CHOROS 2.0: Improving the Performance of Spatial Reasoning in OWL

Mainas Nikolaos

Department of Electronic and Computer Engineering
Technical University of Crete

Dissertation Thesis Committee:
Euripides G.M Petrakis, Associate Professor (Supervisor)
Stavros Christodoulakis, Professor
Michail G. Lagoudakis, Associate Professor

November 2, 2013

Acknowledgments

This thesis would not have been possible without the help of several people who contributed in the preparation and completion of this study.

Firstly and foremost I would like to thank my advisor, Professor Euripides G. M. Petrakis for his constant supervision. For guiding, advising and supporting me in every step of this thesis. I am grateful for giving me the opportunity to work on this very interesting field of research.

Also, I would like to thank Professor Stavros Christodoulakis and Associate Professor Michail G. Lagoudakis who agreed to evaluate my diploma thesis.

Moreover, I would like to thank my laboratory colleagues for their patient and constructive comments.

I would like to thank all my friends for these great years we spent together and for the many wonderful memories.

Most of all, I would like to thank my family for their enormous help, understanding and support all these years.

Abstract

In this thesis, we investigate on potential improvements to CHOROS v1, a qualitative spatial reasoning engine for qualitative topologic and directional spatial information. CHROROS v1 reasoner is implemented in Java and supports consistency checking and query answering over spatial data represented with the Region Connection Calculus (RCC) and the Cone-Shaped directional logic formalism (CSD) as well as standard RDF/OWL semantic relations, both represented in RDF/OWL. Our implementation is referred to as CHOROS 2.0 and inherits all features of CHOROS v1 and suggests certain improvements to the basic path-consistency algorithm enhancing its run-time performance. In addition CHOROS 2.0 handles information referring to the relevant position of objects expressed in numerical values, and also updates the ontology with inferred facts. These features are not supported by its predecessor. Finally, we discuss on further improvements to the reasoning engine (i.e., based on reducing the size of compositions of basic relations) and we compare their performance to that of CHOROS v1 as well as to that of SOWL, a spatial reasoner implemented in SWRL and runs under Protégé.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

The increasing use of the World Wide Web (WWW) in the recent years has generated additional interest for new intelligent mechanisms and applications, capable of handling tasks that are typically handled manually by users. For example, searching and buying a product on the Web requires careful selection among different products that satisfy user needs, at the best available price. All these tasks are handled by using a search engine and by manually browsing, selecting and comparing products in Web pages. In recent years, there is an increasing need for Web services that accomplish these tasks automatically without requiring user intervention, besides task description. These applications must be able to comprehend the meaning of the content of Web pages and reason over their content similarly to the way humans do. Semantic Web is a solution to this need, that is realized with the aid of specific tools capable of representing the meaning of Web pages as well as, for reasoning and querying over their content. Ontologies provide the means for representing high-level concepts, their properties and interrelationships. More specifically, ontologies contain definitions of concepts and of their properties by means of binary relations between concepts or a concept and a numerical (concrete) domain.

Reasoning is the process by which implicitly logical conclusions are extracted from an explicit set of facts or statements, which form the knowledge base.

Spatial information is an important aspect of Web content. It is inherent to the content of Web pages using either qualitative or quantitative descriptions (e.g., position coordinates, distances). In this work, we present CHOROS 2.0 reasoner which supports consistency checking over qualitative spatial information in OWL (i.e., using natural language terms for expressing locations in space in terms of their relations to other objects especially when exact locations are unknown) using the Region Connection Calculus (RCC) or the Cone-Shaped Directional logic formalism (CSD).

1.1 Background

Formal spatial representations have been studied extensively in the Database [5] and recently, the Semantic Web literature [6]. Spatial entities (e.g., objects, regions) in classic database systems are represented using points, lines (polygonal lines) or Minimum Bounding Rectangles (MBRs) enclosing objects or regions and their relationships. Relations among spatial entities can be topological, orientation or distance-based relations. Furthermore, spatial relations are distinguished into qualitative (i.e., relations described using lexical terms such as "Into", "South" etc.) and quantitative (i.e., relations described using numerical values such as "10 Km away", "45 degrees North" etc.). The motivation for using a qualitative approach is that it is considered to be closer to the way humans represent and reason about common sense knowledge. Another motivation is that it is possible to deal with incomplete knowledge.

Accordingly, spatial ontologies are defined based-upon a reference coordinate system in conjunction with a set of qualitative topological and directional relations (e.g., RCC-8 relations). A spatial ontology can also be defined without a coordinate reference system, such as in the case of topological relations [1]. Nevertheless, it is not always possible to directly encode the semantics of these relations using the expressivity of OWL and Description Logics (DL) that OWL is based on [2]. There might be inconsistencies within a set of spatial relations that will not be detected by an OWL reasoner or, an OWL reasoner might not compute all spatial inferences. To deal with this problem, reasoning rules for various relation sets have been proposed [3, 4].

In our previous work, we introduced CHOROS reasoner [7] (referred henceforth as CHOROS 1.0) which supports consistency checking and query answering over spatial information in OWL expressed using the Region Connection Calculus (RCC) [1] or the Cone-Shaped Directional logic formalism (CSD) [8, 9]. Choosing either representation is a design decision that depends mainly on the application. However, both RCC and CSD expressions in OWL may co-exist within the same ontology together with standard OWL semantic relations. In that respect, CHOROS 1.0 extends PelletSpatial [2] to support CSD-9 relations in addition to RCC-8 relations. A limitation of CHOROS 1.0 (and also of PelletSpatial) is that the ontology is not updated with reasoning results (i.e., the inferred triples are not added to the ontology).

Reasoning relies on the path consistency algorithm of PelletSpatial [2] extended to take into account the CSD relations in addition to the RCC ones of the original implementation. Plausible optimizations of CHOROS 1.0 are also discussed and evaluated including, a multi-threading (faster) implementation enabling the parallel execution of CSD and RCC reasoning. We also suggest reducing the 9 CSD relations to 8 by encoding the *identical to* relation using the *sameAs* OWL axiom. As shown in the experimental results of CHOROS 1.0 , this optimization speed-up the reasoner by at least 10%.

1.2 Problem Definition

Typically, when the spatial locations of all objects are given, their relations to other objects are computed by means of arithmetic operations. No further reasoning is necessary in this case (i.e., where all information is expressed quantitatively). However, there are cases where although the exact locations of objects is unknown, their spatial relations can still be expressed qualitatively by means of their relations to other objects (e.g., below, above or South, North etc). Qualitative representations are very common in natural language expressions such as in free text or speech and can be proven to be particularly useful in dealing with incomplete information in applications of the Semantic Web. This approach approximates the way humans think and reason common sense spatial knowledge.

Qualitative spatial reasoning is the process of inferring new spatial relations or checking existing spatial relations for consistency (e.g., relations "A is above B" and "B is above A" are inconsistent) in applications where spatial information is expressed qualitatively (or part of this information is expressed qualitatively and part of it is expressed quantitatively). In this work, qualitative relations between objects can be expressed by means of their topologic relations (e.g., "inside", "outside", "in touch") or means of their relative placement in the 2D space (e.g., "North", "South" etc) or combinations of the two (e.g., "A is outside B" and "A is North of B").

The most popular reasoning methods are constraint-based techniques [3, 10]. Reasoning applies on sets of qualitative spatial relations which are jointly exhaustive and pairwise disjoint. This means, that between any pairs of spatial entities exactly one of the basic relations holds. The set of all possible relations is then the set of all possible unions of the basic relations. Reasoning is realized by exploiting composition of relations, i.e. given binary relations R_1 and R_2, which hold between pairs A, B and B, C respectively, then the composition of R_1 and R_2 generates the possible relations between A and C. Compositions of relations are usually pre-computed and stored in composition tables.

1.3 Proposed Work

We present CHOROS 2.0 a reasoner engine which handles spatial data represented by topologic RCC-8 and directional CSD-9 relations. It is an improved version of CHOROS 1.0, inheriting all the basic components of its predecessor while improving its performance and functionality. It supports consistency checking over RCC and CSD relations, and computes new spatial inferences from asserted relations. Also CHOROS 2.0 supports standard RDF/OWL semantic relations, yielding a consistent ontology on OWL which can be reused or queried (e.g., using SPARQL).

We investigate on potential improvements to CHOROS 1.0 reasoning. Building upon PelletSpatial [2], CHOROS 2.0 supports consistency checking for spatial

information using Region-Connection Calculus (RCC), but also using the Cone-Shaped Directional (CSD) logic formalism as CHOROS 1.0 does. In addition CHOROS 2.0 introduces certain improvements to CHOROS 1.0 with regard to implementation of the basic path consistency mechanism and also introduces:

a) a software component that updates the ontology with the results of reasoning (i.e., the ontology is updated with the new inferred relations).

b) a software component that handles quantitative spatial information.

c) an idea for further speeding-up reasoning by introducing decompositions of directional relations into sets of simpler ones yielding smaller compositions of basic relations.

These components are missing from both PelletSpatial and CHOROS 1.0. The experimental results demonstrate that CHOROS 2.0 runs significantly faster than its respective SWRL implementation and CHOROS 1.0 in all cases.

1.4 Contributions of Present Work

The contributions of the present work are summarized below:

- CHOROS 2.0 is capable of handling both qualitative and quantitative spatial relations, by computing the relative position of spatial entities.
- For consistency checking and reasoning a new path-consistency algorithm is implemented, introduced in CHRONOS, which is based on disjunction of the compositions of basic relations and not on full composition tables used in CHOROS.
- In order to improve even more the performance of new path-consistency algorithm, we reduce the number of basic RCC and CSD relations, by replacing equal relations with owl axiom ”sameAs”.
- We introduce an approach for improving the performance of reasoning on directional relations relying on the decomposition of basic CSD-9 relations.
- A new component has been added, which allows updating the ontology with all possible logical inferences coming from spatial reasoning. This component replaces the need of providing a query engine, as updated ontology can be used as an input in any query engine, such as SOWL QL [11].

1.5 Thesis Outline

Related work in the field of knowledge representation and reasoning are discussed in Chapter 2. This includes work on representation of spatial information in ontologies, as well as reasoning using this information. Moreover earlier work regarding reasoning and querying over spatial information is presented. In Chapter

3, our proposed tool, CHOROS 2.0 is presented. In the first part of this Chapter we describe CHOROS 2.0. In the second and third part of this Chapter, the basic reasoning mechanism is discussed along with our suggested improvements over its predecessor implementation of CHOROS 1.0, respectively. Chapter 4 initially presents the theoretical evaluation of path-consistency algorithm following by the experimental evaluation of reasoning and of its extensions and improvement discussed in Chapter 3. Finally, conclusions and issues for future work are discussed in Chapter 5.

Chapter 2

Background and Related Work

2.1 Semantic Web

The ever growing information on the Web in the recent years has generated additional interest for methods and tools for automating the understanding and subsequently the processing of information by the machines. This will pave the way to technologies for automating tasks that are typically handles by humans. For example planning a trip requires selecting and purchasing tickets at specific dates at the best available price. Typically, these tasks are handled by searching the Web (e.g., using a search engine). Semantic Web[1] is intended to provide a solution to these needs by developing Web services that accomplish these tasks automatically without requiring user intervention, besides task description. These services must be capable to understand the meaning of Web pages and reason over their content in a way similar the way humans do. Semantic Web will realize this technology by introducing formal, machine readable semantics for representation of knowledge, combined with reasoning and querying support.

The Semantic Web is a vision for the future of World Wide Web in which information is given explicit meaning, making it easier for machines to automatically process and integrate available information. Currently, the World Wide Web is structured in documents written in Hypertext Markup Language (HTML), a markup convention which encodes the body text enriched with media objects. On the contrary, Semantic Web suggests languages, such as Resource Description Framework (RDF) and Web Ontology Language (OWL)[2], providing descriptions that supplement or replace the content of Web documents. Thus, Web content can be stored as descriptive data, providing machine understandable semantics, which can be used by applications to proceed and accomplish tasks, the same way humans do.

[1] http://www.w3.org/standards/semanticweb/

[2] http://www.w3.org/TR/owl-features/

2.2 Ontologies-OWL

In Semantic Web, ontologies provide the means for representing knowledge in an are of interest. Typically, ontologies comprise of a set of axioms providing definitions for the concepts (objects) of a domain and of definitions for their relations (object properties). This information is supplemented by classificatory information for such entities (i.e., making it possible to define classification hierarchies for objects and object properties) as well as, entity and relation constraints. Ontology axioms provide the semantics allowing for checking existing information for consistency or for inferring new information based on information defined explicitly in the ontology. Data instantiated to objects and object properties of an ontology are interpreted as a set of "individuals" and a set of "property assertions" that relate these individuals with each other.

Typically, ontologies comprise of the following components:

- **Individuals**: instances or objects.
- **Classes**: collections, concepts, types of objects, or kinds of things.
- **Attributes**: properties, features characteristics that objects may have.
- **Relations**: how classes and individuals can be related to each other.
- **Function terms**: complex structures formed from certain relations that can be used in place of an individual term in a statement.
- **Restrictions**: formally stated descriptions of what must be true in order for some assertion to be accepted as input.
- **Rules**: statements in the form of an if-then sentence that describe the logical inferences that can be drawn from an assertion in a particular form.
- **Axioms**: assertions (including rules) in a logical form that together comprise the overall theory that the ontology describes in its domain of application.
- **Events**: the changing of attributes or relations.

By defining shared and common domain theories, ontologies make it feasible for both, people and machines to communicate concisely, supporting the exchange of semantics and not only syntax. In recent years, ontologies have been adopted in many business and scientific communities as a way to share, reuse and process domain knowledge. Nowadays, ontologies are used in applications such as scientific knowledge portals, information management and integration systems, electronic commerce and semantic web services.

OWL

The Web Ontology Language (OWL) is a family of knowledge representation languages for authoring ontologies. OWL can be used to explicitly represent concept of terms in vocabularies and the relationships between them. It is compatible with the existing RDF, describing concepts and properties of objects, while offering increased expressiveness over the RDFS vocabulary description language. RDF and RDFS represent properties or relations between entities in the form of triplets of the form object-predicate-subject.

