NATIONAL AND KAPODISTRIAN UNIVERSITY OF ATHENS

SCHOOL OF SCIENCE

DEPARTMENT OF INFORMATICS AND TELECOMMUNICATIONS

THESIS PLACEMENT

Plug-ins for Spatial Reasoning and Querying in Protégé using the PelletSpatial Reasoner

Supervisors: **Emmanouil Koubarakis**, Professor, N.K.U.A
Charalampos Nikolaou, PhD Candidate, N.K.U.A

ATHENS

OCTOBER 2012

ΕΘΝΙΚΟ ΚΑΙ ΚΑΠΟΔΙΣΤΡΙΑΚΟ ΠΑΝΕΠΙΣΤΗΜΙΟ ΑΘΗΝΩΝ

ΣΧΟΛΗ ΘΕΤΙΚΩΝ ΕΠΙΣΤΗΜΩΝ

ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ ΚΑΙ ΤΗΛΕΠΙΚΟΙΝΩΝΙΩΝ

ΠΤΥΧΙΑΚΗ ΕΡΓΑΣΙΑ

Πρόσθετα για Χωρικό Συμπερασμό και Χωρικά Ερωτήματα στο Protégé με χρήση του Μηχανισμού Συμπερασμού PelletSpatial

Επιβλέποντες: **Εμμανουήλ Κουμπαράκης**, Καθηγητής, Ε.Κ.Π.Α
Χαράλαμπος Νικολάου, Υποψήφιος Διδάκτορας, Ε.Κ.Π.Α

ΑΘΗΝΑ

ΟΚΤΩΒΡΙΟΣ 2012

THESIS PLACEMENT

Plug-ins for Spatial Reasoning and Querying in Protégé using the PelletSpatial Reasoner

Nikolaos I. Begetis

AM: 1115200700281

SUPERVISORS:

Emmanouil Koubarakis , Professor, N.K.U.A
Charalampos Nikolaou , PhD Candidate, N.K.U.A

ΠΤΥΧΙΑΚΗ ΕΡΓΑΣΙΑ

Πρόσθετα για Χωρικό Συμπερασμό και Χωρικά Ερωτήματα στο Protégé με χρήση του Μηχανισμού Συμπερασμού PelletSpatial

Νικόλαος Ι. Μπεγέτης

ΑΜ: 1115200700281

ΕΠΙΒΛΕΠΟΝΤΕΣ :

Εμμανουήλ Κουμπαράκης , Καθηγητής, Ε.Κ.Π.Α
Χαράλαμπος Νικολάου , Υποψήφιος Διδάκτορας, Ε.Κ.Π.Α

ABSTRACT

In this thesis, we present the ProtégéPelletSpatial plugin for Protégé and the PelletSpatial extention of Pellet. ProtegePelletSpatial is a qualitative spatial reasoning engine implemented in Java on top of Pellet, used as a plugin in the Protégé Semantic Web technology. PelletSpatial reasons over spatial data represented with the Region Connection Calculus (RCC-8) as long as provides consistency checking and query answering. It supports all RCC-8 relations as well as standard RDF/OWL semantic relations, and comes with a hybrid architecture which is based on a composition table that implements two RCC reasoners: (a) A reasoner based on the semantics preserving translation of RCC relations to OWL-DL class axioms and (b) a reasoner based on the RCC composition table that implements a path-consistency algorithm. This thesis intention is to add the ProtegePelletSpatial plugin in Protégé. This plugin presents in details results after spatial ontology classification, following the standard format for state of art qualitative spatial reasoners that has been proposed. As a case study in our experiments for finding the time complexity we used the "Administrative Geography Ontology" so that to provide a qualitative spatial description of the spatial regions. Finally, we discuss and evaluate possible optimizations of PelletSpatial engine and present a user guide for installation and usage of our ProtegePelletSpatial Protégé plugin.

SUBJECT AREA: Semantic Web

KEYWORDS: RDF/RDFS, OWL, SPARQL, spatial reasoning, pellet spatial, protégé 4.x

ΠΕΡΙΛΗΨΗ

Στα πλαίσια αυτής της πτυχιακής εργασίας παρουσιάζουμε το ProtegePelletSpatial plugin που χρησιμοποιεί τον PelletSpatial για το Protégé Semantic Web Framework. Ο PelletSpatial είναι μία λογική μηχανή συμπερασμού για χωρικά δεδομένα υλοποιημένη σε γλώσσα Java πάνω από την διεπαφή του συστήματος του Pellet, που τη χρησιμοποιούμε για να επεκτείνουμε με μία πρόσθετη λειτουργία (plugin) το Protégé Framework. Ο κύριος σκοπός του PelletSpatial είναι να προσθέσει στο ολοκληρωμένο σύστημα Pellet τη δυνατότητα ο Pellet να μπορεί να χρησιμοποιήσει, όχι μόνο δεδομένα εκφρασμένα σε απλή OWL-DL γλώσσα οντολογιών, αλλά εκφρασμένα και με συγκεκριμένες επιπλέον σχέσεις που φανερώνουν κάποιες χωρικές ιδιότητες των δεδομένων μεταξύ τους, τα οποία αναπαρίστανται με το Region Connection Calculus (RCC-8) pr'otupo, το οποίο εκφράζει τα δεδομένα ως περιοχές και τα συνδέει με χωρικές τοπολογικές σχέσεις. Ο PelletSpatial υποστηρίζει το πρότυπο των σημασιολογικών σχέσεων σε RDF/OWL και αποτελείται από μία υβριδική αρχιτεκτονική η οποία είναι βασισμένη σε ένα πίνακα σύνθεσης που υλοποιεί δύο RCC μηχανές συμπερασμού: (α) Μία βασισμένη σε μεταφρασμένες σημασιολογίες μετατρέποντας RCC σχέσεις, και διατηρώντας τη σημασιολογία τους, σε αντίστοιχα OWL-DL αξιώματα τάξης και (β) μία μηχανή συμπερασμού βασισμένη στο RCC πίνακα σύνθεσης στον οποίο ενεργεί ένας αλγόριθμος με συνέπεια μονοπατιού. Σκοπός της παρούσας πτυχιακής εργασίας είναι να προσθέσει τον PelletSpatial με τη μορφή plugin στο Protégé. Έτσι το ProtegePelletSpatial plugin που υλοποιήσαμε παρουσιάζει με λεπτομέρειες τα αποτελέσματα που προκύπτουν έπειτα από κάθε είσοδο δεδομένων στη χωρική μηχανή συμ-

περασμού ή στη χωρική μηχανή επερωτήσεων, ακολουθώντας μία καθορισμένη μορφοποίηση που έχει προταθεί. Σαν μελέτη περίπτωσης στα δεδομένα μας για την εύρεση και επιβεβαίωση της χρονικής πολυπλοκότητας χρησιμοποιήσαμε το "Administrative Geography Ontology", που είναι μία οντολογία που αναπαριστά τις διοικητικές περιφέρειες της Μεγάλης Βρετανίας και τις υπηρεσίες τους με βάση τη θέση τους στο χάρτη. Εν κατακλείδι, συζητάμε και εκτιμούμε κάποιες πιθανές βελτιώσεις της μηχανής συμπερασμού του PelletSpatial και παρουσιάζουμε έναν οδηγό εγκατάστασης και χρήσης του Protégé plugin που φτιάξαμε.

ΘΕΜΑΤΙΚΗ ΠΕΡΙΟΧΗ: Σημασιολογικός Ιστός (Semantic Web)

ΛΕΞΕΙΣ - ΚΛΕΙΔΙΑ : RDF/RDFS, OWL, SPARQL, spatial reasoning, pellet spatial, protégé 4.x

To Charlie angel,

ACKNOWLEDGEMENTS

First of all, I would like to express my sincere thanks to my advisor, Prof. Manolis Koubarakis for his deep glance in my interests and for assigning me this thesis which really excited me. I should also express my sincere thanks to the PhD candidate colleague Charalampos Nikolaou for his support and guidance throughout these months in accomplishing this thesis. Great thanks to some of my best professors for inspiring me and disseminating me with an enthusiasm and a desire for perfection in my scientific field. These are: Prof. Manolis Koubarakis, Prof. Ioannis Ioannidis, Associate Prof. Panagiotis Rontogiannis, Prof. Koutsoupias Elias, Prof. Ioannis Emiris, Prof. Alex Delis, and Assistant Prof. Panagiotis Stamatopoulos. I would also like to give many thanks to all my colleagues in the Department of Informatics and Telecommunications for their continuous support and help.

Furthermore, I will always be thankful to my family for all their support all these years and for having educated me properly from my infancy despite the financial problems that existed occasionally in times. Special thanks to my mother, Irene, and my father, Yannis, who are both teachers of computer science in secondary education, and whom I consider as my mentors in computer science and even more in my whole life, and also to my younger sisters, Sophia, Eleni and Demetra, from who I receive daily their endless love and encouragement and for who I would like to engrave the path for them to go. Special thanks also go to my whole parent tree who have been for me the den where I habituate to perch. My heartfelt thanks to my "novia", Mariana, for her unconditional support, love and encouragement. And last but not least, special thanks go to my school friends, who have taken part to my behaviour shaping in all these years. The support that everyone above has given me, in all my pursuits has been tremendous. I hope they are proud of me. And for me, I hope to worthy and to continue my work in research for my faculty under Prof. Ioannis Ioannidis supervision and in the two postgraduate programs I have been selected.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF ALGORITHMS

FOREWORD

This thesis was conducted in Athens, Greece during the period 2011 - 2012 under the supervision of the Prof. Manolis Koubarakis, Professor of Informatics and Telecommunications faculty of National and Kapodistrian University of Athens. It was performed in two periods. The first was dedicated in the java implementation of PelletSpatial as a plugin in the Protégé, and the second in the redaction of this thesis. Before you carry on with the document reading, I would like to point out that the material of the ProtegePelletSpatial implementation (source code, executable jar, examples) is included in the accompanying media (CD) deposited to the library. Hopefully, after some more modifications, this plugin will be uploaded to the Protégé online storage system.

