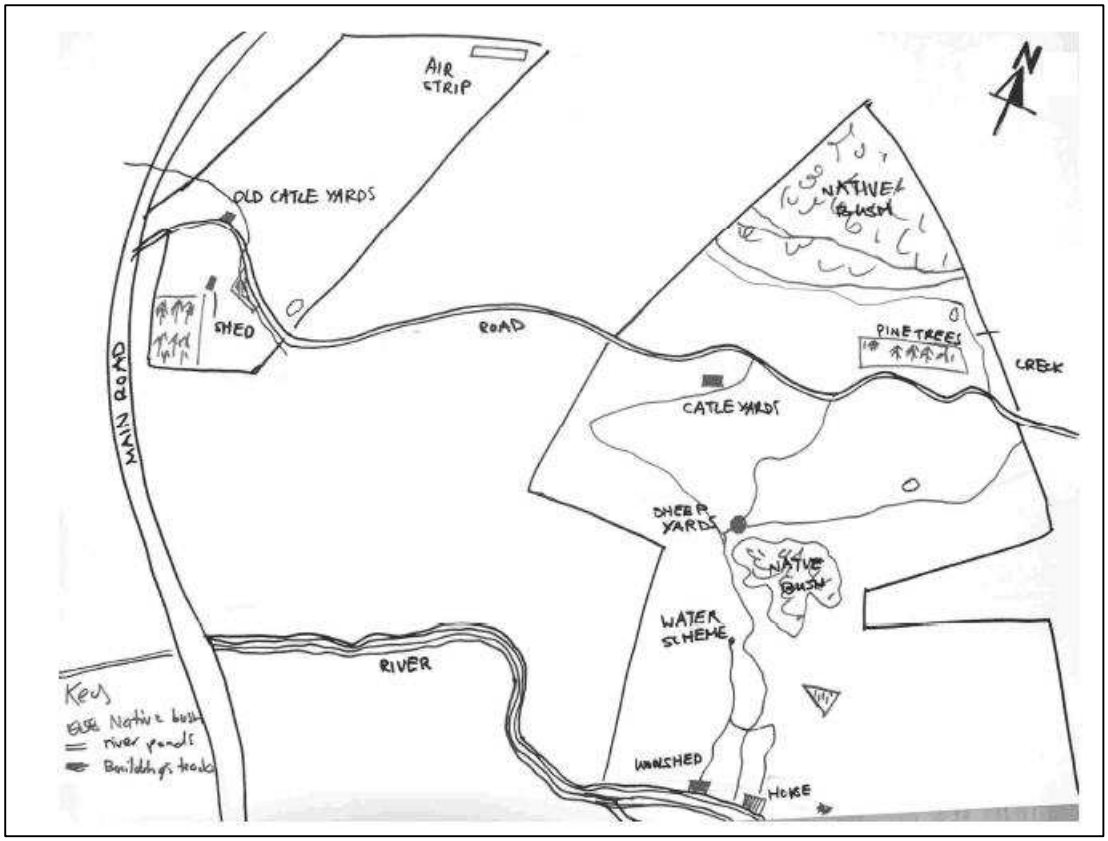


Figure A3: Sketch 3

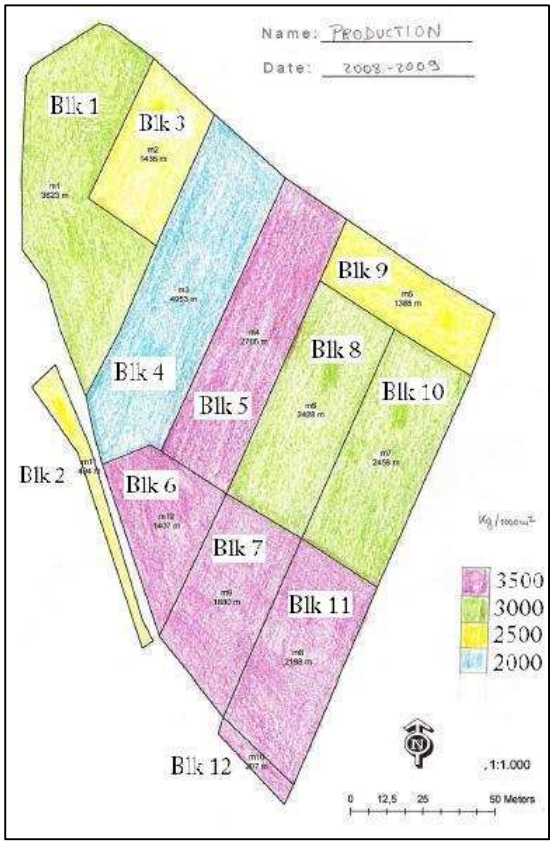


Figure A4: Sketch 4

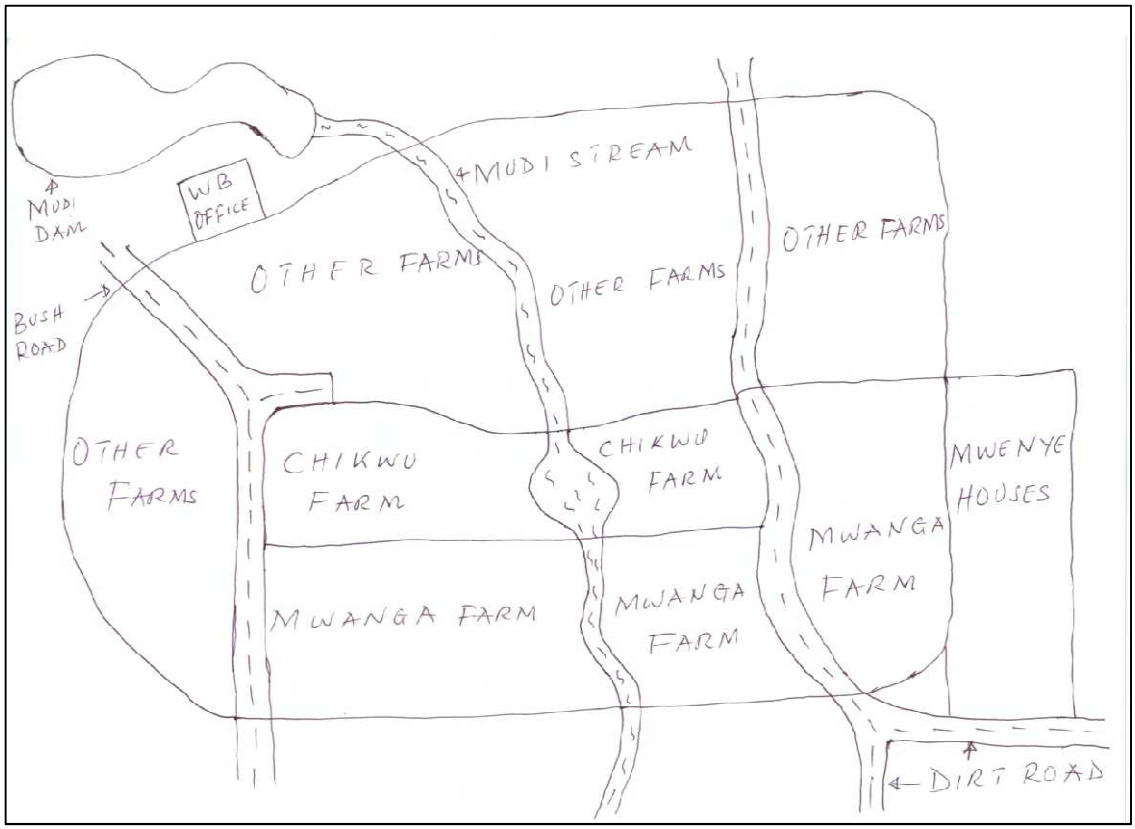


Figure A5: Sketch 5

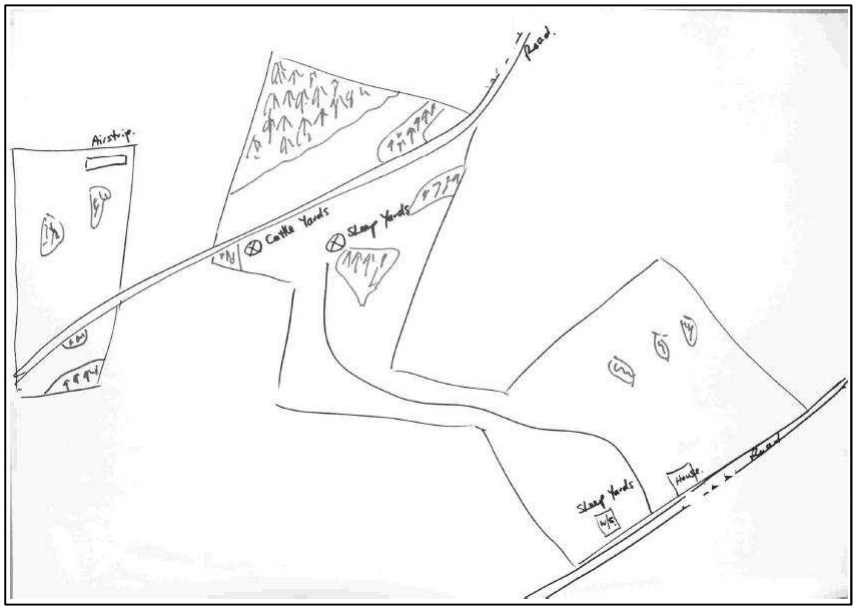


Figure A6: Sketch 6

Annex B: Sample scripts

```
DROP TABLE IF EXISTS farms CASCADE;

CREATE TABLE farms(
       feature_id serial,
       name text,
       description text,
       the_rcc_region varchar(48) UNIQUE
);


DROP TABLE IF EXISTS farm_properties CASCADE;

CREATE TABLE farm_properties (
       property_id serial,
       feature varchar,
       name text,
       value text,
       CONSTRAINT fk_feat FOREIGN KEY (feature) REFERENCES farms
(the_rcc_region) ON UPDATE CASCADE ON DELETE CASCADE