OWL is distinguished in three sublanguages, with different levels of expressiveness:

OWL-Full which is fully compliant with RDF. It is the most expressive variant but lacks of decidability and it is not supported by existing OWL reasoners.

OWL-DL which is based on Description Logic, and is most used version of OWL. It provides maximum possible expressiveness while preserving computational completeness, decidability and the availability of practical reasoning algorithms.

OWL-Lite which is a subset of OWL-DL, but less expressive. It allows definitions of class hierarchies and simple constraint features.

In 2007, OWL 2[3] was introduced, as an extension of OWL-DL, and became the current Semantic Web Standard. OWL 2 preserves the features of OWL-DL, while adding extra functionality and expressiveness. Some of its additional characteristics are richer data types and data ranges, qualified cardinality restrictions, asymmetric, reflexive and disjoint object properties and enhanced annotation capabilities.

2.3 SWRL

The Semantic Web Rule Language (SWRL)[4] is a proposed language for the Semantic Web that can be used to express rules as well as logic, combining OWL DL or OWL Lite with a subset of the Rule Markup Language. The proposal extends the set of OWL axioms to include Horn-like rules. It thus enables Horn-like rules to be combined with an OWL knowledge base. the proposed rules are of the form of an implication between an antecedent (body) and consequent (head). SWRL allows users to write rules that can be expressed in terms of OWL concepts to provide more powerful deductive reasoning capabilities than OWL alone.

[3] http://www.w3.org/TR/owl2-overview/

[4] http://www.w3.org/Submission/SWRL/

2.4 Qualitative Spatial Models

Space and time are fundamentals aspects of the conceptualization of the physical world. The notions of time and space as well as the evolution of concepts and individuals into space and time are important issues in almost all application domains. This is due to the fact that in many real-life situations, spatial or temporal information can not be represented easily with numerical values, or a less accurate approach is needed.

Typically, qualitative spatial information is represented by introducing a (binary) relation model for each spatial domain, containing a finite set of binary relations defined over entities of each domain. Many spatial relation models have been developed, most of them focusing on one single aspect of space, such as topology, direction, or position.

2.4.1 Topological Relations

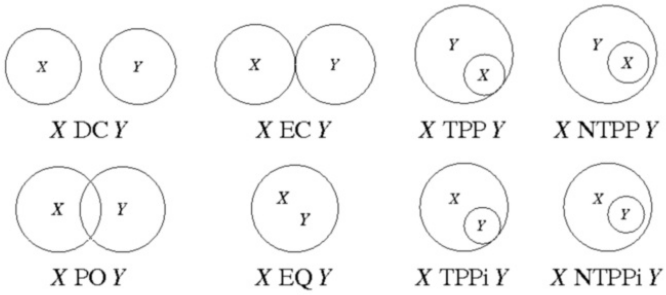


Figure 2.1: RCC8 topological relations.

Topological Relations are expressed via the region connection calculus (RCC) [1]. It abstractly describes regions (in Euclidean space, or in a topological space) by their possible relations to each other. RCC (Figure 2.1) consists of 8 basic relations that are possible between two regions:

- disconnected (DC)
- externally connected (EC)
- equal (EQ)
- partially overlapping (PO)
- tangential proper part (TPP)
- tangential proper part inverse (TPPi)
- non-tangential proper part (NTPP)
- non-tangential proper part inverse (NTPPi)

2.4.2 Directional Relations

Directional representation of points in two-dimensional space can be applied by specifying a limited number of relations, each one covering a part of the 360 degrees range. Directional relations are expressed using the cone shaped directional model, in which angular directions are assigned to the nearest cone shaped area. CSD model [8, 9] (Figure 2.2) suggests 9 basic relations between pairs of points in a 2-D space:

- north (N)
- north-east (NE)
- east (E)
- south-east (SE)
- south (S)
- south-west (SW)
- west (W)
- north-west (NW)
- identical (O)

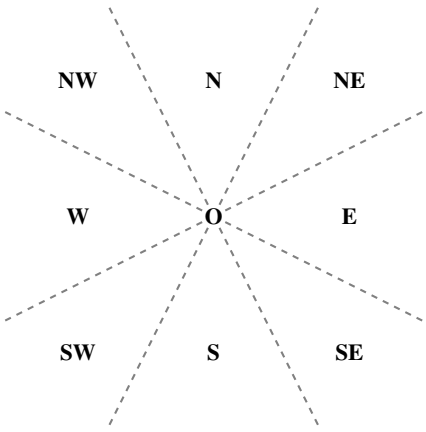


Figure 2.2: Cone Shaped Directional relations.

2.5 PelletSpatial

PelletSpatial [2] extends the Pellet OWL reasoner [15] with qualitative spatial reasoning capabilities. It supports consistency checking of spatial relations expressed using RCC-8 calculi and computes new spatial inferences from asserted relations.

Spatial relations are expressed in RDF/OWL and can be combined with arbitrary domain ontologies.

It implements two reasoners: (a) A reasoner based on the translation of RCC-8 relations to OWL-DL class axioms and, (b) A reasoner that uses the RCC-8 composition table and implements a variant of the path-consistency algorithm by Renz and Nebel. Without further optimizations, performance of the former reasoner does not scale-up well with the size of the dataset. In addition to translating RCC-8 relations to OWL class axioms, one axiom is defined for each region to satisfy the regularity condition of region (i.e., to be a non-empty concept and to contain all of the regions interior points). This significantly affects non-determinism as well as the number of qualified existential quantifiers in the ontology. Qualified existential and universal quantifiers, is one of the sources of complexity (AND-branching) in DL reasoning. The later reasoner design using path-consistency and the RCC composition table, has shown better performance.

PelletSpatial can also answer SPARQL queries that mix spatial relations with RDF/OWL relations.

2.6 CHOROS 1.0

CHOROS 1.0 [7] is a reasoner for directional and topologic informaiton in OWL. CHOROS 1.0 supports consistency checking and query answering over spatial information in OWL expressed using the Region Connection Calculus (RCC) or the Cone-Shaped Directional logic formalism (CSD). Choosing either representation is a design decision that depends mainly on the application. However, both RCC and CSD expressions in OWL may co-exist within the same ontology together with standard OWL semantic relations. In that respect, CHOROS 1.0 extends PelletSpatial to support CSD-9 relations in addition to RCC-8 relations. As such, it can answer mixed SPARQL queries over all spatial and non spatial relation types.

Architecture of CHOROS 1.0 reasoner (Figure 2.3) consists of 3 basic components:

- *Parser* implements an RDF and an ARQ parser for parsing ontologies and queries respectively. RDF parser extracts from the RDF graph the spatial triples and inserts them to the corresponding RCC and CSD constraint network for consistency checking and query answering. The rest of the non spatial graph is handled by Pellet's KB. In ARQ parser, query atoms are characterized as RCC, CSD or non spatial.

- *Reasoner* is implemented by using an exclusive reasoner for each spatial calculus. Spatial reasoning is then achieved by applying the path consistency algorithm of PelletSpatial. In order to improve its performance CHOROS 1.0 introduces a a multi-threading implementation enabling the parallel execution of CSD and RCC reasoning. Also, it suggests reducing the 9 CSD relations to 8 by replacing the "*identical to*" relation with OWL axiom "*sameAs*".

- *Query Engine* is similar to PelletSpatial's with the addition of CSD operators. It answers conjunctive queries specifying spatial and non spatial patterns. The query engine implements a dual stage query answering technique, where the set of query solutions returned by the first stage in response to a given spatial query are inserted to the second stage whose purpose is to guarantee that the non-spatial part of the query is satisfied as well.

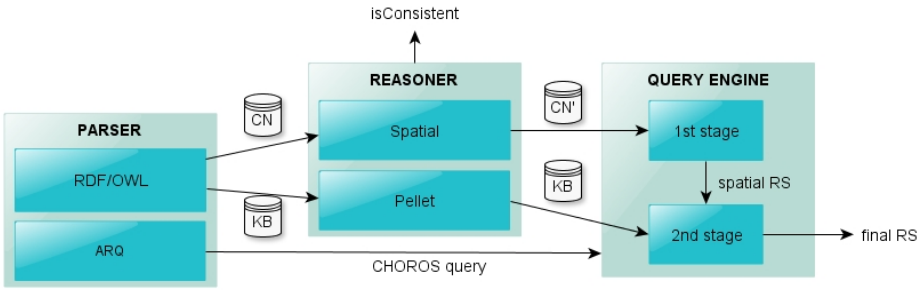


Figure 2.3: Main Components of the CHOROS 1.0 reasoner

2.7 SOWL

SOWL [12] is an ontology for representing and reasoning over spatio-temporal information in OWL. It is built-upon well established standards of the semantic web (OWL 2.0, SWRL) and enables representation of static as well as of dynamic information based on the 4D-fluents (or, equivalently, on the N-ary) approach. Representation of qualitative spatio-temporal information as well as quantitative information is a distinctive feature of SOWL. SOWL reasoner is capable of inferring new relations and checking their consistency, while retaining soundness, completeness, and tractability over the supported sets of relations.

Reasoning in SOWL, is realized by introducing a set of SWRL rules operating on spatial and temporal relations. Reasoners which support DL-safe rules, like Pellet, apart from consistency checking and inferring over OWL semantics, can be also used over spatio-temporal relations.

Regarding spatial information, both RCC topological and cone-shaped directional relations are integrated in SOWL. The SOWL spatial reasoner in particular, implements rules for RCC and CSD relations using SWRL and OWL 2.0 property axioms. All basic relations are pairwise disjoint. Their inverse relations are also defined. The "*identical to*" relation is replaced by the OWL axiom "*sameAs*". Specifically, the nine directional relations are transitive OWL relations, whilst their inverse relations are defined. Furthermore, the identity relation is symmetric. As for topological relations, relations *DC*, *EC* and *PO* are symmetric and relations *NTPPi* and *TPPi* are the inverse of *NTPP* and *TPP* respectively. Path consistency

[13] is implemented by introducing rules defining compositions and intersections of supported relations until a fixpoint is reached or until an inconsistency is detected.

2.8 Reasoning on Decomposition of Directional Relations

In [14], Batsakis proposes a new approach for reasoning over cone shaped directional relations, which is based on the decomposition of basic relations into two sets of relations, one for the East-West axis (horizontal) and one for the North-South axis (vertical). Relations on each set are jointly exclusive and pairwise disjoint but for each pair of spatial objects two relations, one for each set can hold. For example, object A can be *North* and *East* of object B corresponding to the *North-East* CSD-9 cone shaped relation.

The basic relations on North-South set, as presented in Figure 2.4, are:

- North
- South
- Equal-Horizontal
- Identical-Horizontal

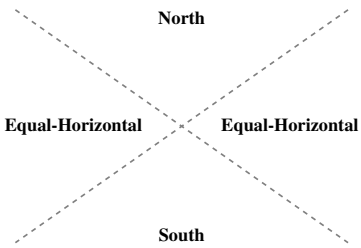


Figure 2.4: North-South Relations

The basic relations on East-West set, as presented in Figure 2.5, are:

- East
- West
- Equal-Vertical
- Identical-Vertical

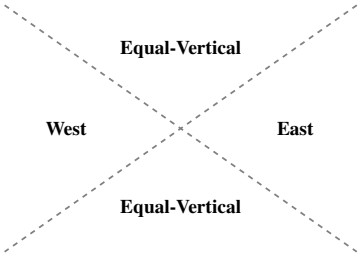


Figure 2.5: East-West Relations

The decomposition of CSD-9 relations is defined as follows:

$$
\begin{aligned}
N_{CSD9}(x,y) &\equiv North(x,y) \wedge Equal-Vertical(x,y) \\
NE_{CSD9}(x,y) &\equiv North(x,y) \wedge East(x,y) \\
E_{CSD9}(x,y) &\equiv Equal-Horizontal(x,y) \wedge East(x,y) \\
SE_{CSD9}(x,y) &\equiv South(x,y) \wedge East(x,y) \\
S_{CSD9}(x,y) &\equiv South(x,y) \wedge Equal-Vertical(x,y) \\
SW_{CSD9}(x,y) &\equiv South(x,y) \wedge West(x,y) \\
W_{CSD9}(x,y) &\equiv Equal-Horizontal(x,y) \wedge West(x,y) \\
NW_{CSD9}(x,y) &\equiv North(x,y) \wedge West(x,y) \\
Identical_{CSD9}(x,y) &\equiv Identical-Horizontal(x,y) \wedge Identical-Vertical(x,y)
\end{aligned}
$$

Reasoning in CSD relations, is realized by introducing a set of SWRL rules operating on the decomposed relations, as described above. These rules define compositions and intersections of supported relations until a fixpoint is reached or until an inconsistency is detected. In the same work, Batsakis showed that reasoning using the new formalism runs significantly faster than SOWL in both the average and worst cases. Both implementations resort to SWRL.

2.9 SPARQL

SPARQL [16] is the W3C recommendation query language for RDF. The basic evaluation mechanism of SPARQL queries is based on graph matching. The query criteria are given in the form of RDF triples possibly with variables in the place of the subject, object, or predicate of a triple, referred to as basic graph patterns. SPARQL is capable of querying required and optional graph patterns along with their conjunctions and disjunctions.

2.10 SOWL QL

SOWL Query Language (SOWL QL) [11], is a high-level query language for spatio temporal ontologies in OWL. SOWL QL is independent from the underlying

SOWL representation so that, the user need not be familiar with the peculiarities of the spatio-temporal model applied (i.e., 4D-fluents or N-ary relations). It relies on the idea of extending SPARQL with spatio-temporal operators and handles dynamic (spatio-temporal) ontologies almost like static ones.