CHAPTER 1

INTRODUCTION

Formal spatial representations have been studied extensively in the Database [2] and recently, the Semantic Web literature [3]. 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 "10Km away", "45 degrees North" etc.). A spatial ontology can also be defined without a coordinate reference system, such as in the case of topological relations [4].

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 [1]. For instance, there might be inconsistencies in spatial relations that will not be detected by an OWL reasoner or an OWL reasoner might not return all the answers to spatial queries since it cannot compute all spatial inferences. To deal with this problem, reasoning rules for various relation sets have been proposed [5], [6]. The most popular reasoning methods used in qualitative spatial reasoning are constraint based techniques adopted from previous work on temporal reasoning [5], [7]. Reasoning applies on sets of qualitative spatial relations which are jointly exhaustive and pairwise disjoint (i.e., between any two 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. For instance, if the binary relation R1 holds between entities A and B and the binary relation R2 holds between B and C, then the composition of R1 and R2 restricts the

possible relationships between A and C. Compositions of relations are usually pre-computed and stored in a composition table.

In this thesis, we are going to analyse PelletSpatial which supports qualitative spatial reasoning based on the Region Connection Calculus (RCC-8) logic formalism. PelletSpatial, extends the DL reasoning features of Pellet [8] with qualitative spatial reasoning capabilities. PelletSpatial supports consistency checking and answering SPARQL queries over a set of spatial relations and non-spatial semantic relations.

We provide a proof-of-concept implementation for the translation of RCC-8 relations to OWL-DL class axioms presented in [9]. To the best of our knowledge, there has been only a few implementations for this translation. In this thesis, we present our experimental evaluation of this approach by using the ProtégéPelletSpatial plugin we created.

1.1 Motivation

Qualitative reasoning is an approach for dealing with commonsense knowledge without using numerical computation. Instead, one tries to represent knowledge using a limited vocabulary such as qualitative relationships between entities or qualitative categories of numerical values. The motivation for using a qualitative approach is that it is considered to be closer to the way humans represent and reason about commonsense knowledge. Moreover, another aspect that motivated me is that it is possible to deal with incomplete knowledge.

1.2 Problem Definition

The problem of representing temporal relationships in a binary-limited representation is not new. Approaches to deal with this problem do exist but they all suffer from significant drawbacks (data redundancy, object proliferation, limited reasoning support). In addition, all, result in complicated ontologies compared with their static counterparts where all relations do not change in time (as every temporal relation is substituted by a set of binary OWL relations) [10].

It is a common practice to use Web Ontology Language (OWL) ontologies to describe spatial regions and relations between these regions, such as relative directional position or spatial containment and overlap. However, it is not possible to directly encode the semantics of these relations using the expressivity of OWL and the Description Logics (DL) that OWL is based

on. As a consequence, there might be inconsistencies in spatial relations that will not be detected by an OWL reasoner or an OWL reasoner might not return all the answers to spatial queries since it cannot compute all spatial inferences. But here comes the PelletSpatial to solve this problem by translating ontologies to spatial and non-spatial and giving them to two different reasoners to be classified.

OWL ontology editors such as Protégé [11] provide a well-suited environment allowing users to create or edit static OWL ontologies with binary relations. But, yet, even Pellet exists as a plugin for reasoning, it doesn't support spatial reasoning. So, this is exactly the problem that this work is dealing with. We are going to extend pellet function so as to classify spatial data too.

1.3 Background

Spatial information is typically encoded by means of a (binary) relation model of spatial entities. Most spatial calculi focus on a single aspect of space (e.g. topology, direction, distance or position [11]). Spatial relations between regions can be easily extracted from their surrounding minimum bounding rectangles (or their surrounding contours) by comparing their coordinates [12]. For computing directional relations in particular, it is convenient to approximate spatial entities such as regions, by points (e.g., by their centers of gravity). This is not sufficient for representing topological information, which encodes adjacency, overlap and containment relations between regions. Nevertheless, both representations may co-exist in a spatial representation (using one of them or both, is a design decision).

What is more, the most popular reasoning methods used in qualitative spatial reasoning are constraint based techniques adopted from previous work on temporal reasoning [13], [14]. Reasoning applies on sets of qualitative spatial relations which are jointly exhaustive and pairwise disjoint, i.e., between any two 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. For instance, if the binary relation R1 holds between entities A and B and the binary relation R2 holds between B an C, then the composition of R1 and R2 restricts the possible relationship between A and C. Compositions of relations are usually pre-computed and stored in a composition table [15].

PelletSpatial [1] extends Pellet OWL reasoner with qualitative spatial reasoning capabilities. It supports checking the consistency of spatial relations expressed using RCC-8 and computes new spatial inferences from asserted relations. The spatial relations are expressed

in RDF/OWL and and can be applied on arbitrary domain ontologies. PelletSpatial implements two RCC reasoner components: (a) A reasoner implementing the translation of RCC relations to OWL-DL class axioms while preserving their semantics and (b) a reasoner operating on the RCC composition table implementing a path-consistency algorithm [16].

1.4 Proposed work

The Region Connection Calculus (RCC), in the version of the theory we use, was introduced by Randell et al. in [4]. At the basis of the formalism, the authors define a primitive reflexive and symmetric dyadic relation C(x, y), meaning that region x connects with region y. On top of this relation, a number of other dyadic relations are defined, in particular eight jointly exhaustive pairwise disjoint relations known as the RCC-8 relations are defined. These relations are disconnected (DC), externally connected (EC), equals (EQ), partially overlap (PO), tangential proper part of (TPP), non-tangential proper part of (NTPP), has tangential proper part (TPPi), and has non-tangential proper part (NTPPi). Moreover, the authors define a composition table for these eight base relations enumerating the inferences that can be drawn by composing two relations.

The reasoner implemented in PelletSpatial as an alternative to the semantics preserving translation of RCC relations to OWL-DL class axioms uses a path-consistency algorithm to check the consistency of a set of defined RCC-8 relations. In [16], the authors provide a reference implementation for a pathconsistency algorithm applied to consistency checking for RCC.

1.5 Thesis Objective

Taking under consideration the proposed work of J.Renz and B.Nebel [16] we are going to implement a reasoner plugin for Protégé using this RCC-8 reasoner of PelletSpatial for spatial reasoning and querying in geodata.

1.6 Thesis Outline

Background knowledge and related research are discussed in Chapter 2. A description of Web Data Models and Languages and of qualitative spatial reasoning is presented. OWL Reasoners

and Semantic Web Technologies and Applications are introduced as well. In Chapter 3 we discuss the ProtegePelletSpatial Reasoning and more specifically we present the handling of the spatial ontologies and the architecture of our engine. In continuation we present the implementation of the plugin. Chapter 4 presents the results from our experiments while using one case study in comparison to the corresponding results from the RacerPro Reasoner and finally gets to a conclusion. Finally, conclusions and issues for further research are discussed in Chapter 5.

CHAPTER 2

Related Bibliography

The rapid growth of available information in the World Wide Web (WWW) has complicated the task of information retrieval and processing over the Web. Search engines such as Google, Bing and Ask improve the task of information retrieval as much as possible, however in most cases, the users still have to browse through the returned Web pages in order to fulfil tasks such as on-line shopping, travel planning or ticket reservations. Advanced search mechanisms may also be implemented based-upon learning or focused crawlers but still are incapable of fully automating tasks such as those referred to above. Automating these tasks require that machine understandable semantics become available along with data readable by humans in HTML pages that are currently available. The requirement of machine interpreted Semantics is the core idea of the Semantic Web vision [17]. In this chapter of the thesis we are going to talk about all the related bibliography to our thesis objective, which more specifically is: the Semantic Web Data Models and Languages, Qualitative Spatial Reasoning, OWL Reasoners and some of the Semantic Web Frameworks that are being used.

2.1 Semantic Web Data Models and Languages

Introducing machine readable semantics calls for a formal language for conceptualization of application domains, the related concepts, their properties and their relationships. Based-upon existing work on knowledge representation, logic and ontologies along with more recent approaches such as frames, Description Logics [18] form the basis for Semantic Web standards. Specifically the OWL representation language [19] and the SWRL rule language [20] are the description and rule languages respectively of the Semantic Web. Below we are going

to give some information about OWL and RDF which is a schema of SWRL.

2.1.1 Resource Description Framework

In this section we introduce the Resource Description Framework (RDF) together with its accompanying vocabulary RDFS. In addition, on the follow we will present the corresponding standardized query language, called SPARQL (SPARQL Protocol And RDF Query Language), for querying RDF and RDFS data. The Resource Description Framework (RDF) [21] is a framework for representing information about Web resources. It consists of W3C[1] recommendations that enable the encoding, exchange and reuse of structured data, providing means for publishing both human-readable and machine-processable vocabularies. Nowadays the current W3C recommendations for RDF are used in a variety of application areas [79], [80]. The Linked Data initiative[2], which aims at connecting data sources on the Web, has already become very popular and has exposed many datasets using RDF and RDFS. DBpedia[3], BBC music information [22], government datasets[4] are only a few examples of the constantly increasing Linked Data cloud[5]. The RDF data model offers the following basic concepts:

- ◇ Resources: In RDF, a resource is anything that we want to describe in the World Wide Web. A resource may be a Web page, a book, an author, a paper or a computer file. Every resource is uniquely identified by a Universal Resource Identifier (URI) [23].