);

SELECT set_rcc_table('farms', 'the_rcc_region');
```

Listing A1: Script for creating tables that were used to test the database model

```
SELECT use_sketch('KiwiFarm 1');

INSERT INTO farms(name, description, the_rcc_region) VALUES ('Road', 'Main road
passing near farm', 'S1Road');

INSERT INTO farm_properties(feature, name, value) VALUES ('S1Road', 'Category',
'transport');
INSERT INTO farm_properties(feature, name, value) VALUES ('S1Road', 'Feature',
'roads');

INSERT INTO farms(name, description, the_rcc_region) VALUES ('Old Pines', 'Old pine
tree lot', 'S1OldPines');

INSERT INTO farm_properties(feature, name, value) VALUES ('S1OldPines', 'Category',
'other biota based activities');
INSERT INTO farm_properties(feature, name, value) VALUES ('S1OldPines', 'Feature',
'pine trees');

INSERT INTO farms(name, description, the_rcc_region) VALUES ('Pines', 'Pine tree
lot', 'S1Pines');

INSERT INTO farm_properties(feature, name, value) VALUES ('S1Pines', 'Category',
'other biota based activities');
INSERT INTO farm_properties(feature, name, value) VALUES ('S1Pines', 'Feature', 'pine
trees');

INSERT INTO farms(name, description, the_rcc_region) VALUES ('Frost Area', 'Frost
Area', 'S1FrostArea');

INSERT INTO farm_properties(feature, name, value) VALUES ('S1FrostArea', 'Category',
'climate');
INSERT INTO farm_properties(feature, name, value) VALUES ('S1FrostArea', 'Feature',
'frost area');
```