Main advantages of SOWL QL are summarized below:

- it is independent over the underlying model of spatio temporal representation.
- user need not be familiar with the peculiarities of the underlying model.
- it supports an exhaustive set of spatial and temporal operators including all temporal Allen, as well as all topological and directional operators
- it supports querying of qualitative expressions (defined using natural language terms such as before, after, East, West) in addition to quantitative spatio-temporal expressions
- it supports reasoning during the querying process (i.e., queries specifying exact temporal or spatial values call for reasoning support)

SOWL QL queries are translated into equivalent SPARQL queries. In fact, SOWL QL is an extension of SPARQL, using the same clauses as SPARQL does and fully supports all SPARQL features. The structure of a SOWL QL query is the same with SPARQL, with the additions of spatio temporal operators (all temporal Allen relations and all spatial topological and directional relations operators are implemented as operators), which are not supported by other query languages, meaning that any SOWL QL query can also be expressed in SPARQL and vice versa. Query selection criteria are expressed by RDF triples of the form *subject - predicate - object* referred to as basic graph patterns.

SOWL QL is fully implemented and supported by a Graphical User Interface (GUI).

2.11 Jena

Jena[5] is an open source Semantic Web framework for Java. It provides an API for reading, processing and writing data in RDF graphs, while offering efficient storage techniques of large numbers of RDF triples. Jena is compliant with W3C recommendations providing an ontology API for handling OWL and RDFS ontologies, and also a query engine according to SPARQL specifications. The framework has various internal reasoners and the Pellet reasoner can be set up to work in Jena.

[5] http://jena.apache.org/

Chapter 3

CHOROS 2.0

CHOROS 2.0 is a reasoning engine for spatial information in OWL expressed using RCC and CSD models. It is the result of an investigation on suggesting potential improvements over CHOROS 1.0 [7]. CHOROS 1.0, is an early version of a qualitative spatial reasoning engine supporting consistency checking, inference of new spatial relations as well as query answering on qualitative spatial information expressed using Region-Connection-Calculus (RCC) and Cone-Shaped Directional logic Formalism (CSD). It works with all RCC and CSD relations in combination with standard RDF/OWL semantic relations in an OWL ontology which are handled by Pellet. It is practically an extension over PelletSpatial [2] for handling directional in addition to topologic information. CHOROS 2.0 inherits all features of CHOROS 1.0 and in addition suggests certain improvements to the basic path consistency mechanism for enhancing its run-time performance. In addition, CHOROS 2.0 supports updating of the ontology with inferred facts, a feature which is not supported by PelletSpatial or by its predecessor implementation. These new spatial inferences can be stored in an updated ontology which can be re-used or queried using SPARQL.

3.1 Spatial Relations in CHOROS 2.0

CHOROS 2.0 defines an RDF/OWL vocabulary of terms denoting RCC topological (Figure 2.1) and Cone shaped directional (Figure 2.2) models. More specifically, using this formalism, a region (a spatial entity in general) is defined as an OWL individual (e.g., Individual: Town1, Individual: Town 2 oftype class: Town) while, a spatial relation between two entities is defined as an OWL object property assertion (e.g., Individual: Town1 ObjectProperty: WestOf Individual: Town2).

As shown in Figure 3.1, CSD and RCC relations are defined as simple object properties with no extra characteristics (e.g., inverse, transitive). One can either use the vocabulary provided, or use his own by defining sub-property axioms (e.g., *ObjectProperty: N subPropertyOf: NorthOf*).

Non-spatial relations are represented as ordinary OWL assertions (e.g., Individ-

ual: Town1 DatatypeProperty: hasName: Athens). In order to perform efficiently CHOROS 2.0 strictly separates spatial reasoning from semantic DL reasoning by using an exclusive spatial reasoner component.

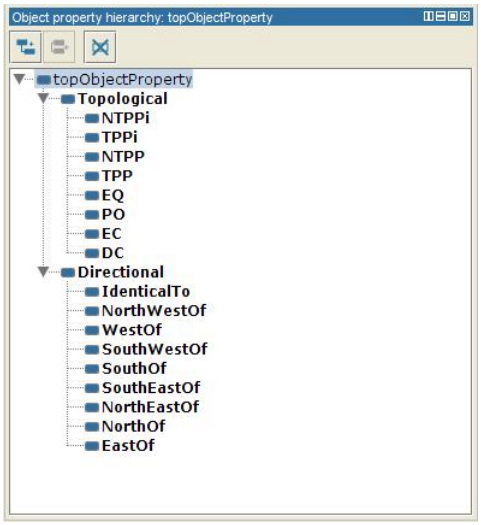


Figure 3.1: RCC and CSD Relations as object properties, in Protégé

3.2 CHOROS 2.0 Architecture

Figure 3.2 illustrates CHOROS 2.0 architecture. It consists of several modules which are described in the following focusing on differencies and unique characteristics over CHOROS 1.0 (Figure 2.3).

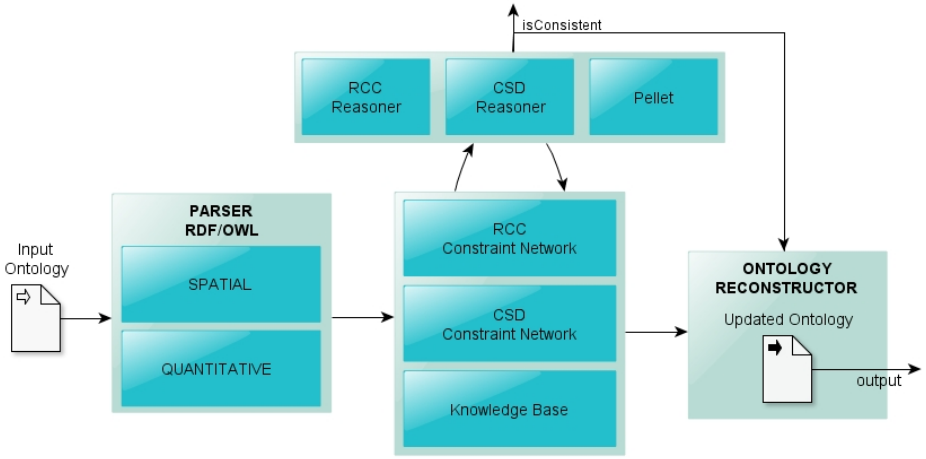


Figure 3.2: Main Components of CHOROS 2.0 reasoner

The Parser component of CHOROS 1.0 is responsible for loading and handling of ontologies and SPARQL queries. It implements an Ontology Parser which loads ontologies and separates spatial relations from the Ontology model. Also an ARQ Parser is implemented in order to handle SPARQL queries by separating spatial from non-spatial OWL triples. The Parser component of CHOROS 2.0 implements only an Ontology Parser and introduces an additional Parser for handling quantitative spatial relations.

Furthermore, CHOROS 1.0 implements a Query Engine for answering spatial queries, derived by the ARQ Parser. It implements a dual stage query answering technique, where the set of query solutions returned by the first stage in response to a given spatial query are inserted to the second stage whose purpose is to guarantee that the non-spatial part of the query is satisfied as well. This implementation is restrictive, as queries must contain at least one spatial triple, otherwise an empty result set is returned. Notice also that inferred relations are inserted to the respective (CSD or RCC) constraint network rather that the ontology itself. CHOROS 2.0 deals this problem by replacing Query Engine with an Ontology Reconstruction component, which updates the ontology with inferred facts. Thus, the updated ontology can be used by any Query Engine such as SPARQL or SOWL-QL [11].

CHOROS 2.0 consists of 4 main components:

- *Parser* which loads the ontologies and extracts their spatial relations.
- *Constraint Network* which stores spatial property assertions as long as non-spatial standard OWL assertions.
- *Reasoner* which is responsible for consistency checking and logical inference.
- *Reconstructor* which updates ontology with new spatial inferences.

3.2.1 Parser

The Parser component is responsible for loading ontologies into the memory. Its main task is to handle RDF/OWL information. It implements, the *Spatial Parser* which separates spatial and non-spatial relations from ontologies, and the *Quantitative Parser* which handles specific quantitative spatial information. Both tools are implemented with the use of Jena[1] framework, which provides an API for reading, processing and writing data in RDF graphs.

Spatial Parser

The spatial parser extracts spatial information from RDF graphs. In order to recognize spatial information, spatial vocabularies are defined consisting of name definitions of the basic CSD and RCC relations. All spatial triples are identified and

[1] http://jena.apache.org/

removed from the graph, and then stored in the appropriate RCC or CSD constraint network. The rest of the graph, containing non-spatial standard OWL assertions, is stored as an ontology in the knowledge base and is handled by an ordinary Pellet[2] reasoner.

Quantitative Parser

Handling quantitative spatial information is a new feature introduced by CHOROS 2.0. Quantitative spatial information refers to the representation of certain aspects of spatial information, such as position, expressed in numerical values. Similarly to SOWL model [12], spatial objects containing information about their relative position, must be defined as individuals of *Point* Class. *Point* Class has also 2 numerical attributes, namely X and Y, to represent footprints of objects. Spatial objects described by their relative position to other objects, can be used to infer spatial relations between other objects. The task of making this information accessible by the reasoner is undertaken by the Quantitative Parser.

The Quantitative Parser retrieves all *Point* individuals from the ontology, computes the angle between every pair of *Point* individuals, and is mapped to the corresponding CSD relation between the two points (i.e., object locations). Finally, this relation is inserted into the CSD Constraint Network. Figure 3.3 shows how the angle between two object locations (x1,y1) and (x2,y2) is computed:

$$angle = \arctan\left(\frac{Dy}{Dx}\right) * \frac{180}{\pi}$$

$$\text{where,} \quad Dy = y_2 - y_1 \quad \text{and} \quad Dx = x_2 - x_1$$

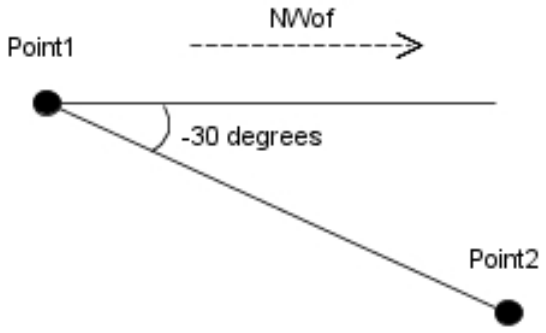


Figure 3.3: Computing a CSD relation from point locations

The mapping of angles to CSD relations is realized according to the following rules:

[2]http://clarkparsia.com/pellet/

- If $-22.5^\circ \leq angle \leq 22.5^\circ$ then *WestOf* relation holds.
- If $22.5^\circ < angle \leq 67.5^\circ$ then *SouthWestOf* relation holds.
- If $67.5^\circ < angle \leq 112.5^\circ$ then *SouthOf* relation holds.
- If $112.5^\circ < angle \leq 157.5^\circ$ then *SouthEastOf* relation holds.
- If $157.5^\circ < angle \leq 180^\circ$ or $-157.5^\circ < angle \leq -180^\circ$ then *EastOf* relation holds.
- If $-112.5^\circ < angle \leq -157.5^\circ$ then *NorthEastOf* relation holds.
- If $-67.5^\circ < angle \leq -112.5^\circ$ then *NorthOf* relation holds.
- If $-22.5^\circ < angle \leq -67.5^\circ$ then *NortWestOf* relation holds.
- If no angle is calculated then *Identical* relation holds.

3.2.2 Constraint Network

A Constraint Network (CN) is a set of variables together with a set of constraints. CHOROS 2.0 implements two constraint networks, one for RCC relations and the other for CSD relations. They store spatial relations, extracted from the ontology during parsing. They also provide mechanisms for retrieving these relations.

Apart from spatial relations, a spatial ontology consists also of non-spatial standard OWL assertions. These non-spatial relations are stored in Pellet's Knowledge Base, a structure similar to a constraint network. A Knowledge Base (KB) is a special kind of database for knowledge management, providing the means for the organization, and retrieval of knowledge. The Knowledge Base of Pellet is a combination of an assertional box (that contains assertions about individuals) and a terminological box (that contains axioms about classes). This KB is used for consistency checking and inference on non-spatial information.

3.2.3 Reasoner

CHOROS 2.0 handles separately spatial from semantic DL reasoning. Non-spatial relations are stored as a Knowledge Base and are managed by Pellet. On the other hand, spatial relations are stored in the corresponding RCC and CSD Constraint Networks. CHOROS 2.0 provides a dedicated reasoner engine for each spatial model. These two reasoner engines are implemented in Java as separate threads, in order to enable "parallel" and concurrent execution of their tasks.

Both CSD and RCC spatial models, are expressed by a set of jointly exclusive and pairwise disjoint basic relations, which is closed under several operations (composition, intersection, inverse operation).

Spatial reasoning (i.e., inferring implied relations or detecting inconsistencies) can be viewed as a constraint satisfaction problem which is NP in the general case.

It is implemented by means of a path consistency algorithm [2], which computes all inferred relations using compositions of existing relations defined, until a fixed point is reached (i.e., the algorithm does not yield new inferences) or until an inconsistency is detected (i.e., yield the empty relations as a result). Path consistency when applied on a set of assertions containing only basic relations retains tractability and guarantees soundness and completeness of reasoning [17]. The possible compositions of basic relations are stored in separate composition tables, one for each type of spatial model. Because the compositions of basic relations for the two models are mutually exclusive, CHOROS 2.0 clearly separates reasoning for each type of spatial model (i.e., reasoning is implemented and run separately).

Composition Tables of Spatial Relations

A composition table represents the result of the composition of pairs of basic relations. For example, if relation R_1 holds between $object_1$ and $object_2$, and relation R_2 holds between $object_2$ and $object_3$, then the corresponding entry of composition table denotes the possible relation(s) holding between $object_1$ and $object_3$

$$R_1(object_1, object_2) \wedge R_2(object_2, object_3) \rightarrow R_3(object_1, object_3)$$

The composition tables for spatial relations are represented below. Specifically, Table 3.1 refers to RCC basic relations, and Table 3.2 refers to CSD basic relations.