- ◇ Properties: A property is a characteristic of a resource. For example, "hasTitle" may be used for describing the title of a book. Properties are also identified by URIs.

- ◇ Literals: Literals are constant values of any property. For example, "Artificial Intelligence Modern Approach" may be the value of property "hasTitle".

- ◇ Statements: Statements are the constructs offered by RDF for representing information about a domain. A statement has three parts: the resource the statement is about, the property of the resource the statement refers to, and the value of that property. The three parts of a statement are named, respectively, subject, predicate and object. The object of a statement can be another resource or a literal.

[1] http://www.w3.org/
[2] http://linkeddata.org/
[3] http://dbpedia.org
[4] http://www.data.gov/, http://data.gov.uk/
[5] 5http://www4.wiwiss.fu-berlin.de/lodcloud/state/

Two possible representations of RDF data are labeled graphs or triple sets. In the first one, a resource of a literal is depicted as a node and a property as an arc. In the triple sets representation, all statements are represented as "triples" in the form subject predicate object (resource, property, value).

2.1.2 SPARQL Query Language

In brief, the SPARQL query language [29] can be used to express queries across diverse data sources, whether the data is stored natively as RDF or viewed as RDF via middleware. SPARQL contains capabilities for querying required and optional graph patterns along with their conjunctions and disjunctions. SPARQL also supports extensible value testing and constraining queries by source RDF graph. The results of SPARQL queries can be results sets or RDF graphs.

Verbosely, during the past years, many query languages have been proposed for the RDF data model. Some of them include RQL [24], RDQL [25], SeRQL [26], and TRIPLE [27]. On 15 January 2008, SPARQL [28] became the official W3C recommendation language for querying RDF data. SPARQL [28] is a query language for RDF graphs and has the ability to extract information about both the data and the schema. SPARQL is based on the concept of matching graph patterns. The simplest graph patterns are triple patterns, which are like an RDF triple but with the possibility of a variable in any of the subject, predicate or object positions. A query that contains a conjunction of triple patterns is called basic graph pattern. A basic graph pattern matches a subgraph of the RDF data when RDF terms from the subgraph may be substituted with the variables of the graph pattern. The syntax of SPARQL follows an SQLlike select-from-where paradigm. The select clause specifies the variables that should appear in the query results. Each variable in SPARQL is prefixed with character "?". The from clause specifies the RDF(S) graph that should be used for answering the query. If not used, the query runs over the whole RDF(S) triples stored in the system. The graph patterns of the query are set in the where clause.

At this point, we should note that the current recommendation of SPARQL has a lot of shortcomings. For instance, it does not support aggregates, updates, and more importantly, it does not deal with the RDFS entailment. Therefore, RDFS entailment depends on the implementation of each system. The next version of SPARQL, i.e., SPARQL 1.1, will introduce several of these issues. One of these is RDFS regimes which define the evaluation of basic graph patterns using semantic entailment relations instead of explicitly given RDF graphs.

This effort is still a W3C working group and will be soon a W3C recommendation. Apart from the W3C working group, several extensions for SRARQL have been proposed in the literature that capture the RDFS entailment semantics. One of them is nSPARQL, a navigational language for RDF which provides the expressivity of RDFS semantics as well [29], [30].

2.1.3 The Web Ontology Language

The Web Ontology Language OWL is a semantic markup language for publishing and sharing ontologies on the World Wide Web. It is developed as a vocabulary extension of RDF (the Resource Description Framework) and is derived from the DAML+OIL Web Ontology Language. Descriptions on OWL classes are discussed in details in [31]. A property restriction is an unnamed class containing all individuals that satisfy the restriction. Properties are binary relationships between two objects. In general they are the relationships between two classes which apply to the individual of those classes. They are known as roles in description logic and are represented through links in the graphical representation. OWL provides two main categories of properties:

- ◇ Object properties: relationships between concepts and consequently instances of the concepts and
- ◇ Data properties: relation of an instance to the data value.

OWL is intended to provide a language that can be used to describe the classes and relations between them that are inherent in Web documents and applications. OWL provides three increasingly expressive sublanguages:

1. OWL Lite supports classification hierarchy and simple constraints.
2. OWL DL supports maximum expressiveness without loosing computational completeness and decidability.
3. OWL Full provides maximum expressiveness and the syntactic freedom of RDF, but without computational guarantees.

Each of these sublanguages is an extension of it's simpler predecessor. OWL is part of W3C's Semantic Web technology stack. In October 2007 a new W3C working group was started to extend OWL with several new features. This new version was called OWL 2 [32], [33].

More verbosely, the evolution of the OWL specification was based on the observation that additional constructs can be added in OWL-DL without compromising decidability, while increasing expressiveness. Extending OWL-DL with the additional constructs leaded to the adoption of OWL 2 [34] as the current Semantic Web standard [35]. OWL 2 is based on SROIQ(D) description logic [37] (i.e., SROIQ(D) offers all constructs of SHOIN(D) with the addition of qualified number restrictions (Q) and complex role inclusion axioms (R)). The computational properties of SHROIQ(D)-OWL 2.0 are analysed in [36], along with a description of the corresponding tableaux algorithm for reasoning over OWL 2. In addition to constructs offered by OWL-DL, OWL 2 offers support for qualified number restrictions (Q) in addition to the unqualified ones and complex role inclusion axioms (R)along with disjoint, symmetric, asymmetric, reflexive, irreflexive properties and property negation in addition to subproperties offered by OWL-DL. SHROIQ(D) is decidable thus offering additional expressiveness, while retaining the computational properties of OWL-DL. SOWL aims at expressing dynamic concepts using current Semantic Web standards, such as OWL 2 and SWRL.

2.1.4 Spatio-temporal information in OWL

SOWL [11], is an ontology for representing and reasoning over spatio-temporal information in OWL. Building-upon well established standards of the semantic web (OWL 2.0, SWRL) SOWL enables representation of static as well as of dynamic information. Both RCC-8 topological and cone-shaped directional(CSD-9) relational calculi are integrated in SOWL. Representing both qualitative temporal and spatial information (i.e., information whose temporal or spatial extents are unknown such as "left-of" for spatial and "before" for temporal relations) in addition to quantitative information (i.e., where temporal and spatial information is defined precisely) is a distinctive feature of SOWL. The SOWL reasoner is capable of inferring new relations and checking their consistency, while retaining soundness, completeness, and tractability over the supported sets of relations. The SOWL spatial representation implements reasoning rules for RCC-8 relations and cone-shaped direction relations using SWRL and OWL 2.0 property axioms. Specifically, the nine direction relations have been declared as transitive OWL relations (i.e., a relation such as South is transitive meaning that if the relation holds between locations A and B, and between locations B and C, it also holds between locations A and C). Their inverse relations (e.g., North is the inverse of South) are defined as well. Furthermore, the identity relation (O) is symmetric. All basic relations are pairwise disjoint. Path consistency is implemented by introducing rules defining compositions and

intersections of supported relations until a fixed point is reached or until an inconsistency is detected. Reasoners that support DL-safe rules such as Pellet can be used for inference and consistency checking over spatio-temporal relations.

But, in contrast to PelletSpatial, reasoning is part of the ontology (rather than a separate system), so that maintenance of the ontology requires that changes are applied only to the ontology and not to the system. Moreover, PelletSpatial is using the Path-Consistency algorithm instead of SOWL reasoning rules. Similarly to PelletSpatial, extracting spatial relations from the raw spatial data depends on the application and is not part of the reasoning mechanism. For more details on the implementation of the spatial reasoner the interested reader is referred to [11].

2.2 Qualitative Spatial Reasoning

In this chapter section, reasoning over both, topological and directional relations is discussed. Choosing either representation is a design decision that depends mainly on the application. The qualitative approach to spatial as well as to temporal information is popular in Artificial Intelligence and related research fields [37], [38]. This is mainly because precise numerical information is often unavailable or not necessary in many real world applications. Typically, the qualitative approach represents spatial information by introducing a (binary) relation model on the universe of spatial entities, which contains a finite set of binary relations defined on the universe. Finding a proper relation model, or a qualitative calculus, is the key to the success of the qualitative approach to spatial reasoning. In the past twenty years, dozens of spatial relation models have been developed. Since relations in the same model are ideally homogenous, most spatial calculi focus on one single aspect of space, e.g. topology, direction, distance, or position. When representing spatial direction it is convenient to approximate spatial entities by points. But this is inappropriate as far as spatial topological information is concerned: topology concerns sets of points, i.e. regions. In the next subsections, we are going to give the definitions of the RCC topological region connection calculus and the directional CSD Calculus and show their contents.

2.2.1 Region Connection Calculus (RCC-8)

In brief, topological relations represent the relative position of regions in the plane. The most widespread formalism for representing such relations is the so called Region Connection

Calculus(RCC) formalism [4]. The most commonly used form of this calculus is referred to as RCC-8 calculus and specifies the 8 mutually exclusive relations between region pairs (DC, EC, EQ, NTPP, NTPPi, TTP, TPPi, PO) which are shown in Fig. 2.1

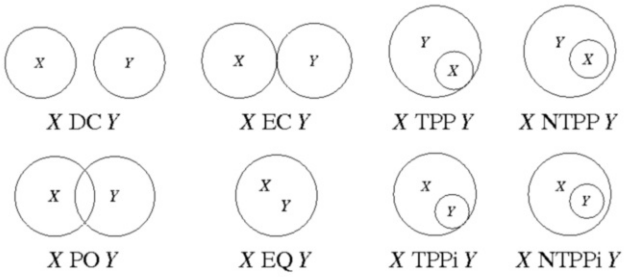

Fig. 2.1: *The set of RCC-8 Topologic Relations.*[78]

More specifically, the Region Connection Calculus (RCC) is an axiomatization of certain spatial concepts and relations in first order logic [4], [39]. The basic theory assumes just one primitive dyadic relation: C(x, y) read as "x connects with y". Individuals (x, y) can be interpreted as denoting spatial regions. The relation C(x, y) is reflexive and symmetric.