Listing A2: Script for inserting data from sketch 3 into database tables

Listing A2: Continued…

```
INSERT INTO farms(name, description, the_rcc_region) VALUES ('Block 1', 'Farm division
block 1', 'S1Block1');

INSERT INTO farm_properties(feature, name, value) VALUES ('S1Block1', 'Category',
'spatial organisation');
INSERT INTO farm_properties(feature, name, value) VALUES ('S1Block1', 'Feature',
'blocks');

INSERT INTO farms(name, description, the_rcc_region) VALUES ('Block 2', 'Farm division
block 2', 'S1Block2');

INSERT INTO farm_properties(feature, name, value) VALUES ('S1Block2', 'Category',
'spatial organisation');
INSERT INTO farm_properties(feature, name, value) VALUES ('S1Block2', 'Feature',
'blocks');

INSERT INTO farms(name, description, the_rcc_region) VALUES ('Block 3', 'Farm division
block 3', 'S1Block3');

INSERT INTO farm_properties(feature, name, value) VALUES ('S1Block3', 'Category',
'spatial organisation');
INSERT INTO farm_properties(feature, name, value) VALUES ('S1Block3', 'Feature',
'blocks');

INSERT INTO farms(name, description, the_rcc_region) VALUES ('Block 4', 'Farm division
block 4', 'S1Block4');

INSERT INTO farm_properties(feature, name, value) VALUES ('S1Block4', 'Category',
'spatial organisation');
INSERT INTO farm_properties(feature, name, value) VALUES ('S1Block4', 'Feature',
'blocks');

INSERT INTO farms(name, description, the_rcc_region) VALUES ('House Site', 'Farm house
yard or lot', 'S1HouseSite');

INSERT INTO farm_properties(feature, name, value) VALUES ('S1HouseSite', 'Category',
'spatial organisation');
INSERT INTO farm_properties(feature, name, value) VALUES ('S1HouseSite', 'Category',
'buildings');
INSERT INTO farm_properties(feature, name, value) VALUES ('S1HouseSite', 'Feature',
'blocks');
INSERT INTO farm_properties(feature, name, value) VALUES ('S1HouseSite', 'Feature',
'houses');

INSERT INTO farms(name, description, the_rcc_region) VALUES ('Shed', 'Farm storage and
work shed', 'S1Shed');

INSERT INTO farm_properties(feature, name, value) VALUES ('S1Shed', 'Category',
'buildings');
INSERT INTO farm_properties(feature, name, value) VALUES ('S1Shed', 'Feature', 'sheds');

INSERT INTO farms(name, description, the_rcc_region) VALUES ('Shelter', 'shelter',
'S1Shelter');

INSERT INTO farm_properties(feature, name, value) VALUES ('S1Shelter', 'Category',
'wind');
INSERT INTO farm_properties(feature, name, value) VALUES ('S1Shelter', 'Feature',
'shelters');

INSERT INTO farms(name, description, the_rcc_region) VALUES ('Block 2-1 Boundary', 'Farm
division boundary between block 2 and block 1', 'S1Block21Boundary');

INSERT INTO farm_properties(feature, name, value) VALUES ('S1Block21Boundary',
'Category', 'spatial organisation');
INSERT INTO farm_properties(feature, name, value) VALUES ('S1Block21Boundary',
'Feature', 'blocks');
INSERT INTO farm_properties(feature, name, value) VALUES ('S1Block21Boundary',
'Feature', 'boundaries');
```