Path Consistency Algorithm

Path consistency algorithm (Algorithm 1) [2] computes all inferred relations using compositions of existing relations defined, until a fixed point is reached (i.e., new inferences don't arise) or until an inconsistency is detected (i.e., $\emptyset$ is produced as a result).

Given a constraint network N, N is consistent if it is either empty, or if every relation in the network is consistent. Notice that, the requirement for the relations in N to be defined, i.e. of the set of eight basic RCC relations, is relevant to the tractability of a sound and complete path-consistency procedure. According to [12, 18], a sound and complete path-consistency is tractable for the set of CSD and RCC relations.

A queue Q is used as a structure to store relations that have to be processed. The algorithm is executed until $Q = \emptyset$, or an inconsistency is detected. Q is initialized with all defined relations $R_{ij} \in N$ (Line 6).

	DC	**EC**	**PO**	**TPP**	**NTPP**	**TPPi**	**NTPPi**	**EQ**
DC	DC, EC, PO, TPP, NTPP, TPPi, NTPPi, EQ	DC, EC, PO, TPP, NTPP	DC, EC, PO, TPP, NTPP	DC, EC, PO, TPP, NTPP	DC, EC, PO, TPP, NTPP	DC	DC	DC
EC	DC, EC, PO, TPPi, NTPPi	DC, EC, PO, TPP, TPPi, EQ	DC, EC, PO, TPP, NTPP	EC, PO, TPP, NTPP	PO, TPP, NTPP	DC, EC	DC	EC
PO	DC, EC, PO, TPPi, NTPPi	DC, EC, PO, TPPi, NTPPi	DC, EC, PO, TPP, NTPP, TPPi, NTPPi, EQ	PO, TPP, NTPP	PO, TPP, NTPP	DC, EC, PO, TPPi, NTPPi	DC, EC, PO, TPPi, NTPPi	PO
TPP	DC	DC, EC	DC, EC, PO, TPP, NTPP	TPP, NTPP	NTPP	DC, EC, PO, EQ, TPP, NTPP	DC, EC, PO, TPPi, NTPPi	TPP
NTPP	DC	DC	DC, EC, PO, TPP, NTPP	NTPP	NTPP	DC, EC, PO, TPP, NTPP	DC, EC, PO, TPP, NTPP, TPPi, NTPPi, EQ	NTPP
TPPi	DC, EC, PO, TPPi, NTPPi	EC, PO, TPPi, NTPPi	PO, TPPi, NTPPi	EQ, PO, TPPi, TPP	PO, TPP, NTPP	TPPi, NTPPi	NTPPi	TPPi
NTPPi	DC, EC, PO, TPPi, NTPPi	PO, TPPi, NTPPi	PO, TPPi, NTPPi	PO, TPPi, NTPPi	PO, TPP, NTPP, EQ, TPPi, NTPPi	NTPPi	NTPPi	NTPPi
EQ	DC	EC	PO	TPP	NTPP	TPPi	NTPPi	EQ

Table 3.1: RCC Composition Table

	N	**NE**	**E**	**SE**	**S**	**SW**	**W**	**NW**	**O**
N	N	N,NE	N,NE,E	N, NE, E, SE	N, NE, E, SE, S, SW, W, NW, O	W, NW, SW, N	NW, W, N	NW,W	N
NE	NE, E	NE	NE, E	E, NE, SE	E, NE, SE, S	N, NE, E, SE, S, SW, W, NW, O	N, NE, NW, W	N, NE, NW	NE
E	NE, E, N	NE, E	E	SE, E	SE, E, S	S, SW, SE, E	N, NE, E, SE, S, SW, W, NW, O	N, NW, NE, E	E
SE	E, SE, NE, N	E, SE, NE	SE, E	SE	SE, S	S, SE, SW	S, W, SE, SW	N, NE, E, SE, S, SW, W, NW, O	SE
S	N, NE, E, SE, S, SW, W, NW, O	E, S, NE, SE	SE, E, S	SE, S	S	S, SW	S, W, SW	W, S, NW, SW	S
SW	W, SW, N, NW	N, NE, E, SE, S, SW, W, NW, O	S, SW, SE, E	S, SW, SE	SW,S	SW	SW, W	W, NW, SW	SW
W	N, W, NW	N, NW, NE, W	N, NE, E, SE, S, SW, W, NW, O	S, SE, SW, W	W, S, SW	W, SW	W	W, NW	W
NW	N, NW	N, NW, NE	N, NW, NE, E	N, NE, E, SE, S, SW, W	W, NW, SW, S	W, NW, SW	NW, W	NW	NW
O	N	NE	E	SE	S	SW	W	NW	O

Table 3.2: CSD Composition Table

Algorithm 1 Path-Consistency Algorithm

```
1:  procedure PATHCONSISTENCY(N)
2:      if N = ∅ then
3:          return true
4:      end if
5:      complete(N)
6:      Q ← {R_ij \ R_ij ∈ N}
7:      while Q ≠ ∅ do
8:          R_ab ← remove(Q)
9:          if !isConsistent(N, Q, R_ab) then
10:             return false
11:         end if
12:     end while
13:     return true
14: end procedure

15: procedure ISCONSISTENT(N, Q, R_ab)
16:     for S_bc ∈ N do
17:         T_bc ← R_ab ∘ S_bc
18:         add(N, Q, T_ac)
19:         if !isConsistent then
20:             return false
21:         end if
22:     end for
23:     return true
24: end procedure

25: procedure ADD(N, Q, T_ac)
26:     if T = ⊤ then
27:         return
28:     end if
29:     U_ac ← {R_ij \ i = a, j = c, R_ij ∈ N}
30:     if ∄U_ac then
31:         V_ac ← T_ac
32:     else
33:         V_ac ← T_ac ∩ U_ac
34:         if V = ∅ then
35:             isConsistent = false
36:             return
37:         end if
38:         if U = V then
39:             return
40:         end if
41:         N ← N \ {U_ac}
42:     end if
43:     N ← N ∪ {V_ac}
44:     Q ← Q ∪ {V_ac}
45:     add(N, Q, {V_ĉa})
46: end procedure
```

The $complete$ step (line 5), includes the $inverse$ step and the $equals$ step. In $inverse$ step for every defined relation $R_{ij} \in N$, we ensure that, $R_{\widehat{ji}} \in N$ (its inverse relation is also in N). In $equal$ step for every region $x \in N$, we ensure that relation $Equal(x, x) \in N$.

A relation R_{ab} (Line 15) is path-consistent if the rule for combining a compositional inference with existing information [19],

$$V_{ac} \leftarrow U_{ac} \cap R_{ab} \circ S_{bc}$$

results in a non-empty set ($V \neq \emptyset$) for regions a and c, where $S_{bc} \in N$ relations with a transitive path with R_{ab} from a, through b, to c and U_{ac} a relation possibly $\in$ N as existing information.

The compositional inference $T_{ac} \leftarrow R_{ab} \circ S_{bc}$ (Line 17), is computed for regions a and c as the union set T for the composition of each pair (r,s) in the set $R \times S$, $r \in R$, $s \in S$. The composition of a pair (r,s) consists of a lookup for the CSD or RCC composition table given that r and s are elements of the set of defined relations.

If $U_{ac} \in N$ (i.e., there is existing information for the pair (a,c)), we complete the rule by computing the intersection $V_{ac} \leftarrow T_{ac} \cap U_{ac}$ (Line 33), where V is

the intersection set of relations $v \in T \cap U$. This step refines the already existing relation $U_{ac} \in N$ and is essential for the path-consistency algorithm as it defines the inconsistent state: if $V = \emptyset$ an inconsistency is found.

Line 38 states that if $U = V$, the step $V_{ac} \leftarrow T_{ac} \cap U_{ac}$ (line 33), could not refine relation U_{ac}. Hence, combining compositional inference T_{ac} with existing information U_{ac} does not add new information. In this case, we can return. Otherwise, U_{ac} is removed from N, the refined V_{ac} is added to N and Q, and the inverse $V_{\widehat{ca}}$ is processed.

To conclude, path consistency is implemented by consecutive application of the formula:

$$R_s(x, y) \leftarrow R_i(x, y) \cap (R_j(x, k) \circ R_l(k, y)), \forall x, y, k$$

representing intersection of compositions of relations with existing relations (symbol $\cap$ denotes intersection, symbol $\circ$ denotes composition and R_i, R_j, R_l, R_s denote temporal relations) [12]. This formula is applied until there are no new inferences, or until until the empty set is reached, implying that the network is inconsistent.

In order to comprehend the functionality of path consistency algorithm consider the following examples:

CONSISTENT EXAMPLE

Given the following CSD constraint network which represents the directional relations between 4 houses:

- house1 N house2
- house2 NW house3
- house1 NE house4
- house4 N house3

Using the CSD composition table 3.2 and the path-consistency algorithm described above, we can refine the network in the following way:

$$(house1\ \mathbf{N}\ house2) \circ (house2\ \mathbf{NW}\ house3)$$
$$\rightarrow house1\ \{\mathbf{N} \cup \mathbf{NW}\}\ house3$$

$$(house1\ \mathbf{NE}\ house4) \circ (house4\ \mathbf{N}\ house3)$$
$$\rightarrow house1\{\mathbf{N} \cup \mathbf{NE}\}\ house3$$

$$(house1\ \{\mathbf{N} \cup \mathbf{NW}\}\ house3) \cap (house1\ \{\mathbf{N} \cup \mathbf{NE}\}\ house3)$$

$$\rightarrow house1 \ \mathbf{N} \ house3$$

That is, the first house is north of the third which is the intersection of the above two relations.

INCONSISTENT EXAMPLE

Given the following CSD constraint network which represents the directional relations between 4 houses:

- house1 N house2
- house2 NW house3
- house1 SE house4
- house4 E house3

Using the CSD composition table and the path-consistency algorithm, we can refine the network in the following way:

$$(house1 \ \mathbf{N} \ house2) \circ (house2 \ \mathbf{NW} \ i_3)$$
$$\rightarrow house1 \ \{\mathbf{N} \cup \mathbf{NW}\} \ house3$$

$$(house1 \ \mathbf{SE} \ house4) \circ (house4 \ \mathbf{E} \ house3)$$
$$\rightarrow house1\{\mathbf{E} \cup \mathbf{SE}\} \ house3$$

$$(house1 \ \{\mathbf{N} \cup \mathbf{NW}\} \ house3) \cap (house1 \ \{\mathbf{E} \cup \mathbf{SE}\} \ house3)$$
$$\rightarrow \emptyset, \quad \textbf{inconsistency}$$

That is, network is inconsistent because the intersection of the above two relations is the empty relation.

3.2.4 Re-constructor

Re-constructor component updates the ontology with information inferred by the reasoner and by this, make this information accessible to users. At the end of the reasoning process, if no inconsistency is detected in any Constraint Network or in the Knowledge Base, Re-constructor creates an ontology as an output of CHOROS 2.0. The new ontology contains all original relations of input ontology, enriched with new spatial inferences computed during reasoning.

Spatial relations from both CSD and RCC constraint networks are asserted into the new ontology, as OWL object property assertions. Finally, Pellet reasoner

is applied in order to ensure consistency over the total set of OWL-DL axioms. The output ontology can be used by any query engine, such as SOWL Ql [11] or SPARQL, since spatial information are handled as typical OWL object properties assertions.

3.3 Optimizations

The reasoning engine is the core of CHOROS 2.0 system. In order to improve its performance, possible optimizations are discussed in the following:

Handling Disjunctions of Basic Relations

CHOROS 1.0 implements the same reasoning engine introduced by PelletSpatial and is based on the composition table of basic relations. However, as Table 3.1 and Table 3.2 illustrates, not all compositions yield a unique relation as a result. For example, the composition of relations $NTPP$ and EC returns as a result relation DC, while compositions of relations EC and PO returns as a result a set of five possible relations (DC, EC, PO, TPP, $NTPP$). Disjunctions of relations are represented using new relations, whose compositions must also be defined and asserted into the knowledge base.

CHOROS 1.0 pre-computes the set of possible compositions over all disjunctions of basic CSD and RCC relations and stores them into a static full composition table. Therefore, full composition table of RCC relations contains $2^8 \times 2^8 = 65536$ relations, and the corresponding full composition table of CSD relation contains $2^9 \times 2^9 = 262144$ relations.

On the contrary, CHOROS 2.0 adopts the approach introduced in CHRONOS [20]. In CHRONOS compositions of disjunctions of basic relations are defined as the disjunction of the compositions of these basic relations. For example, composition between the disjunction of relations (DC, EC, $TPPi$) and relation $TPPi$ is computed as follows:

$$(DC \cup EC \cup TPPi) \circ TPPi$$
$$\rightarrow (DC \circ TPPi) \cup (EC \circ TPPi) \cup (TPPi \circ TPPi)$$
$$\rightarrow DC \cup (DC \cup EC) \cup (TPPi \cup NTPPi)$$
$$\rightarrow DC \cup EC \cup TPPi \cup NTPPi$$

Following the later (referred to as "dynamic approach") a series of compositions of relations is computed "on the fly" (rather that storing all intermediate composition results in the memory) which involves only a single look-up operation in the CSD or RCC composition table respectively. This not only saves memory but also (as will be shown in the experiments) computes faster, since it stores only the compositions over the basic relations. Specifically, composition table of RCC relations contains $2^8 = 256$ relations, and the corresponding composition table of CSD relations contains $2^9 = 512$ relations.

Reduction of Basic Relations

In CHOROS 1.0, the set of possible compositions over all disjunctions of basic CSD relations is an order of magnitude greater than the corresponding set of basic RCC relations. Therefore, the composition table of basic CSD relations greatly affects the performance of CHOROS 1.0 reasoner engine, even for small datasets. To deal this problem CHOROS 1.0 proposes reduction of basic CSD relations from 9 to 8, by replacing the directional relation "*identicalTo*" with owl axiom "*sameAs*".

CHOROS 2.0 also proposes reduction of CSD basic relations, extending this approach to RCC basic relations. Specifically, topological relation "EQ" is also replaced by owl axiom "sameAs". With this representation the spatial reasoning engines handle less relations, improving reasoning performance, while identical relations are asserted into Knowledge Base and treated as standard OWL axioms.