Using the primitive relation C(x, y) a number of intuitively significant relations can be defined. The most common of these are illustrated in Fig. 2.1 and their definitions together with those of additional relations are given in table 2.1. The asymmetrical relations P, PP, TPP and NTPP have inverses which we write, in accordance with [39], as Ri, where R $\in$ P, PP, TPP, NTPP. These relations are defined by definitions of the form Ri(x, y) $\equiv_{def}$ R(y, x).

Of the defined relations, DC, EC, PO, EQ, TPP, NTPP, TPPi and NTPPi have been proven to form a jointly exhaustive and pairwise disjoint set, which is known as RCC-8. Similar sets of one, two, three and five relations are known as RCC-1, RCC-2, RCC-3 and RCC-5, respectively: RCC-1 = SR, RCC-2 = O, DR, RCC-3 = ONE, EQ, DR, RCC-5 = PP, PPi, PO, EQ, DR. RCC also incorporates a constant denoting the universal region, a sum function and partial functions giving the product of any two overlapping regions and the complement of every region except the universe [39].

According to [4], regions support either spatial or temporal interpretation. In case of spatial interpretation, there is a variety of models among which to choose. The authors provide some examples such as interpreting the relation C ("connects with") in terms of two regions whose closures share a common point or stating that two regions connect when the distance between them is zero.

In order to check consistency of a knowledge base holding spatial relations, so-called

Table 2.1: *Basic RCC Relations.*[51]

$SR(x, y)$	$\equiv_{def} \top(x, y)$	(Spatially Related)
$C(x, y)$	(primitive relation)	(Connects with)
$DC(x, y)$	$\equiv_{def} \neg C(x, y)$	(DisConnected from)
$P(x, y)$	$\equiv_{def} \forall z[C(z, x) \rightarrow C(z, y)]$	(Part of)
$O(x, y)$	$\equiv_{def} \exists z[P(z, x) \wedge P(z, y)]$	(Overlaps)
$DR(x, y)$	$\equiv_{def} \neg O(x, y)$	(DiscRete from)
$EC(x, y)$	$\equiv_{def} C(x, y) \wedge \neg O(x, y)$	(Externally Connected to)
$EQ(x, y)$	$\equiv_{def} P(x, y) \wedge P(y, x)$	(EQual to)
$ONE(x, y)$	$\equiv_{def} O(x, y) \wedge \neg EQ(x, y)$	(Overlaps Not Equal)
$PP(x, y)$	$\equiv_{def} P(x, y) \wedge \neg P(y, x)$	(Proper Part of)
$PO(x, y)$	$\equiv_{def} O(x, y) \wedge \neg P(x, y) \wedge \neg P(y, x)$	(Partially Overlaps)
$TPP(x, y)$	$\equiv_{def} PP(x, y) \wedge \exists z[EC(z, x) \wedge EC(z, y)]$	(Tangential Proper Part of)
$NTTP(x, y)$	$\equiv_{def} PP(x, y) \wedge \neg\exists z[EC(z, x) \wedge EC(z, y)]$	(Non-Tangential Proper Part of)

composition tables are used (cf. the composition table for RCC-8 in table 2.2). The entries in these tables share a uniform inference pattern which can be formalized as composition axioms of the general form ?x, y, z. S(x, y)?T(y, z) > R1(x, z) ? ... ? Rn(x, z) where S, T, and Ri are variables for relation symbols. A similar approach which is based on the description of topological relations between two spatial regions was introduced as the 9-intersection model in [40]. In this model, eight out of nine relations can be interpreted in the same way as we interpret the RCC-8 relations, namely as spatial relations between polygons in the integral plane [41]. Only the ninth relation is specific for the model. Given this extensive agreement on the interpretation of the relations between the two approaches we believe that the latter, which is axiomatized in first order logic, is easier to combine with description logics than the first, which is based on a topological framework. The reason therefore is that description logics themselves can be seen as fragments of first order logic [42].

2.2.2 Cone-shaped Directional Calculus

Cone-shaped Directional Calculus is the the representation of directional relations which are defined based on cone-shaped areas [43], [44]. Moreover, the goal of a qualitative representation of the direction between points in two-dimensional space is to specify a limited number of relations such that each relation covers a part of the 360 degrees range and all relations

Table 2.2: *RCC-8 composition table (T(x, z) $\equiv_{def}$ DC(x, z), EC(x, z), PO(x, z), TPP(x, z), NTPP(x, z), TPPi(x, z), NTPPi(x, z), EQ(x, z))*

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

taken together cover the 360 degrees range completely. If in addition the relations do not overlap, they form a jointly exhaustive and pairwise disjoint (JEPD) set of relations, called basic relations. In CSD model an angular direction is assigned the nearest named direction which results in cone-shaped areas for which a symbolic direction is applicable. This model has the property that "the area of acceptance for any given direction increases with distance" [45] and is sometimes called 'triangular'. Cone-shaped Directional (Fig. 2.2) defines the set of 9 basic relations that are possible between two points (e.g., region's centroid), with 8 turns of 45 degrees being the identity function:

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

One can also form a subset without O. That would be {N, NE, E, SE, S, SW, W, NW}

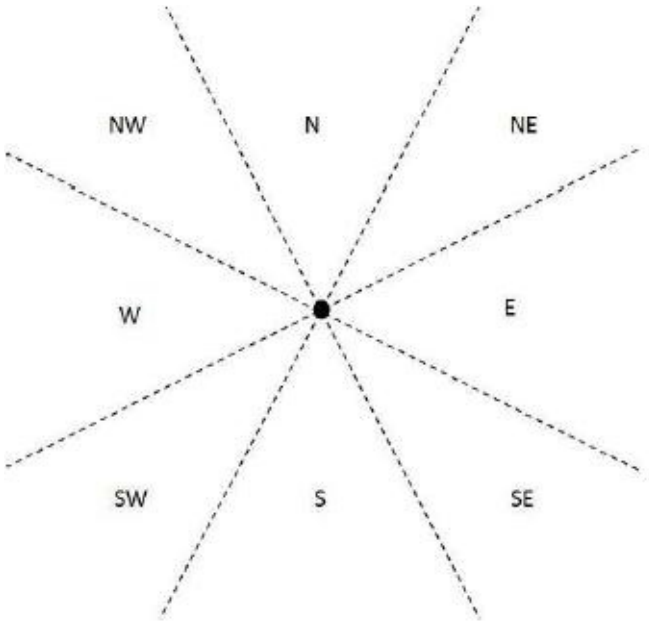

Fig. 2.2: *The set of Cone-Shaped Directional Relations.*[78]

2.2.3 Complexity Analysis

The complexity of a decision problem is usually measured in term 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, i.e., the size of the problem instance, and can be expressed as a function f(n). The asymptotic behavior of f is specified in terms of the O-notation which gives an upper bound on the running time. In areas like database systems where instances are very large size, a running time of O(n^3) corresponds to a solution which is very slow in practice. In these cases efficient algorithms have linear running time. For most of the tractable subsets of qualitative spatial calculi, path-consistency or even simpler methods are sufficient for deciding consistency. So except for very large instances or for calculi over a large set of relations, there are usually no efficiency problems. As shown in [16], by restricting the supported relations set to a tractable subset of RCC-8 (or corresponding directional) spatial relations, path consistency has O(n^5) time complexity (with n being the number of individuals) and is sound and complete. Note that, extending the model for the full set of relations would result into an intractable reasoning procedure. In PelletSpatial, path consistency algorithm (and hence reasoning RCC-8 relation sets) has O(n^3) complexity in the worst case. Later, in our experiments this complexity will be shown from the case study graphs.

2.3 OWL Reasoners

A semantic reasoner, reasoning engine, rules engine, or simply a reasoner, is a piece of software able to infer logical consequences from a set of asserted facts or axioms. The notion of a semantic reasoner generalizes that of an inference engine, by providing a richer set of mechanisms to work with. The inference rules are commonly specified by means of an ontology language, and often a description language [6].

2.3.1 Pellet

Pellet[7] [4], is a complete OWL reasoner with very good performance and a number of unique features. It is written in Java and is open source under a very liberal license. It has been adopted in projects and application from pure research to industrial settings. Pellet supports reasoning with the full expressivity of OWL-DL ($\mathcal{SHOIN}(\mathcal{D})$ in Description Logic jargon) and has been extended to support the more recent OWL 2 specification ($\mathcal{SROIQ}(\mathcal{D})$). It provides all the standard inference services that are traditionally provided by DL reasoners:

- ◇ Consistency checking: Ensures that an ontology does not contain any contradictory facts. The OWL 2 Direct Semantics provides the formal definition of ontology consistency used by Pellet.
- ◇ Concept satisfiability: Determines whether its possible for a class to have any instances. If a class is unsatisfiable, then defining an instance of that class makes the entire ontology inconsistent.
- ◇ Classification: Computes the subclass relations between every named class to create the complete class hierarchy. The class hierarchy can be used to answer queries such as getting all or only the direct subclasses of a class.
- ◇ Realization: Finds the most specific classes that an individual belongs to; i.e., realization computes the direct types for each of the individuals. Realization can only be performed after classification since direct types are defined with respect to a class hierarchy. Using the classification hierarchy, it is also possible to get all the types for each individual.

[6] http://en.wikipedia.org/wiki/Semantic_reasoner
[7] http://clarkparsia.com/pellet/

Pellet relies on an implementation of a direct tableau algorithm [46], [47] for a DL-safe rules extension to OWL-DL. This implementation allows one to load and reason with DL-safe rules encoded in SWRL [48] and includes support for some SWRL built-ins.