Listing A2: Continued…

```
INSERT INTO farms(name, description, the_rcc_region) VALUES ('Drive In', 'Drive way from
the main road to the farm', 'S1DriveIn');

INSERT INTO farm_properties(feature, name, value) VALUES ('S1DriveIn', 'Category',
'transport');
INSERT INTO farm_properties(feature, name, value) VALUES ('S1DriveIn', 'Feature',
'driveways');

SELECT end_sketch();

SELECT related_regions('S1Road', 'S1OldPines',  'EC');

SELECT related_regions('S1Road', 'S1Pines', 'EC');

SELECT related_regions('S1Road', 'S1Block1', 'EC');

SELECT related_regions('S1Road', 'S1FrostArea', 'EC');

SELECT related_regions('S1Road', 'S1Shelter', 'EC');

SELECT related_regions('S1Road', 'S1Block3', 'EC');

SELECT related_regions('S1Road', 'S1DriveIn', 'EC');

SELECT related_regions('S1Road', 'S1Block21Boundary', 'EC');

SELECT related_regions('S1Pines', 'S1Shelter', 'EC');

SELECT related_regions('S1OldPines', 'S1FrostArea', 'EC');

SELECT related_regions('S1OldPines', 'S1Block4', 'EC');

SELECT related_regions('S1OldPines', 'S1DriveIn', 'EC');

SELECT related_regions('S1FrostArea', 'S1Block4', 'EC');

SELECT related_regions('S1FrostArea', 'S1Block3', 'EC');

SELECT related_regions('S1Block3', 'S1Block4', 'EC');

SELECT related_regions('S1Block3', 'S1HouseSite', 'EC');

SELECT related_regions('S1Block3', 'S1Shelter', 'EC');

SELECT related_regions('S1Block4', 'S1HouseSite', 'EC');

SELECT related_regions('S1Block4', 'S1DriveIn', 'EC');

SELECT related_regions('S1HouseSite', 'S1Shelter', 'EC');

SELECT related_regions('S1HouseSite', 'S1DriveIn', 'EC');

SELECT related_regions('S1HouseSite', 'S1Shed', 'NTPPI');

SELECT related_regions('S1DriveIn', 'S1Block2', 'EC');

SELECT related_regions('S1Block2', 'S1Block21Boundary', 'EC');

SELECT related_regions('S1Block1', 'S1Block21Boundary', 'EC');
```

```
SELECT rcc_clear_qry();

SELECT unary_qry_constraint('A', 'farms ', 'select a.the_rcc_region as the_rcc_region from farms a, farm_properties b
            where a.the_rcc_region = b.feature and (((b.name = ''Feature'' and b.value = ''roads'')
                  and (a.description ~* ''main road'' or a.name ~* ''road'')))
');

SELECT unary_qry_constraint('B', 'farms ', 'select a.the_rcc_region as the_rcc_region from farms a, farm_properties b
            where a.the_rcc_region = b.feature and (((b.name = ''Feature'' and b.value = ''pine trees'')
                  and (a.description ~* ''pine trees'' or a.name ~* ''old pine trees'' or a.name ~* ''old pines'')))
');

SELECT unary_qry_constraint('C', 'farms ', 'select a.the_rcc_region as the_rcc_region from farms a, farm_properties b
            where a.the_rcc_region = b.feature and (((b.name = ''Feature'' and b.value = ''driveways'')
                  and (a.description ~* ''drive way'' or a.name ~* ''drive way'' or a.name ~* ''driveway''
                        or a.name ~* ''drive in'' or a.name ~* ''drivein'')))
');

SELECT unary_qry_constraint('D', 'farms ', 'select a.the_rcc_region as the_rcc_region from farms a, farm_properties b
            where a.the_rcc_region = b.feature and (((b.name = ''Feature'' and b.value = ''houses'')
                  and (a.description ~* ''farm house'' or a.name ~* ''house'' or a.name ~* ''farm house'')))
');

SELECT unary_qry_constraint('E', 'farms ', 'select a.the_rcc_region as the_rcc_region from farms a, farm_properties b
            where a.the_rcc_region = b.feature and (
                      ((b.name = ''Feature'' and b.value = ''shelters'') and (a.description ~* ''shelter'' or a.name ~* ''shelter'')))
');

SELECT unary_qry_constraint('F', 'farms ', 'select distinct a.the_rcc_region as the_rcc_region from farms a, farm_properties b
            where a.the_rcc_region = b.feature
');

SELECT unary_qry_constraint('G', 'farms ', 'select a.the_rcc_region as the_rcc_region from farms a, farm_properties b
            where a.the_rcc_region = b.feature and (((b.name = ''Feature'' and b.value = ''frost area'')
                  and (a.description ~* ''frost area'' or a.name ~* ''frost area'')) )
');

SELECT binary_qry_constraints('A B EC AND A C EC AND A D DC AND A E EC AND A F EC AND A G EC AND B C EC AND B D DC AND B E DC AND B F EC
AND B G EC AND C D EC AND C E DC AND C F EC AND C G DC AND D E EC AND D F EC AND D G DC AND E F EC AND E G DC AND F G EC');

SELECT rcc_eval_qry(false);      SELECT refine_solutions();

SELECT * FROM rcc_return_next(5);       SELECT * FROM rcc_return_unique(7);
```

Listing A3: Script for query 2_1_1 whose sketch was based on sketch 3 of the original sketches in Annex A.

MASTERS PROGRAM IN

GEOSPATIAL TECHNOLOGIES

Dissertation submitted in partial fulfilment of the requirements
for the Degree of *Master of Science in Geospatial Technologies*

UNIVERSIDADE NOVA DE LISBOA

UNIVERSITAT JAUME·I

WESTFÄLISCHE WILHELMS-UNIVERSITÄT MÜNSTER

MASTERS PROGRAM IN

GEOSPATIAL TECHNOLOGIES

Supported by:

Education and Culture

ERASMUS MUNDUS