Decomposition of CSD Relations

CHOROS 2.0 implements the approach described in section 2.8 [14] by modifying the basic components of its architecture (Figure 3.2). Since, CSD relations are expressed by the new sets, CHOROS 2.0 replaces CSD Constraint Network with the North-South Constraint Network and the East-West Constraint Network in order to store the relations created from the decomposition of basic CSD relations. Decomposition of CSD relations is applied in Parser component, while spatial relations from input ontology are extracted.

Reasoning over directional relations, is realized by implementing the North-South reasoner and the East-West reasoner, for the corresponding Constraint Networks. Both reasoners are based on the path-consistency algorithm (Algorithm 1), as described above, and define compositions of basic relations.

Table 3.3 represents the compositions of relations of North-South set. Relations *North*, *South*, *Equal-Horizontal* and *Identical-Horizontal*, are denoted by *N*, *S*, *EqH*, *IdH* respectively.

	N	**S**	**EqH**	**IdH**
N	N	N, S, EqH, IdH	N, EqH	N
S	N, S, EqH, IdH	S	S, EqH	S
EqH	N, EqH	S, EqH	N, S, EqH, IdH	EqH
IdH	N	S	EqH	IdH

Table 3.3: Composition Table for North-South Directional Relations

Table 3.4 represents the compositions of relations of East-West set. Relations *East*, *West*, *Equal-Vertical* and *Identical-Vertical*, are denoted by *E*, *W*, *EqV*, *IdV* respectively.

Finally, in Re-constructor component directional relations from both Constraint Networks are combined to compose the CSD-9 relations, which are inserted into the output ontology. Composition of CSD-9 relations is achieved by applying an

	E	**W**	**EqV**	**IdV**
E	E	E, W, EqV, IdV	E, EqV	E
W	E, W, EqV, IdV	W	W, EqV	W
EqV	E, EqV	W, EqV	E, W, EqV, IdV	EqV
IdV	E	W	EqV	IdV

Table 3.4: Composition Table for East-West Directional Relations

addition set of rules (which are proposed in this work).

Notice though, that the experimental results demonstrate that reasoning on directional relations decomposed to relations of North-South and East-West set, infers less relations than the CSD-9 model. These results are due to the semantics of the set of relations replacing the ordinary CSD-9 relations, from which the reasoning process depends on.

For example, given 4 objects and their relations as shown below:

- Object1 SE Object2
- Object2 SW Object3
- Object3 E Object4

Applying reasoning process (Algorithm 1) according to the CSD-9 model, infers the following relations:

- Object1 (SE, SW, S) Object3
- Object2 (S, SE, SW, E) Object4
- Object1 (S, SE, SW, E) Object4

On the other hand, to apply the reasoning process according to the decomposition of directional relations, it is necessary to express initial relations on the North-South set and the East-West set firstly.

North-South axis relations	**East-West axis relations**
• Object1 S Object2 • Object2 S Object3 • Object3 EqH Object4	• Object1 E Object2 • Object2 W Object3 • Object3 E Object4

Then the reasoning process is applied on each set separately, and inferred relations for each set are shown below.

North-South axis relations	East-West axis relations
• Object1 S Object3 • Object2 (S, EqH) Object4 • Object1 (S, EqH) Object4	Additional relations cannot be inferred. Composition of relations *E* and *W* yields all possible relations of the East-West set, which don't add any information to the network, causing the algorithm to skip them.

It is impossible, by combining existing information in both networks, to generate the same relations as the CSD-9 model. Using the Re-constructor component all possible relations that can be generated from both networks are:

- Object1 (SE, SW, S) Object3
- Object2 (S, SE, SW) Object4
- Object1 (S, SE, SW) Object4

We observe that relation *E* is not generated between pairs Object2, Object4 and Object1 and Object4.

To deal with this problem, we propose replacement of relations *Equal-Horizontal* and *Equal-Vertical*. Regarding the North-South set, relation *Equal-Horizontal* is replaced by relations *Horizontal-East* and *Horizontal-West* (Figure 3.5) in order to be more accurate over the horizontal axis. Similarly for the East-West set, relation *Equal-Vertical* is replaced by relations *Vertical-North* and Vertical-South (Figure 3.4).

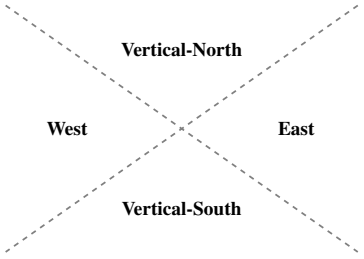


Figure 3.4: New East-West Relations

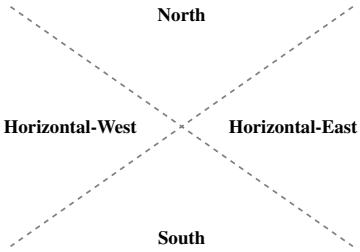


Figure 3.5: New North-South Relations

Existence of new relations, slightly changes decomposition of CSD relations, which are defined as follows:

$$
\begin{aligned}
N_{CSD9}(x,y) &\equiv North(x,y) \wedge Vertical - North(x,y) \\
NE_{CSD9}(x,y) &\equiv North(x,y) \wedge East(x,y) \\
E_{CSD9}(x,y) &\equiv Horizontal - East(x,y) \wedge East(x,y) \\
SE_{CSD9}(x,y) &\equiv South(x,y) \wedge East(x,y) \\
S_{CSD9}(x,y) &\equiv South(x,y) \wedge Vertical - South(x,y) \\
SW_{CSD9}(x,y) &\equiv South(x,y) \wedge West(x,y) \\
W_{CSD9}(x,y) &\equiv Horizontal - West(x,y) \wedge West(x,y) \\
NW_{CSD9}(x,y) &\equiv North(x,y) \wedge West(x,y) \\
Identical_{CSD9}(x,y) &\equiv Identical - Horizontal(x,y) \wedge Identical - Vertical(x,y)
\end{aligned}
$$

Table 3.5 represents compositions of relations of East-West set. Relations *Vertical-North*, *Vertical-South* are denoted by *VerN*, *VerS* respectively.

	E	**W**	**VerN**	**VerS**	**IdV**
E	E	E, W, VerN, VerS, IdV	E, VerN	E, VerS	E
W	E, W, VerN, VerS, IdV	W	W, VerN	W, VerS	W
VerN	E, VerN	W, VerN	VerN	E, W, VerN, VerS, IdV	VerN
VerS	E, VerS	W, VerS	E, W, VerN, VerS, IdV	VerS	EqV
IdV	E	W	VerN	VerS	IdV

Table 3.5: Composition Table for new East-West Directional Relations

Table 3.6 represents compositions of relations of new North-South set. Relations *Horizontal-East*, *Horizontal-West* are denoted by *HorE*, *HorW* respectively.

	N	**S**	**HorE**	**HorW**	**IdH**
N	N	N, S, HorE, HorW, IdH	N, HorE	N, HorW	N
S	N, S, HorE, HorW, IdH	S	S, HorE	S, HorW	S
HorE	N, HorE	S, HorE	HorE	N, S, HorE, HorW, IdH	HorE
HorW	N, HorW	S, HorW	N, S, HorE, HorW, IdH	HorW	HorW
IdH	N	S	HorE	HorW	IdH

Table 3.6: Composition Table for new North-South Directional Relations

The benefits from the definition of relations can be seen by applying the reasoner process on the example above. Firstly, original relations are expressed on the North-South set and the East-West set.

North-South axis relations	**East-West axis relations**
• Object1 S Object2 • Object2 S Object3 • Object3 HorE Object4	• Object1 E Object2 • Object2 W Object3 • Object3 E Object4

Then, reasoning is applied on each set separately, and inferred relations are shown below.

North-South axis relations	**East-West axis relations**
• Object1 S Object3 • Object2 (S, HorE) Object4 • Object1 (S, HorE) Object4	Additional relations cannot be inferred. Composition of relations *E* and *W* yields all possible relations of the East-West set, which don't add any information to the network, causing the algorithm to skip them.

Regarding the East-West axis, no additional relations can be inferred. However, from the North-South network there is enough information to generate all possible relations as CSD-9 model does. Below we present the CSD re-composition rules which are the inverse of the de-composition rules described above, enriched with rules handling relations between spatial entities that exist in either the East-West or the North-South Network.

For each relation $relNS(x, y)$ in North-South Network:

- If a relation $relEW(x, y)$ exists in East-West Network then:
  - if *North* $\in relNS(x, y)$ AND *Vertical-North* $\in relEW(x, y)$ add in ontology $North_{CSD9}(x, y)$.

  - if *South* $\in relNS(x, y)$ AND *Vertical-South* $\in relEW(x, y)$ add in ontology $South_{CSD9}(x, y)$.
  - if *Horizontal-East* $\in relNS(x, y)$ AND *East* $\in relEW(x, y)$ add in ontology $East_{CSD9}(x, y)$.
  - if *Horizontal-West* $\in relNS(x, y)$ AND *West* $\in relEW(x, y)$ add in ontology $West_{CSD9}(x, y)$.
  - if *North* $\in relNS(x, y)$ AND *East* $\in relEW(x, y)$ add in ontology $NorthEast_{CSD9}(x, y)$.
  - if *North* $\in relNS(x, y)$ AND *West* $\in relEW(x, y)$ add in ontology $NorthWest_{CSD9}(x, y)$.
  - if *South* $\in relNS(x, y)$ AND *East* $\in relEW(x, y)$ add in ontology $SouthEast_{CSD9}(x, y)$.
  - if *South* $\in relNS(x, y)$ AND *West* $\in relEW(x, y)$ add in ontology $SouthWest_{CSD9}(x, y)$.
  - if *Identical-Horizontal* $\in relNS(x, y)$ AND *Identical-Vertical* $\in relEW(x, y)$ add in ontology $Identical_{CSD9}(x, y)$.

- else:
  - if *North* $\in relNS(x, y)$ add in ontology:
    * $North_{CSD9}(x, y)$
    * $NorthEast_{CSD9}(x, y)$
    * $NorthWest_{CSD9}(x, y)$
  - if *South* $\in relNS(x, y)$ add in ontology:
    * $South_{CSD9}(x, y)$
    * $SouthEast_{CSD9}(x, y)$
    * $SouthWest_{CSD9}(x, y)$
  - if *Horizontal-East* $\in relNS(x, y)$ add in ontology:
    * $East_{CSD9}(x, y)$
  - if *Horizontal-West* $\in relNS(x, y)$ add in ontology:
    * $West_{CSD9}(x, y)$

For each relation $relEW(x, y)$ in East-West Network:

- If a relation $relNS(x, y)$ exists in North-South Network then:
  - if *East* $\in relEW(x, y)$ AND *Horizontal-East* $\in relNS(x, y)$ add in ontology $East_{CSD9}(x, y)$.
  - if *West* $\in relEW(x, y)$ AND *Horizontal-West* $\in relNS(x, y)$ add in ontology $West_{CSD9}(x, y)$.
  - if *Vertical-North* $\in relEW(x, y)$ AND *North* $\in relNS(x, y)$ add in ontology $North_{CSD9}(x, y)$.
  - if *Vertical-South* $\in relEW(x, y)$ AND *South* $\in relNS(x, y)$ add in ontology $South_{CSD9}(x, y)$.
  - if *East* $\in relEW(x, y)$ AND *North* $\in relNS(x, y)$ add in ontology $NorthEast_{CSD9}(x, y)$.

  - if *East* $\in relEW(x, y)$ AND *South* $\in relNS(x, y)$ add in ontology $SouthEast_{CSD9}(x, y)$.
  - if *West* $\in relEW(x, y)$ AND *North* $\in relNS(x, y)$ add in ontology $NorthWest_{CSD9}(x, y)$.
  - if *West* $\in relEW(x, y)$ AND *South* $\in relNS(x, y)$ add in ontology $SouthWest_{CSD9}(x, y)$.
  - if *Identical-Vertical* $\in relEW(x, y)$ AND *Identical-Horizontal* $\in relNS(x, y)$ add in ontology $Identical_{CSD9}(x, y)$.

- else:
  - if *East* $\in relEW(x, y)$ add in ontology:
    * $East_{CSD9}(x, y)$
    * $NorthEast_{CSD9}(x, y)$
    * $SouthEast_{CSD9}(x, y)$
  - if *West* $\in relEW(x, y)$ add in ontology:
    * $West_{CSD9}(x, y)$
    * $NorthWest_{CSD9}(x, y)$
    * $SouthWest_{CSD9}(x, y)$
  - if *Vertical-North* $\in relEW(x, y)$ add in ontology:
    * $North_{CSD9}(x, y)$
  - if *Vertical-South* $\in relEW(x, y)$ add in ontology:
    * $South_{CSD9}(x, y)$

The Re-constructor component using the above rules, combines information in both networks and generates the following final directional relations, which are similar to the relations inferred using the CSD-9 model.

- Object1 (SE, SW, S) Object3
- Object2 (S, SE, SW, E) Object4
- Object1 (S, SE, SW, E) Object4

Chapter 4

Evaluation

The purpose of the experimental evaluation is to demonstrate the improved performance of CHOROS 2.0 (and of its variants implementing the optimizations discussed in Chapter 3) over PelletSpatial [2], CHOROS 1.0 [7] and SOWL [12], a spatial reasoner implemented in SWRL. We carried-out two different sets of experiments corresponding to measurements of performance in the average and the worst case. The average case performance is encountered when less than n^2 relations are inferred from input set of n locations. In our experiments, exactly kn relations (k=8 in the case or RCC-8 and k=9 in the case of CSD-9) are asserted. This is for example the case of a random set of objects. Accordingly, the worst case performance is encountered when the number of asserted relations are in the order of n^2. This is for example the case of objects given in a certain arrangement (i.e., each one is N of another or inside the one another). In all experiments we compare the running time of the competitor reasoner implementations as a function of the number of instances (regions or locations) in the ontology.