2.3.2 PelletSpatial

PelletSpatial [1] extends Pellet OWL reasoner with qualitative spatial reasoning capabilities. It supports checking the consistency of spatial relations expressed using RCC-8 and computes new spatial inferences from asserted relations. The spatial relations are expressed in RDF/OWL and can be combined with arbitrary domain ontologies. PelletSpatial can answer SPARQL queries that mix spatial relations with arbitrary RDF/OWL relations. PelletSpatial implements two RCC-8 reasoners: (a) A reasoner based on the semantics preserving translation of RCC relations to OWL-DL class axioms and (b) a reasoner based on the RCC-8 composition table that implements a variant of the path-consistency algorithm of 2.1. Results show that, without further optimizations, the first reasoner based on the translation of RCC-8 relations to OWL-DL class axioms does not scale-up well with the size of the dataset, and lacks practicability even for small datasets. In addition to translation 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 axiom 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 source of complexity (AND-branching) in DL reasoning [49]. The later spatial reasoner based on a path-consistency algorithm 2.1 and the RCC-8 composition table (table 2.2) exhibited most promising performance and so, it is more promising with regards to performance[8], and occasionally it is the one used in some implementations for supporting reasoning with CSD-9 calculus(Fig. 2.2).

[8] http://clarkparsia.com/pellet/spatial

Algorithm 2.1 *Path Consistency Algorithm*[1]

```
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_ac ← 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_ca~)
46: end procedure
```

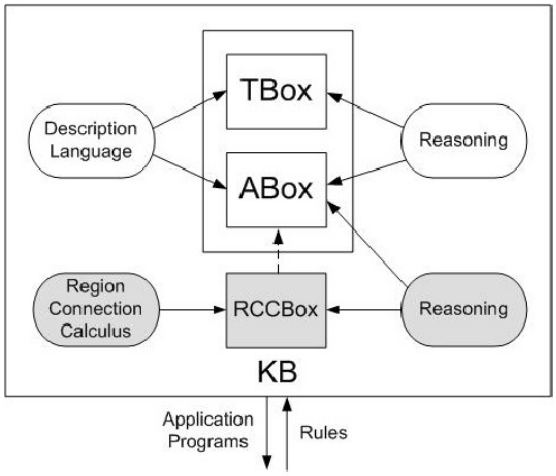

Fig. 2.3: *Architecture of the hybrid knowledge representation system of PelletSpatial*[51]

Furthermore, while taking a deeper glance at the PelletSpatial engine, (Fig. 2.3) illustrates the PelletSpatial hybrid system architecture in its simplest form. The grey shaded components are extensions to the original architecture. The shortcut KB denotes the knowledge

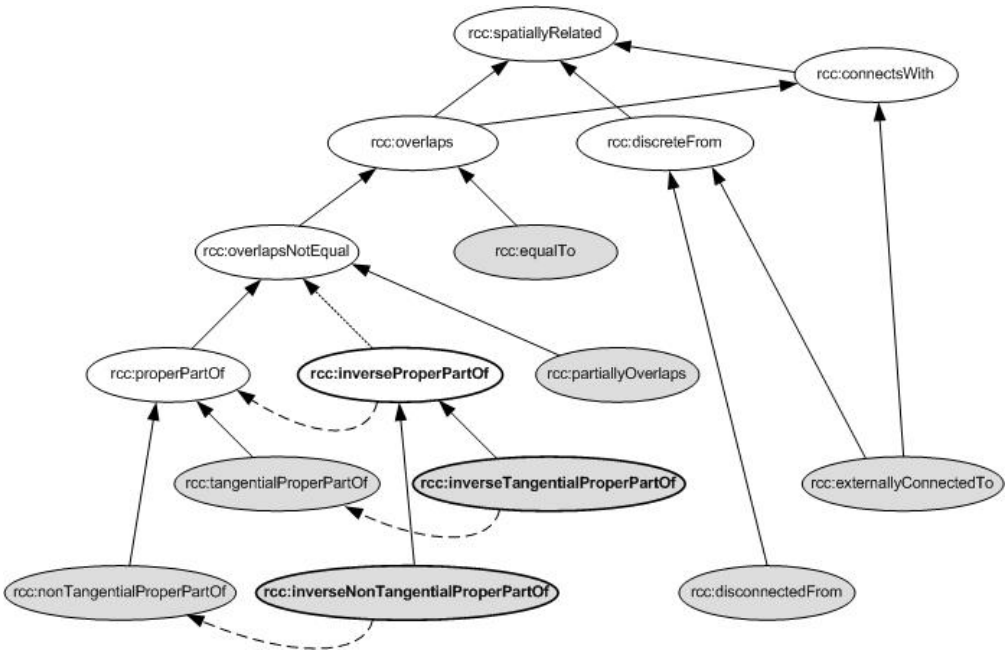

Fig. 2.4: *RCC-8 family tree hierarchy of object properties in OWL. Regular arrows represent the rdfs:subPropertyOf property, dotted arrows the owl:inverseOf property. Nodes in bold face represent relations that can be defined in terms of OWL DL. Regular nodes represent relations that cannot be defined in terms of OWL DL. Grey shaded nodes represent the RCC-8 relations*[51]

base. The TBox holds the thematic terminology which is introduced by an ontology specifying the users' conceptualization of the domain (not shown). It also contains the relation names of the different RCC species structured in a hierarchy of object properties. These names are used to assert selected relations between connecting individual regions in the ABox. The label RCCBox stands for Region Connection Calculus Box, a term which is inspired by the role box in [50]. The RCCBox contains the composition tables for RCC-1, RCC-2, RCC-3, RCC-5 and RCC-8. The RCC reasoner uses the role assertions in the ABox in order to calculate those pairs of regions which are not connected, and it uses the composition tables in order to check spatial consistency of the ABox. The dotted arrow pointing from the RCCBox back to the ABox indicates, that the calculated relations can be asserted in the ABox in order to speed up the processing of similar queries in the future.

Figure moreover, in Fig. 2.4 is shown an acyclic directed labeled graph with the names of the different RCC species which are structured in a hierarchy of object properties in OWL. By traveling down the hierarchy from the top to the bottom the relations are successively refined to yield the different RCC species.

Table 2.3: *Terminology and world description introduced to the knowledge base*[51]

DL Syntax	Semantics
$\top$	$\Delta^{\mathcal{I}}$
Region	$\text{Region}^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}}$
spatiallyRelated	$\text{spatiallyRelated}^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}}$
$\exists$spatiallyRelated.$\top \sqsubseteq$ Region	$\{a \in \Delta^{\mathcal{I}} \mid \exists b.\ (a, b) \in \text{spatiallyRelated}^{\mathcal{I}}\} \subseteq \text{Region}^{\mathcal{I}}$
$\top \sqsubseteq \forall$spatiallyRelated.Region	$\Delta^{\mathcal{I}} \subseteq \{a \in \Delta^{\mathcal{I}} \mid \forall b.\ (a, b) \in \text{spatiallyRelated}^{\mathcal{I}} \rightarrow b \in \text{Region}^{\mathcal{I}}\}$
overlaps $\sqsubseteq$ spatiallyRelated	$\text{overlaps}^{\mathcal{I}} \subseteq \text{spatiallyRelated}^{\mathcal{I}}$
overlapsNotEqual $\sqsubseteq$ overlaps	$\text{overlapsNotEqual}^{\mathcal{I}} \subseteq \text{overlaps}^{\mathcal{I}}$
partiallyOverlaps $\sqsubseteq$ overlapsNotEqual	$\text{partiallyOverlaps}^{\mathcal{I}} \subseteq \text{overlapsNotEqual}^{\mathcal{I}}$
Region(Albiskette-Reppischtal)	$\text{Albiskette-Reppischtal}^{\mathcal{I}} \in \text{Region}^{\mathcal{I}}$
Region(Birmensdorf)	$\text{Birmensdorf}^{\mathcal{I}} \in \text{Region}^{\mathcal{I}}$
partiallyOverlaps(Birmensdorf, Albiskette-Reppischtal)	$(\text{Birmensdorf}^{\mathcal{I}}, \text{Albiskette-Reppischtal}^{\mathcal{I}}) \in \text{partiallyOverlaps}^{\mathcal{I}}$

As Fig. 2.4 shows only the relations rcc:inverseProperPartOf, rcc:inverseTangentialProperPartOf and rcc:inverseNonTangentialProperPartOf can be defined in terms of OWL DL. For the majority of relations the TBox contains necessary but not sufficient axioms of the form $\equiv_{def} R_{RCC-i^+} \sqsubseteq R_{RCC-i}$ (i.e., rdfs:subPropertyOf) with i^+, i $\in$ {1, 2, 3, 5, 8} and i^+ > i, where R_{RCC-i^+} and R_{RCC-i} denote arbitrary relations of the species R_{RCC-i^+} and R_{RCC-i} (cf. table 2.3 for examples). Note that because of the transitive property of rdfs:subPropertyOf it holds that $((R_{(RCC-i^+)^+} \sqsubseteq R_{RCC-i^+}) \wedge (R_{RCC-i^+} \sqsubseteq R_{RCC-i})) \rightarrow R_{(RCC-i^+)^+})+ \sqsubseteq R_{RCC-i}$ [51].

2.3.3 RacerPro

RACER engine stands for Renamed ABox and Concept Expression Reasoner. The origins of RacerPro are within the area of description logics. Since description logics provide the foundation of international approaches to standardize ontology languages in the context of the so-called semantic web, RacerPro can also be used as a system for managing semantic web ontologies based on OWL (e.g., it can be used as a reasoning engine for ontology editors such as Protégé). However, RacerPro can also be seen as a semantic web information repository with optimized retrieval engine because it can handle large sets of data descriptions (e.g., defined using RDF). Last but not least, the system can also be used for modal logics such as Km.

RacerPro is a knowledge representation system that implements a highly optimized tableau calculus for a very expressive description logic. It offers reasoning services for multiple T-boxes and for multiple A-boxes as well. The system implements the description logic

$\mathcal{ALCQHIR}+$ also known as $\mathcal{SHIQ}$. This is the basic logic $\mathcal{ALC}$ augmented with qualifying number restrictions, role hierarchies, inverse roles, and transitive roles.

In addition to these basic features, RacerPro also provides facilities for algebraic reasoning including concrete domains for dealing with:

- ◇ min/max restrictions over the integers,
- ◇ linear polynomial (in-)equations over the reals or cardinals with order relations,
- ◇ equalities and inequalities of strings.

For these domains, no feature chains can be supported due to decidability issues. RacerPro supports the specification of general terminological axioms. A T-box may contain general concept inclusions (GCIs), which state the subsumption relation between two concept terms. Multiple definitions or even cyclic definitions of concepts can be handled by RacerPro [52] (Fig. 2.5).

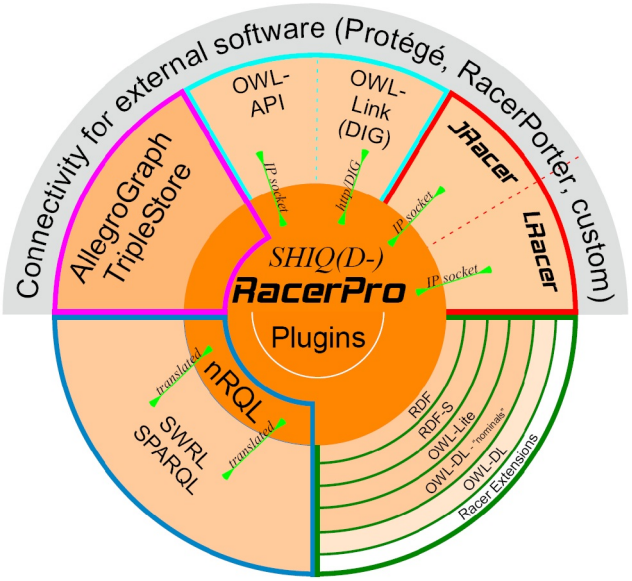

Fig. 2.5: *RacerPro Architecture* [9]

2.3.3.1 RCC-8 reasoning support

RacerPro engine, although it supports spatial reasoning using the RCC calculus, in contrary to the PelletSpatial that is a hybrid reasoner and contains two reasoners has only one reasoner

and depending on the data it inputs for reasoning or querying it treats them as spatial or non-spatial. RacerPro engine's complexity for spatial data depends on how spatial data are recognised. Complexity is not really good at closed domain reasoning, since the open domain assumption is made in DLs. In contrast, since the geometry of the map is completely specified, there is neither unknown nor underspecified spatial information,and this motivates the classification of the data as "spatial data", in contrast to "spatial information" [53].

2.4 Semantic Web Frameworks

The Web is growing at an astounding pace surpassing the 8 billion page mark. However, most pages are still designed for human consumption and cannot be processed by machines. One solution to this problem is evolvent of Semantic Web Technologies and Applications, whose objective is to demonstrate how RDF and Semantic Web technologies can be applied to the W3C Process to increase efficiency and reliability in Web processing. Some of such technologies are presented in this thesis[10] [54], [55], [56], [57].

2.4.1 Protégé Framework

Protégé[11], [58] is a free, open-source platform that provides a suite of tools for building domain models and knowledge-based applications with ontologies. It supports the creation, visualization and manipulation of ontologies in various representation formats. Protégé can be customized to provide domain-friendly support for creating knowledge models and entering data. Further, Protégé can be extended by way of a plug-in architecture and a Java-based Application Programming Interface (API)(Fig. 2.6) for building knowledge-based tools and applications. The Protégé platform supports two main ways of modeling ontologies, the Protégé-Frames editor and the Protégé OWL editor. The Protégé-OWL editor enables users to build ontologies for the SemanticWeb, in particular in the W3C's Web Ontology Language (OWL) [59]. The Protégé-OWL editor, also, offers the necessary tools in order to use OWL and RDF ontologies, edit and visualize classes, properties and SWRL rules, define logical class characteristics as OWL expressions. The Protégé platform provides the flexibility of integrating additional modules. Moreover, Protégé supports three types of plug-ins: storage plug-ins, slot widgets, and tabs.

[10] http://infomesh.net/2001/swintro/
[11] http://protege.stanford.edu/

Storage plug-in is a module that saves and loads models in a certain file or database format (CLIPS, XML, XML Schema, RDF, OIL, DAML+OIL, DARPA Agent Markup Language+OIL, UML, XMI metamodels).

Slot widgets are graphical components such placed in Protégé's instance forms to view and edit a slot value. Protégé's plug-in library provides additional slot widgets for specific data types such as calendar and date widgets, and components that display images, sounds, and videos.

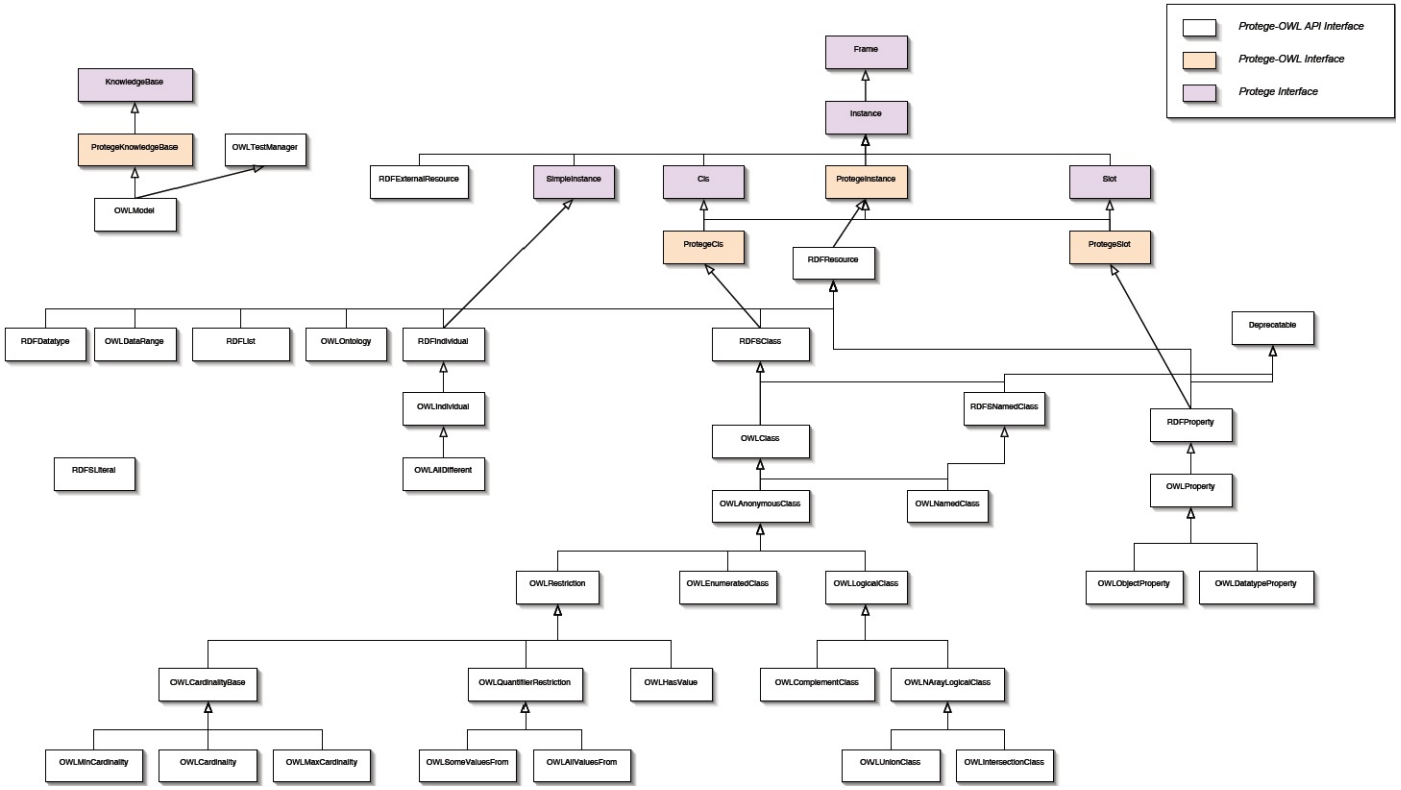

Fig. 2.6: *Protégé Interface*[12]

Tabs plug-ins are graphical plug-ins, displayed as a tab in Protégé's main window. These tabs include:

◇ Visualization tabs

- ○ Jambalaya provides a hierarchical ontology browser that allows for interactive editing of existing data;
- ○ TGVizTab provides an interactive graphs-based visualization method for classes;

- OntoViz provides a configurable graphical display of models in graphs similar to UML diagrams.

◇ Project and file management tabs

- BeanGenerator enforces the interaction between Java and Protege, by generating JavaBeans classes from a Protege class model;
- DataGenie enforces the interaction between databases and Protege, allowing Protégé access to databases using the Java Database Connectivity interface;
- Prompt allows managing different domain models in Protégé (merge models, to extract a part of a model, or to identify differences between a model's two versions).