4.1 Theoretical Evaluation

The complexity of an algorithm is usually measured in terms of the worst case running time or memory consumption. Running time as well as memory consumption of an algorithm depends on the size n of its input and can be expressed as a function $f(n)$. The asymptotic behavior of f is specified in terms of the big O notation which gives an upper bound on the running time.

Previous works have shown that path consistency has $O(n^5)$ time worst case complexity (with n being the number of individuals) and is sound and complete [3]. Clearly, this upper bound is pessimistic, since the overall number of iterations of path-consistency algorithm may be lower than $O(n^2)$ because an inconsistency detection may terminate the reasoning process early, or the asserted relations may yield a small number of inferences.

In CHOROS as well as in PelletSpatial path consistency has $O(n^3)$ worst time complexity [7]. In our implementation any spatial object can be related with every

other object with one basic spatial relation. Between n objects, at most $(n-1)^2$ relations can be asserted. In the most general case where disjunctive relations are supported in addition to the basic ones, any spatial object can be related with every other object by at most k relations, where k is the size of the set of supported relations (maximum k = 8 for RCC relations, and k = 9 for CSD relations) [12]. Therefore, at most $O(kn^2)$ relations can be asserted into the Constraint Networks, for n spatial objects.

4.2 Experimental Evaluation

To evaluate performance of CHOROS 2.0 reasoner, we run a series of experiments. The purpose of these experiments is to present runtime efficiency of reasoning engine against the size of the datasets as well as present their dependance on the type of spatial information at hand. All experiments were carried-out on a home PC, with Intel Core i5 CPU at 2.60 GHz, 4 GB RAM and Windows 8 operating system.

As datasets, simple spatial ontologies are used, containing spatial individuals whose numbers ranges from 10 to 100. In order to prevent inconsistencies, every individual can be spatially related with only one individual. Performance of CHOROS 2.0 reasoner is measured both in the average as well as in the worst cases.

In the average case, which is the most common, every spatial individual can be related spatially to only one spatial individual. Spatial relations between individuals are assigned randomly from the sets of topological or directional relations. For the full range of datasets, results are derived from the average over 10 measurements.

On the other hand, in the worst case the reasoning process on spatial relations infer relations between all spatial individuals. For this, individuals are selected in a sequence so that, each one is included inside the next (i.e., R_i *NTPP* R_{i+1}) in the case of topological representation and, each one is in the same direction (e.g., *North*) with respect to its next (i.e., R_i *N* R_{i+1}) in the case of directional representation.

In the following, performance of various implementations of CHOROS 2.0 reasoner is presented, compared with other reasoning engines, which also support reasoning for both topological as for directional qualitative data, such as CHOROS and SOWL.

4.2.1 CSD Experiments

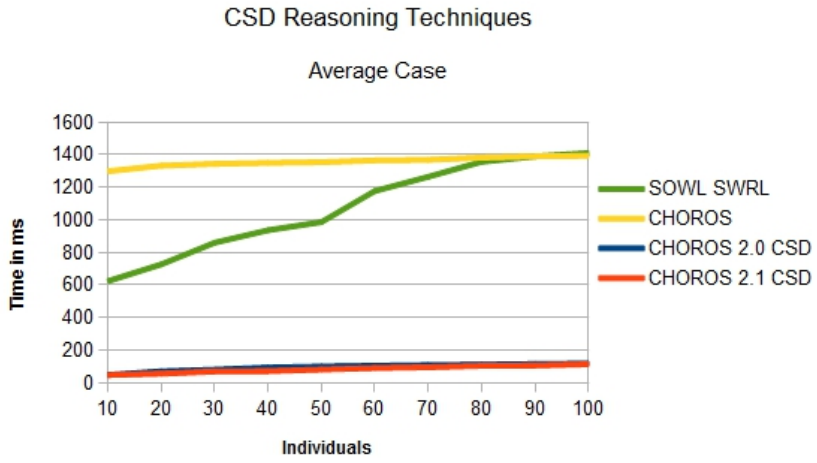
CHOROS 2.0 implements two different approaches for the CSD model:

- CHOROS 2.0 is based on the CSD-9 calculus, by applying consistency checking on the 9 basic CSD relations.
- CHOROS 2.1 is based on consistency checking over 8 basic CSD relations, as directional relation *IdenticalTo* is replaced by OWL axiom *sameAs*.

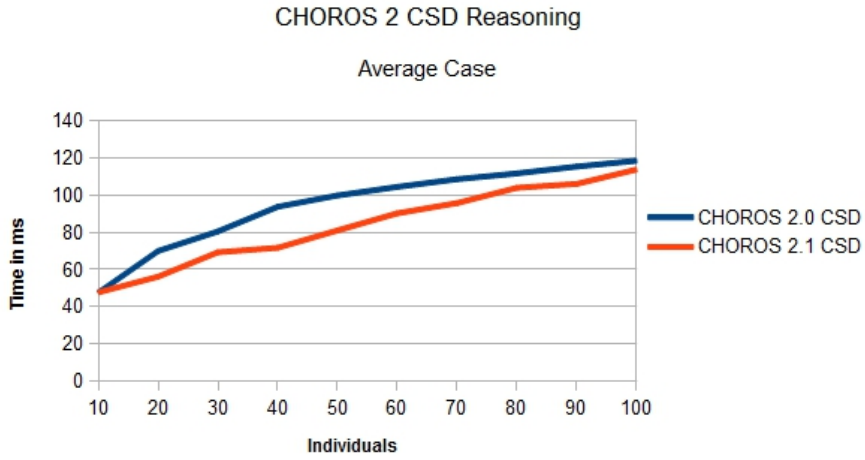
Figure 4.1 shows performance of CHOROS 2.0 and CHOROS 2.1 in the average case, while Figure 4.2 presents performance of both implementations of CHOROS 2 in the worst case. Moreover, both figures also present the performance of the respective implementations of CHOROS and SOWL.

As shown in Figures 4.1(a) and 4.2(a) both implementations of CHOROS 2 outperform CHOROS and SOWL. This significant improvement in performance lies in the fact that spatial reasoning in CHOROS 2 doesn't need to pre-compute the full composition table of basic relations, a time consuming process, as CHOROS does. On the contrary, CHOROS 2 need to store only compositions over the basic relations.

Regarding the representation of directional relations, figures 4.1(b) and 4.2(b) show that CHOROS 2.1 performs slightly better than CHOROS 2.0. These results are expected, since CHOROS 2.1 applies consistency checking on less basic relations than CHOROS 2.0, as directional relation *IdenticalTo* is replaced by OWL axiom *sameAs* handled by Pellet.

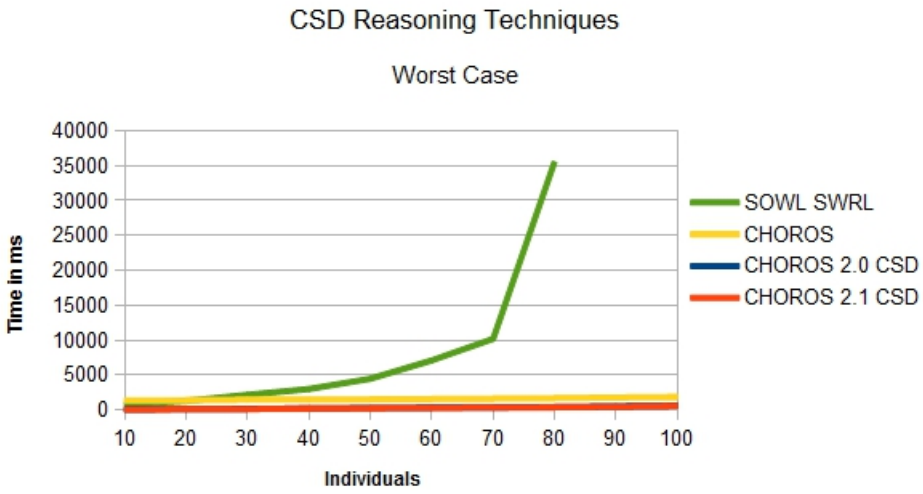


(a)

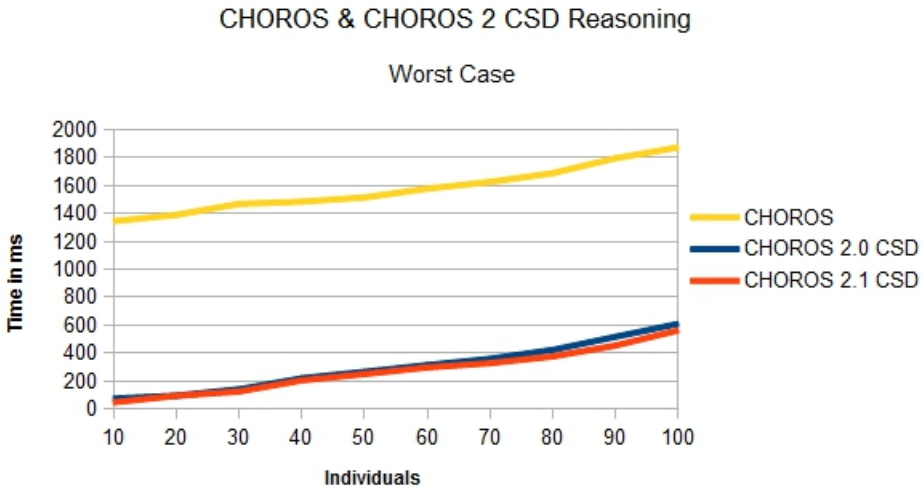


(b)

Figure 4.1: Average case performance of Reasoning over CSD relation sets



(a)



(b)

Figure 4.2: Worst case performance of Reasoning over CSD relation sets

4.2.2 RCC Experiments

In the following, we demonstrate the performance of CHOROS 2 reasoner on topological relations. Similar to directional model, CHOROS 2 presents two implementations of the RCC model:

- CHOROS 2.0 is based on the RCC-8 calculus, by applying consistency checking on the 8 basic RCC relations.
- On the other hand, CHOROS 2.1 is based on consistency checking on 7 basic RCC relations, as topological relation *EQ* is replaced by OWL axiom *sameAs*.

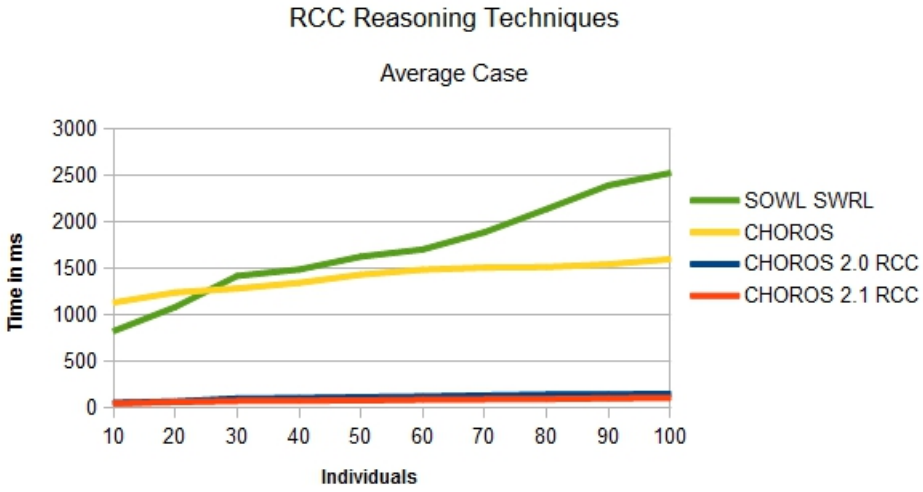
Average case performance of both approaches of CHOROS 2 is presented in Figure 4.3, while Figure 4.4 illustrates the corresponding measurements in the worst case. Relevant implementations of CHOROS and SOWL have also been added in order to be compared with our implementation.

As in the case of directional relations, we find from Figures 4.3(a) and 4.4(a) that both implementations of CHOROS 2 outperform CHOROS and SOWL. It turns out that the replacement of the full composition table with a dynamic approach in the handling of compositions of complex relations, is crucial to the performance of CHOROS 2.

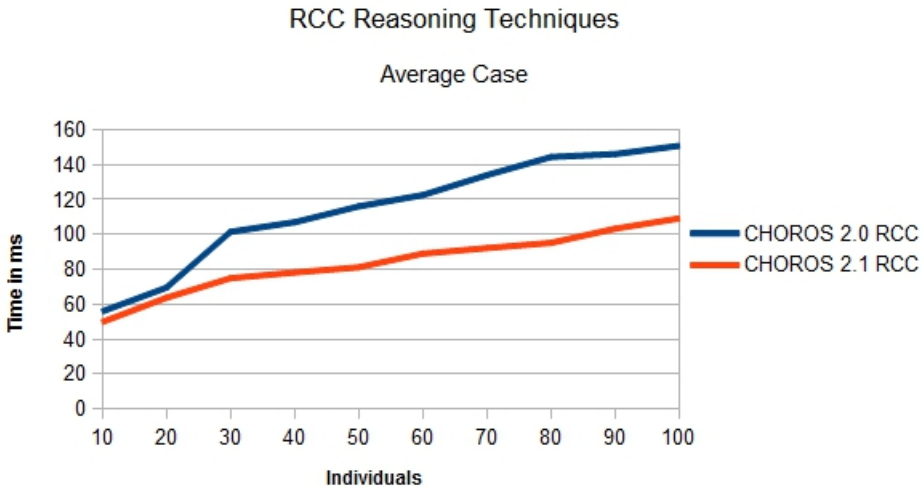
Similar conclusions also apply in the case CHOROS 2.0 and CHOROS 2.1. Figures 4.3(b) and 4.4(b) show that reduction of basic RCC relations improve the performance of CHOROS 2. It appears that this optimization affects more the RCC model than the CSD, comparing Figures 4.4(b) and 4.2(b).