◇ Reasoning tabs

- Query tab for querying the knowledge base;
- PAL constraint and query tabs provide the mechanisms for editing and evaluating expressions in the PAL;

Protégé uses a declarative approach to model ontologies, declaring explicitly the class hierarchies and individual membership to classes. The main elements of ontology are: classes (also called concepts), concept features and attributes (called slots, roles or properties), and restrictions on slots (facets, role restrictions).

The ontology concept has common elements with the object oriented design approach. Protégé classes are similar to Java classes, can be arranged in an inheritance hierarchy, but they do not provide method definition as in Java. Classes can be defined as abstract or concrete, only the latter can be instantiated. Protégé also supports multiple inheritance. Attributes (slots), have a name and a value type: primitive value types: (boolean, integer, float, or string), symbols (represent enumerations of strings), or references to the instances and classes. Slots are also uses to build relationships between instances. Although slots are similar with the class attributes in object oriented languages, there are some specific features of Protégé slots:

◇ a slot can attach to multiple classes;

◇ slots can have values constraints;

- ◇ slots are global objects (they can even exist without being assigned to a class); for each slot's assigned class, the developer can keep the general properties or override the slots' properties;

- ◇ Protégé Axiom Language (PAL), is a very powerful Protégé built-in language, which can be applied when dealing with more complex constraints;

In our case the ProtegePelletSpatial plugin that we developed belongs to the reasoning tabs group and especially needs at least the Version 4.1 of Protégé OWL, which provides full support of OWL 2.0 [32].

2.4.2 Jena Frameworkowlapiowlapi

Jena [60], [66] is a Java framework for building Semantic Web applications. It provides extensive Java libraries for helping developers develop code that handles RDF, RDFS, RDFa, OWL and SPARQL in line with published W3C recommendations. Jena includes a rule-based inference engine to perform reasoning based on OWL and RDFS ontologies, and a variety of storage strategies to store RDF triples in memory or on disk.

At its core (Fig. 2.7), Jena stores information as RDF triples in directed graphs, and allows adding, removing, manipulating, storing and publishing that information. Jena is considered as a number of major subsystems with clearly defined interfaces between them. RDF triples and graphs, and their various components, are accessed through Jena's RDF API. he RDF API has basic facilities for adding and removing triples to graphs and finding triples that match particular patterns.

While the programming interface to Model is quite rich, internally, the RDF graph is stored in a much simpler abstraction named Graph. This allows Jena to use a variety of different storage strategies equivalently, as long as they conform to the Graph interface. Out-of-the box, Jena can store a graph as an in-memory store, in an SQL database, or as a persistent store using a custom disk-based tuple index. The graph interface is also a convenient extension point for connecting other stores to Jena, such as LDAP, by writing an adapter that allows the calls from the Graph API to work on that store.

A key feature of semantic web applications is that the semantic rules of RDF, RDFS and OWL can be used to infer information that is not explicitly stated in the graph. For example, if class C is a sub-class of class B, and B a sub-class of A, then by implication C is a sub-class of A. Jena's inference API provides the means to make these entailed triples appear

in the store just as if they had been added explicitly. The inference API provides a number of rule engines to perform this job, either using the built-in rulesets for OWL and RDFS, or using application custom rules. Alternatively, the inference API can be connected up to an external reasoner, such as description logic (DL) engine, to perform the same job with different, specialised, reasoning algorithms.

The collection of standards that define semantic web technologies includes SPARQL [28] - the query language for RDF. Jena conforms to all of the published standards, and tracks the revisions and updates in the under-development areas of the standard. Handling SPARQL, both for query and update, is the responsibility of the SPARQL API.

While the above capabilities are typically accessed by applications directly through the Java API, publishing data over the Internet is a common requirement in modern applications. Fuseki is a data publishing server, which can present, and update, RDF models over the web using SPARQL and HTTP.

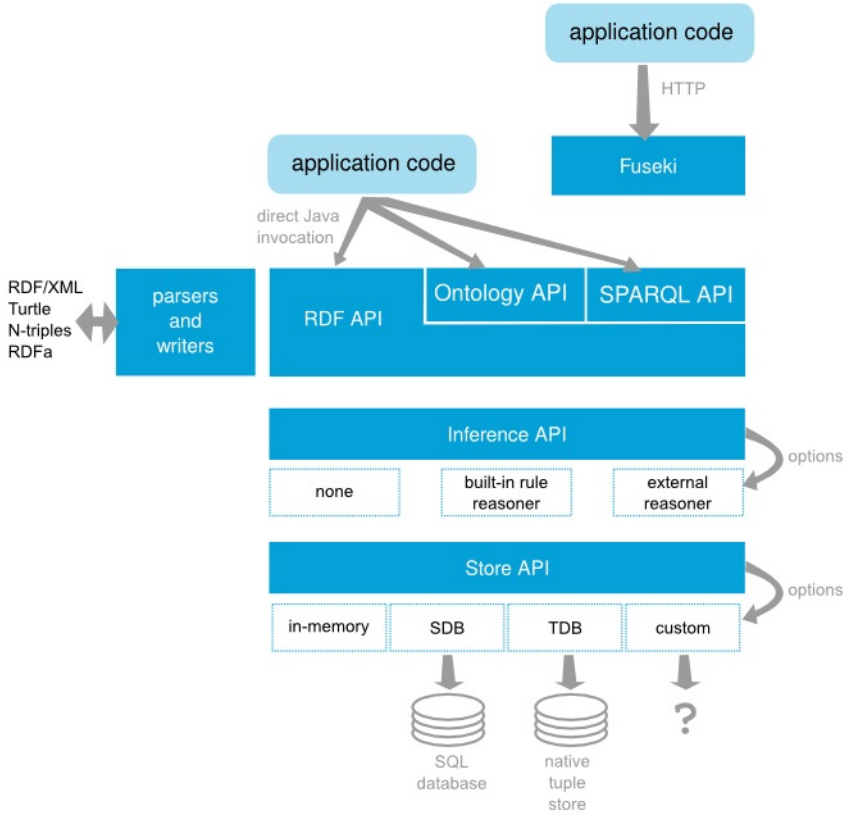

Fig. 2.7: *Jena Architecture*[60]

2.4.3 OWLAPI Framework

The original inspiration for the OWL API [61] came from observations of the impact made by the provision of the XML Document Object Model (DOM) [62]. The DOM, along with freely available implementations (such as the Java implementations in Sun"s JDK [63]) allowed a large number of developers to use and manipulate XML in applications, which in turn facilitated the widespread adoption of XML. Our belief was that a similar effort would prove of great benefit to the adoption of OWL.

The provision of an OWL API can allow developers to work at an appropriate level of abstractions, isolating them from potential issues related to, for example, serialisation and parsing of data structures. This was a particular consideration with OWL, given OWL"s close relationship with RDF [64] and its triple-based representation. In order to provide this "high level view", the design of the OWL API is directly based on the OWL 2 Structural Specification [31]. An ontology is simply viewed as a set of axioms and annotations as depicted in Fig. 2.8. The names and hierarchies of interfaces for entities, class expressions and axioms in the OWL API correspond closely to the structural specification, relating the high level OWL 2 specification directly to the design of the OWL API. The OWL API supports loading and saving ontologies is a variety of syntaxes. However, none of the model interfaces in the OWL API reflect, or are biased to any particular concrete syntax or model. For example, unlike other APIs such as Jena [66], or the Protege 4.X API, the representation of class expressions and axioms is not at the level of RDF triples.

OWLAPI basic design principles have endured. These are outlined below, but can be summarised as:

- ◇ Interfaces providing read-only access to the model structure.
- ◇ Change/manipulation through explicit change operations.
- ◇ Independence from concrete serialisations (in particular triple based representations).
- ◇ A clear separation between components providing particular functionalities, such as representation, manipulation, parsing, rendering.
- ◇ Separation of assertion and inference.

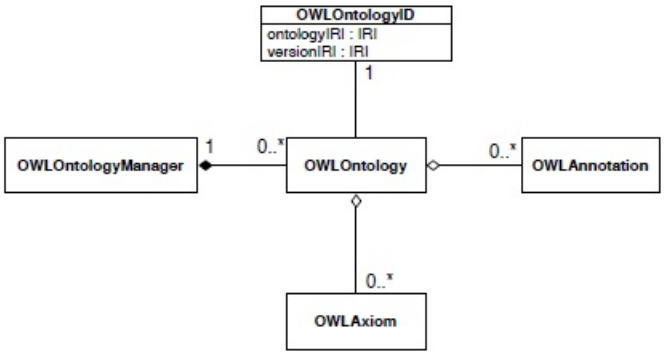

Fig. 2.8: *UML diagram showing the management of ontologies in the OWL API. The OWLOntology interface provides accessors for efficiently obtaining the axioms contained within an ontology. The method of storing axioms is provided by different implementations of the OWLOntology interface. The design of the API makes it possible to mix and match implementations, for example, an in-memory ontology could be used together with one that is stored in a database and one that is stored in some kind of triple store. The OWL API reference implementation provides an efficient in-memory storage solution.*[61]

2.5 Conclusions

To conclude, in this chapter we talked about all the related to our thesis-objective bibliography. The knowledge of all this background bibliography shall help us to go on to the next chapter. To sum up, the Semantic Web uses specific Data Models and Languages such as OWL, RDF, SPARQL and SOWL to describe semantic knowledge, and each of the above in a relevant or irrelevant way among them which depends on what data we have and what we want to do with them. Qualitative Spatial Reasoning, for instance, is a way to describe qualitative relations in our data. OWL Reasoners are engines usually used by the Semantic Web Frameworks that infer logical consequences from a set of asserted facts or axioms. In following chapters we will deal mostly with everything that is relative to spatial reasoning, and so RDF, SPARQL and RCC-8 are some some concepts that will be frequently used.