Comparing the reasoning process on both spatial models, we notice that reasoning on CSD relations is a faster compared to reasoning on RCC relations. This is mainly due to the characteristic of the CSD model, where every relation has its inverse relation, as seen in figure 2.2. It is an important factor as indirectly accelerates the process of reasoning. In CSD model, compositions of inverse relations yields all possible relations, which don't provide any information, causing the algorithm to skip them and thus the process to be applied in fewer relations.

On the other hand, the RCC model does not feature this characteristic. Referring to the RCC basic relations in Figure 2.1, we notice that only relations *NTPP* and *TPP* can be regarded as the inverse relations of *NTPPi* and *TPPi* respectively. However, in most cases as illustrated in table 3.1, compositions of these inverse relations don't yield all possible relations of the model. As a result more relations are inserted into the network to be processed by the path consistency algorithm.

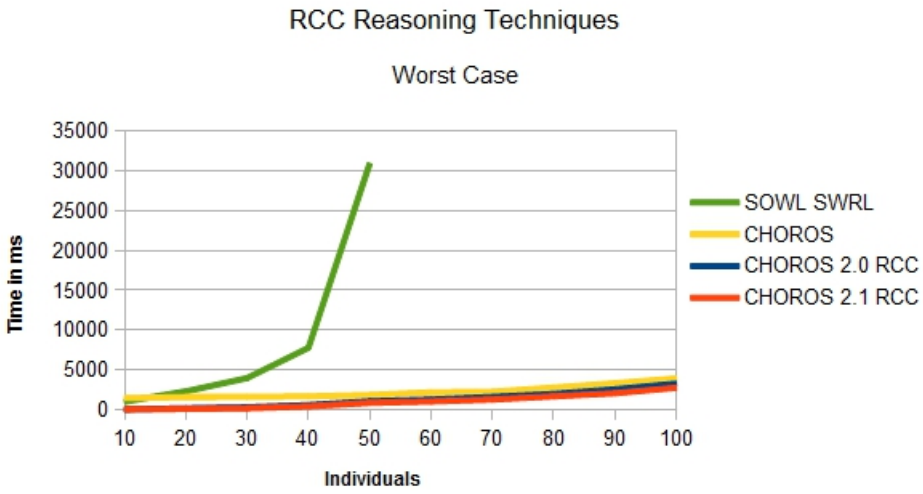


(a)

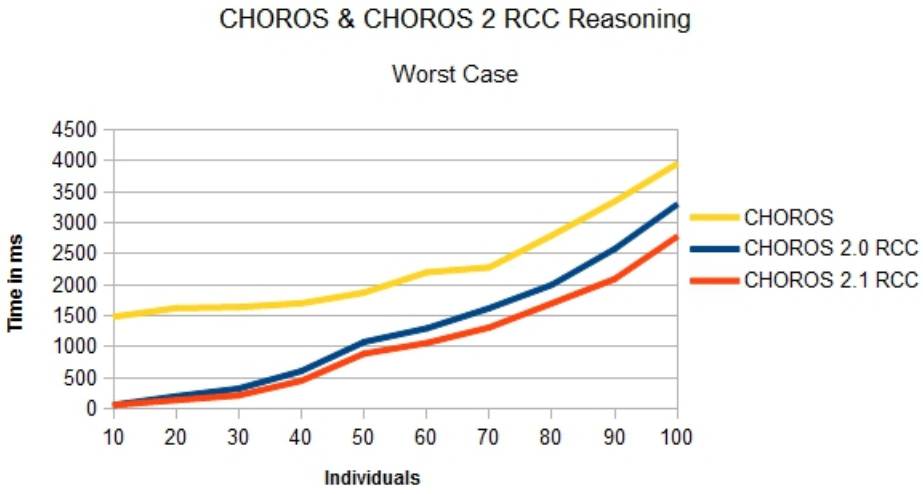


(b)

Figure 4.3: Average case performance of Reasoning over RCC relation sets



(a)



(b)

Figure 4.4: Worst case performance of Reasoning over RCC relation sets

4.2.3 CSD Decomposition Experiments

Apart from reasoning over the CSD model, CHOROS 2 also examines the performance of reasoning over the decomposition of directional relations. Reasoning is applied as described in section 3.3, and both reasoner engines are implemented as threads in order to enable ”parallel” and concurrent execution of their tasks. This reasoning technique is referred to as CHOROS 2.2.

Figure 4.5 presents the average case performance of CHOROS 2.2, while Figure 4.6 illustrates the worst case performance of the new CSD reasoner. Moreover, we compare CHOROS 2.2 with CHOROS 2.1, technique exhibiting the best performance on the CSD model.

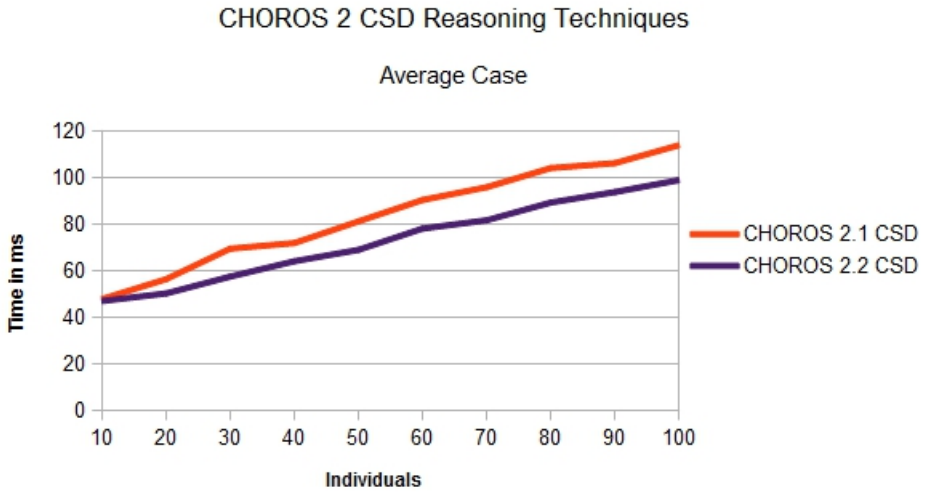


Figure 4.5: Average Case performance of Reasoning on Decomposed CSD relations

As Figure 4.5 shows, despite the fact that CHOROS 2.2 handles more relations than CHOROS 2.1, it performs better in the average case. As mentioned before, both reasoners of CHOROS 2.2 apply the path-consistency algorithm in smaller sets of basic relations than the CSD model, which is faster overall.

However, in the worst case (Figure 4.5) roles are reversed, as CHOROS 2.1 performs better than CHOROS 2.2. This is because each reasoner of CHOROS 2.2 infers relations between all spatial individuals, and thus to infer twice as many relations as CHOROS 2.1 does.

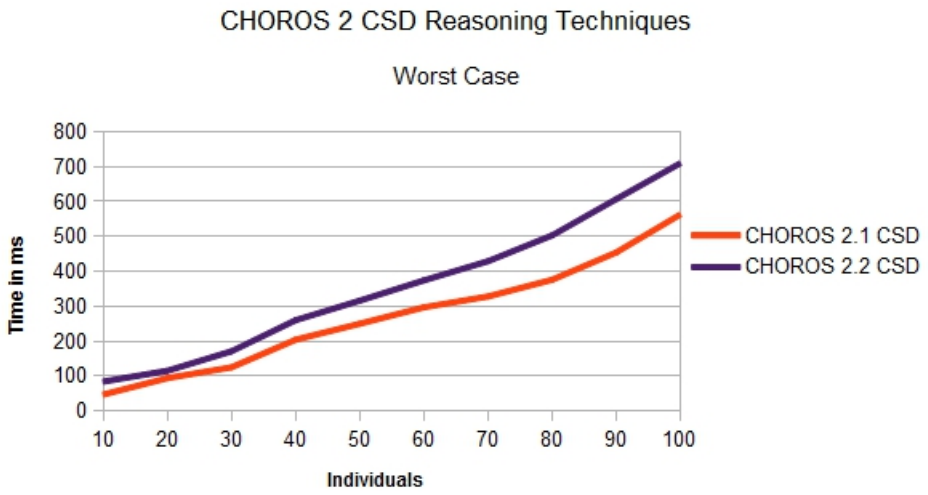


Figure 4.6: Worst Case performance of Reasoning on Decomposed CSD relations

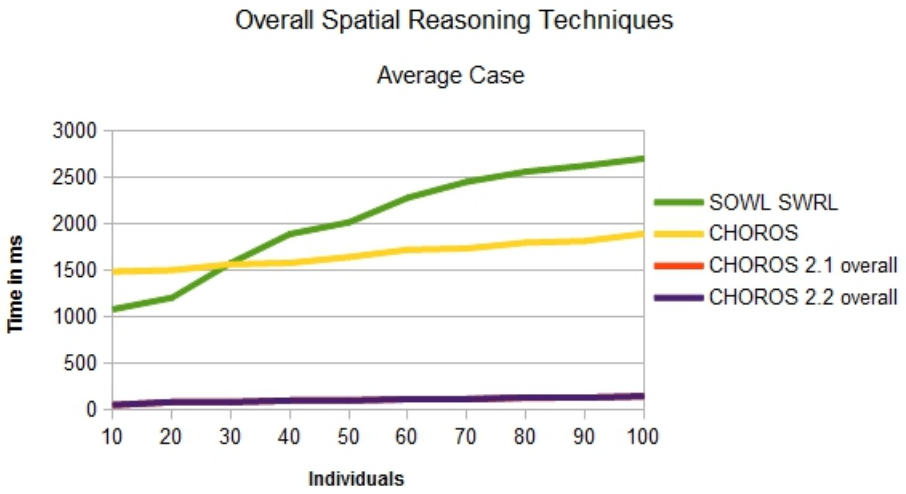
4.2.4 Overall Spatial Experiments

In previous sections, we presented performance of various implementations of CHOROS 2 in either directional or topological data. In this section we demonstrate performance of CHOROS 2 reasoning over spatial data containing both CSD and RCC relations. Figure 4.7 illustrates the average case performance of CHOROS 2, and Figure 4.8 the worst case performance of reasoning. In the following experiments the competitor implementations are:

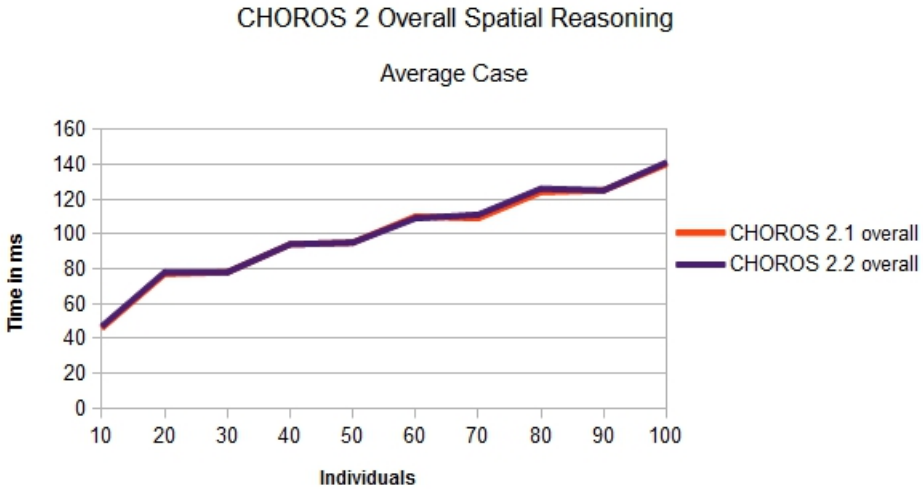
- CHOROS 2.1 applies reasoning on the 8 basic relations of CSD model, as well as on the 7 basic relations of RCC model.
- CHOROS 2.2 applies reasoning on the decomposition of directional relations, as well as on the 7 basic relations of RCC model.
- CHOROS applies reasoning on the 8 basic relations of CSD model, as well as on the 8 basic relations of RCC model.
- SOWL

Notice that each reasoner is implemented as thread in order to enable ”parallel” and concurrent execution of their tasks.

Figures 4.7 and 4.8 illustrate that all implementations of CHOROS 2 outperform CHOROS and SOWL. It is worth mentioning, that performance of CHOROS 2 depends mainly the performance of RCC reasoner (Figures 4.7(b) and **??**), as it is proved to be the most time consuming process in spatial reasoning.

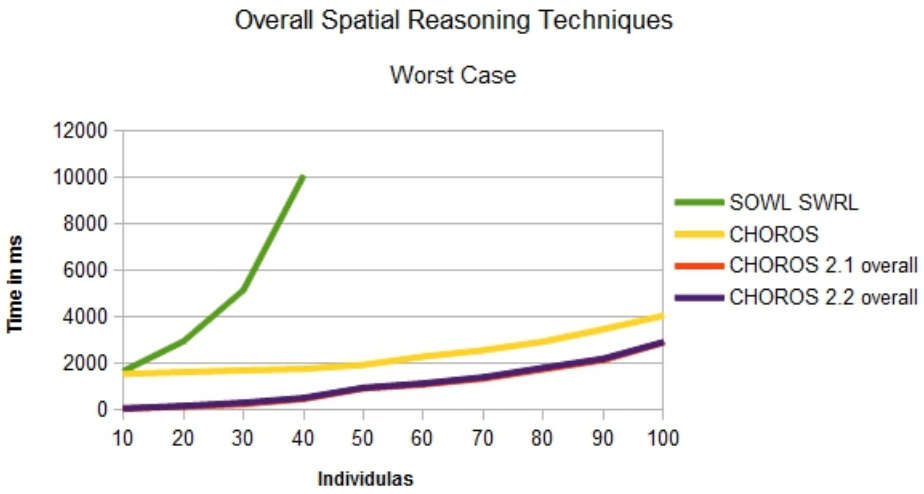


(a)

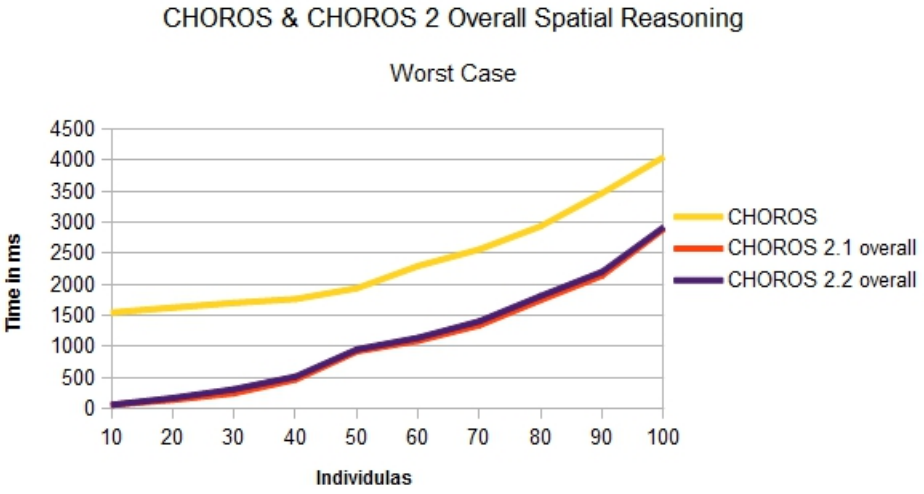


(b)

Figure 4.7: Average case performance of Reasoning on combined RCC and CSD relations sets



(a)



(b)

Figure 4.8: Worst case performance of Reasoning on combined RCC and CSD relations sets.

4.2.5 Case Study - TUC Spatial Ontology

The ”TUC spatial ontology” [7] is a mechanism to describe data related to spatial entities of the University campus of Technical University of Crete (TUC).

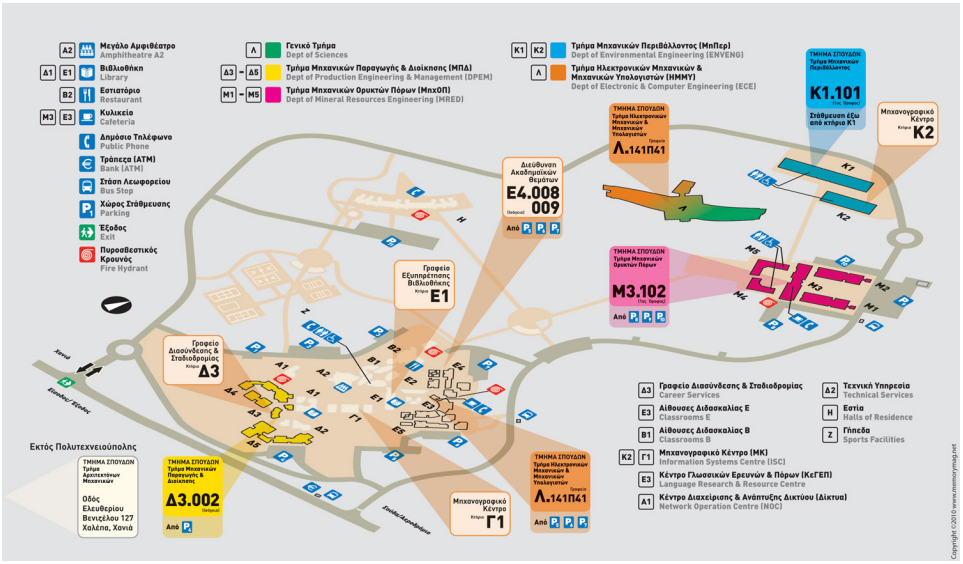


Figure 4.9: Campus Map - Technical University of Crete

Figure 4.9 illustrates the campus map of TUC. The ”TUC spatial ontology” is implemented in OWL. It consists of classes, properties and individuals. Table 4.2 shows all classes, gives a brief description and lists some individuals that belong to each class. Table 4.3 and Table 4.4 illustrates object properties and datatype properties of the ontology.

We use the ”TUC spatial ontology” in order to evaluate the performance of spatial reasoning techniques described in previous sections, on a real application (Table 4.1).

	SOWL SWRL	CHOROS	CHOROS 2.1	CHOROS 2.2
Response (time in ms)	8313	2312	422	407

Table 4.1: Response time of reasoning techniques on the ”TUC spatial ontology”

Class			Description	Individual
Region			Superclass of the ontology. Every individual of this ontology represents a region. Individuals not belonging in any of the other subclasses are defined here directly.	Campus,the road inside the campus and the gate.
	Bus Stop		Bus stops serving a number of destinations throughout the campus.	Bus stop serving ECE department and 5 more bs.
	Classrooms		Groups of classrooms or classrooms-buildings. Parts of a classroom group should not be defined as an individual of this class.	Amphitheatre, group E and 4 more groups.
	Department		Departments of the Technical University of Crete.	ECE, DPEM and 4 more departments.
	Facilities		Facilities of the campus.	
		Sports	Facilities of the campus for sports activities.	Basketball courts and 4 more courts.
		Residence	Facilities of the campus for student's accommodation.	Hestia.
		Food	Facilities of the campus for food serving.	Restaurant, 3 Cafeterias.
	Parking		Areas that serve parking throughout the campus.	Parking area 1-14.
	Services		Central offices, libraries and other buildings providing services.	NOC, ISC 1-2, Career Services and 5 more.

Table 4.2: Classes and instances in the TUC spatial ontology.

Object Property		Description	Subject class to be applied (domain)	Object class to be applied (range)
CSDDefined		Cone shaped directional relations between centroids of regions.	Region, Bus Stop, Classrooms, Department, Facilities, Parking Services.	Region, Bus Stop, Classrooms, Department, Facilities, Parking Services.
	northOf			
	northEastOf			
	eastOf			
	southEastOf			
	southOf			
	southWestOf			
	westOf			
	northWestOf			
	identicalTo			
RCCDefined		RCC-8 topological relations between regions.	Region, Bus Stop, Classrooms, Department, Facilities, Parking Services.	Region, Bus Stop, Classrooms, Department, Facilities, Parking Services.
	disconnected-From			
	equalsTo			
	externally-ConnectedTo			
	hasNon-Tangential-ProperPart			
	hasTangential-ProperPart			
	nonTangential-ProperPartOf			
	partially-Overlaps			
	tangential-ProperPartOf			
hasParking		Non-spatial relation representing that a region is served by a parking area.	Classrooms, Department, Facilities and Services.	Parking.

Table 4.3: Object Properties in the TUC spatial ontology.

Datatype Property	Description	Subject class to be applied (domain)	Object class to be applied(range)
hasName	Relation defining the full name of a region	Region, Bus Stop, Classrooms, Department, Facilities, Parking Services.	Type "String"
hasCodeName	Relation defining a code name of a region according to campus Map	Region, Bus Stop, Classrooms, Department, Facilities, Parking Services.	Type "String"

Table 4.4: Data Properties in the TUC spatial ontology

Chapter 5

Conclusion and Future Work

CHOROS 2.0 is a reasoning engine for spatial information in OWL expressed using RCC and CSD models. It is the result of an investigation on suggesting potential improvements over CHOROS 1.0 [7]. It works with all RCC and CSD relations in combination with standard RDF/OWL semantic relations in an OWL ontology which are handled by Pellet. It is practically an extension over PelletSpatial for handling directional in addition to topologic information. CHOROS 2.0 inherits all features of CHOROS 1.0 and in addition suggests certain improvements summarized below:

- Suggests computing compositions (i.e., disjunctions) of basic relation during reasoning (i.e., on the fly) rather than storing them in a table in memory which results in faster reasoning times.
- Suggests reduction of basic relations of both RCC and CSD models, by replacing EQUAL relation with OWL axiom "sameAs".
- In regards to CSD relations in particular, suggests a reasoning approach based on the decomposition of CSD-9 relations into pairs of directional relations. This approach results in the faster reasoning in certain cases (i.e., in the average case).
- Introduces a software component for updating the ontology with the results of reasoning. These new spatial inferences can be stored in an updated ontology which can be re-used or queried using SPARQL.
- Introduces a software component for encoding quantitative spatial information into directional relations.

Evaluation results demonstrated that all implementations of CHOROS 2.0 outperform CHOROS 1.0 and SOWL. Regarding the directional relations, it appears that reasoning over the decomposition of directional relations performs better than the CSD model in the average case. Comparing the two spatial models (i.e., RCC and CSD), the experimental results demonstrate that the RCC model is the most

time consuming model in spatial reasoning mostly determining the performance of CHOROS 2.0 reasoner overall.

Regarding future work, there are still issues worth considering further including:

- Investigate potential improvements of reasoning over the decomposition of directional relations. Reasoning is performed on smaller sets of basic relations, which may result in faster reasoning.
- Extend the functionality of quantitative parser to generate topologic relations between spatial entities expressed in numerical values. Given the borders of regions, we can identify their topology and encode it in terms of RCC relations.
- Investigate on more effective reasoning methods for smaller sets of basic relations or for tractable sets of relations such as those identified in [12].
- Support OWL 2 restrictions on spatial relations (e.g., ”a country A borders with exactly 3 other countries”).
- Examine the performance of CHOROS 2.0 on real applications and ontologies. In this work, since application data sets or ontologies are not available to us, the experimental evaluation of the performance of reasoning is carried-out using mainly synthetic data sets and simple ontologies.

Bibliography

[1] D. A. Randell, Z. Cui, and A. Cohn, "A Spatial Logic Based on Regions and Connection," in KR92. Principles of Knowledge Representation and Reasoning: Proceedings of the Third International Conference, B. Nebel, C. Rich, and W. Swartout, Eds. San Mateo, California: Morgan Kaufmann, 1992, pp. 165-176.

[2] M. Stocker, and E. Sirin "PelletSpatial: A Hybrid RCC-8 and RDF/OWL Reasoning and Query Engine" In: CEUR Workshop Proceedings, vol. 529-OWLED 2009, pp. 2-31, 2009.

[3] J. Renz and B. Nebel, "Qualitative Spatial Reasoning Using Constraint Calculi." in Handbook of Spatial Logics, M. Aiello, I. Pratt-Hartmann, and J. van Benthem, Eds. Springer, 2007, pp. 161-215.

[4] A. G. Cohn and S. M. Hazarika, "Qualitative Spatial Representation and Reasoning: An Overview." Fundam. Inform., vol. 46, no. 1-2, pp. 1-29, 2001.

[5] R. H. Guting, "An Introduction to Spatial Database Systems." The VLDB Journal, vol. 3, no. 4, pp. 357-399, Oct. 1994.

[6] I. B. Arpinar, A. P. Sheth, C. Ramakrishnan, E. L. Usery, M. Azami, and M.-P. Kwan, "Geospatial Ontology Development and Semantic Analytics." T. GIS, vol. 10, no. 4, pp. 551575, 2006.

[7] G. Christodoulou, E. Petrakis E., and S. Batsakis. "Qualitative Spatial Reasoning using Topological and Directional Information in OWL", 24^{th} International Conference on Tools with Artificial Intelligence (ICTAI 2012), Athens, Greece, November 7-9, 2012.

[8] D. R. Montello and A. U. Frank, "Modeling Directional Knowledge and Reasoning in Environmental Space: Testing Qualitative Metrics," in The Construction of Cognitive Maps (GeoJournal Library), P. Juval, Ed. Kluwer Academic Publishers, 1996, pp. 321-344.

[9] J. Renz and D. Mitra, "Qualitative Direction Calculi with Arbitrary Granularity." in PRICAI, ser. Lecture Notes in Computer Science, C. Zhang, H. W. Guesgen, and W.-K. Yeap, Eds., vol. 3157. Springer, 2004, pp. 65-74.

[10] B. Nebel and H.-J. Brckert, ”Reasoning about Temporal Relations: A Maximal Tractable Subclass of Allens Interval Algebra.” in AAAI, B. Hayes-Roth and R. E. Korf, Eds. AAAI Press / The MIT Press, 1994, pp. 356-361.

[11] Stravoskoufos K. ”SOWL QL : Querying Spatio-Temporal Ontologies In OWL 2.0”, Master’s Thesis, Department of Electronic and Computer Engineering, Technical University of Crete, April2013.

[12] Sotiris Batsakis, Euripides G.M. Petrakis, ”SOWL: A Framework for Handling Spatio-Temporal Information in OWL 2.0”, 5th International Symposium on Rules: Research Based and Industry Focused (RuleML’ 2011), Barcelona, Spain, July 19-21, 2011, pp. 242-249.

[13] P. van Beek, and R. Cohen ”Exact and approximate reasoning about temporal relations.” Computational intelligence, Vol 6(3), pp. 132-147, 1990

[14] S. Batsakis, ”Reasoning over 2D and 3D Directional Relations in OWL: A Rule-Based Approach”, in Proceedings of RuleML 2013, Seattle, USA, July 11-13, 2013

[15] E. Sirin, B. Parsia, B. C. Grau, A. Kalyanpur, and Y. Katz, ”Pellet: Apractical owl-dl reasoner,” Web Semant., vol. 5, no. 2, pp. 51-53, Jun. 2007.

[16] Eric Prud’hommeaux, Andy Seaborne. SPARQL Query Language for RDF. http://www.w3.org/TR/rdf-sparql-query/

[17] B. Nebel, and H.J. Burckert ”Reasoning about Temporal Relations: A Maximal Tractable Subclass of Allen’s Interval Algebra” Journal of the ACM (JACM), Vol.42(1), pages:43-66, 1995.

[18] J. Renz and B. Nebel. Efficient Methods for Qualitative Spatial Reasoning. In Proc. of the 13th European Conference on Artificial Intelligence (ECAI98), 1998.

[19] B. Bennet. Knowledge Representation and Reasoning: Compositional Reasoning, 2007. Lecture notes, School of Computing, University of Leeds.

[20] Eleftherios Anagnostopoulos, Sotiris Batsakis, Euripides G.M. Petrakis, ”CHRONOS: A Reasoning Engine for Qualitative Temporal Information in OWL”, 16th International Conference on Knowledge-Based and Intelligent Information & Engineering Systems (KES’2013), 9-11 September 2013, Kitakyushu, Japan