CHAPTER 3

ProtegePelletSpatial: A Protégé Plug-in for Spatial Reasoning

Protégé 4 allows different OWL reasoners [67] to be plugged in, the reasoner shipped with Protégé is called Fact++, but there are also other widely used reasoner plugins such as Pellet reasoner and RacerPro reasoner. The way that we use reasoners in Protégé is 'sending' the ontology to the reasoner to automatically compute the classification hierarchy, and also to check the logical consistency of the ontology. In Protégé 4 the 'manually constructed' class hierarchy is called the asserted hierarchy. The class hierarchy that is automatically computed by the reasoner is called the inferred hierarchy. To automatically classify the ontology (and check for inconsistencies) the 'Classify...' action should be used. This can be invoked via the 'Classify...' button in the Reasoner drop down menu shown in Fig. 3.1

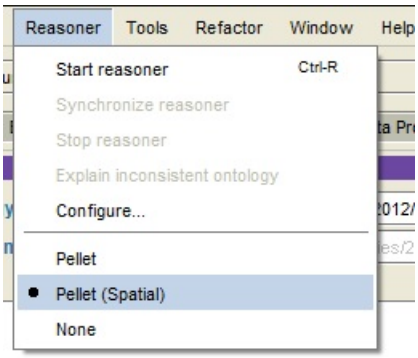

Fig. 3.1: *Ontology classification from the reasoner menu*

When the inferred hierarchy has been computed, an inferred hierarchy window will pop

open on top the existing asserted hierarchy window as shown in Figure 3.2

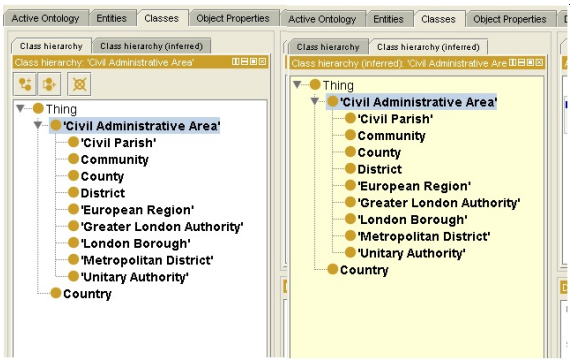

Fig. 3.2: *The Inferred Hierarchy Pane alongside the Asserted Hierarchy Pane after classification has taken place. The two images resulted from the "AdministrativeGeography" ontology*

If a class has been reclassified (i.e. if it's superclasses have changed) then the class name will appear in a blue colour in the inferred hierarchy. If a class has been found to be inconsistent it's icon will be highlighted in red.

Working with non-spatial data is almost the same for every common reasoner as they all use the classical axioms and relation. But when we cope with spatial data where there are triples not recognised from the OWL-DL reasoner, problems occur. This is why PelletSpatial was developed. PelletSpatial's purpose is to handle reasoning when given spatial input data. Yet, PelletSpatial is a library of Pellet engine which hasn't yet been imported in the main Pellet plugin for Protégé. And so, in this thesis framework we developed the plugin which reasons on spatial data using the PelletSpatial library of the Pellet engine.

In the following chapter's sections we are going to show the main aspects which consist one Protégé reasoner and especially our ProtegePelletSpatial reasoner.

3.1 Handling Spatial Ontologies

Spatial relations are expressed in RDF/OWL and can be combined with standard RDF/OWL semantic relations forming an ontology. A relation is represented as a triple. The predicate will be one of the terms used to express a spatial relation, while the subject and object will be the regions or the points involved in the relation. In OWL, such a statement is called an "Object Property Assertion Axiom" (e.g., Individual:Region1 ObjectProperty:disconnectedFrom Individual:Region2).

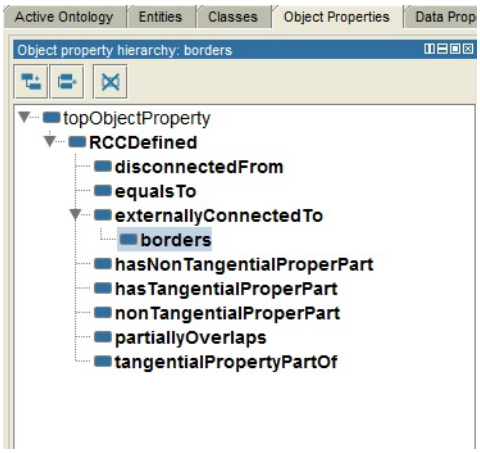

Fig. 3.3: *RCC-8 as object properties, in Protégé*

ProtegePelletSpatial plugin defines an RDF/OWL vocabulary for expressing qualitative spatial relations, with the RCC-8 topological model. As illustrated in Fig. 3.3, RCC terms are defined as simple object properties with no extra characteristics(e.g., inverse, transitive). One can either use the vocabulary provided, or use its own by defining sub-property axioms. Katz et al. [68] propose representing RCC-8 as OWL-DL class axioms [69]:[75], but this approach does not scale-up well for many relations [1].

3.2 ProtegePelletSpatial Architecture

In ProtegePelletSpatial, spatial knowledge is represented by an OWL ontology. In brief,

- ◇ we represent every region as an OWL individual (e.g., Individual:Town1, Individual:Town2 of type Class:Town)
- ◇ we define an OWL object property for each spatial relation (see Fig. 3.3)
- ◇ a spatial relation between two regions is represented as an OWL object property assertion (e.g., Individual:Town1 ObjectProperty:externallyConnected Individual:Town2).

Non-spatial relations, such as region type, size, etc., are represented as ordinary OWL assertions (e.g., Individual:Town1 DatatypeProperty:hasName Individual:Athens). This approach helps us to support reasoning and querying for both spatial relations and standard RDF semantic relations by setting apart each problem. Thus, ProtegePelletSpatial because of

the PelletSpatial hybrid engine, separates spatial reasoning from semantic OWL-DL reasoning by using an exclusive spatial reasoner component (CT Reasoner).

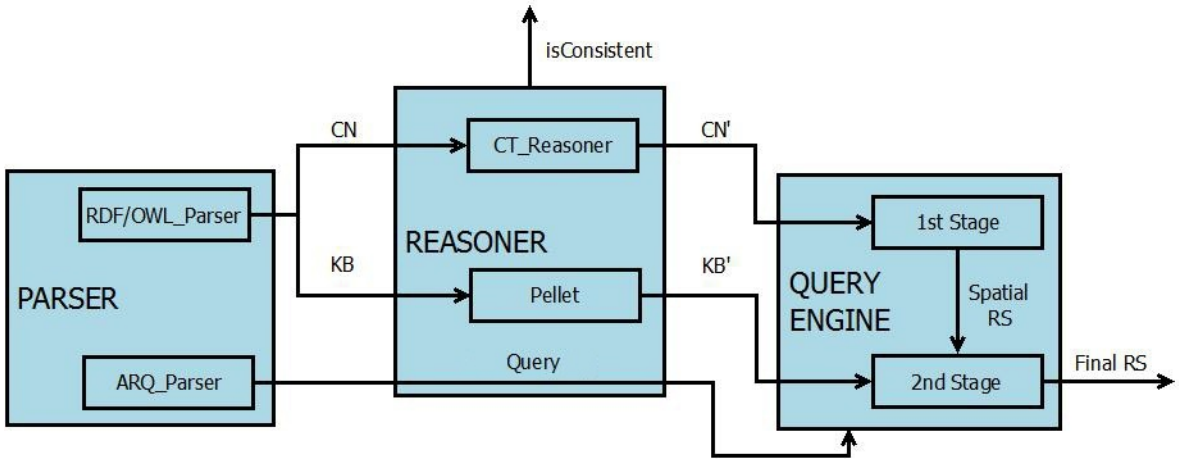

Fig. 3.4: *Main Components of the ProtegePelletSpatial reasoner*[51]

Fig. 3.4 illustrates the main components of the ProtegePelletSpatial reasoner. Ontologies are loaded into the Parser after a step of validation. This step ensures that all resources have a valid triple form. During the loading phase (RDF-Parsing), spatial property assertions are put into a constraint network (CN) and non-spatial standard OWL assertions are stored in a knowledge base (KB). The core of the system is the Reasoner component. The Composition Table Reasoner (CT Reasoner) checks the consistency of a constraint network, while Pellet checks the consistency of the knowledge base.

The queries are also loaded after validation. During the loading phase (ARQParsing), a query structure (SPARQL Query) is created in order to compartmentalize query atoms into spatial and non-spatial. Consistency check is activated every time we send a query. The Query Engine component applies a dual stage query answering technique, but it can also easily be parameterized. The First Stage returns a set of spatial query results (spatial RS). This set is given as input to the Second Stage, consisting of further constraining such that the non-spatial query is satisfied. Thus, we get the final set of query results (final RS).

Above was described the main idea of reasoning architecture of PelletSpatial. In the next three subsections we are going to describe in more details each of the ProtegePelletSpatial architecture components. But first, we will give a short explanation of some commonly used terms:

Knowledge base (KB) is a special kind of database for knowledge management, providing the means for the computerized collection, organization, and retrieval of knowledge. The

Pellet KB is a combination of an assertional box (component that contains assertions about individuals) and a terminological box (component that contains axioms about classes), which provides consistency checking and query services.

A constraint network (CN) is a set of variables together with a set of constraints and perhaps also one or more objective functions. Spatial constraint networks allows to store and handle spatial knowledge by providing similar to a KB functionality for checking consistency as well as for querying.

3.2.1 Parser

ProtegePelletSpatial implements an RDF and an ARQ parser for parsing ontologies and queries respectively. The RDF parser extracts RCC-8 spatial triples from the RDF graph for creating their respective RCC-8 constraint networks for checking the consistency of spatial relations defined in the ontology and for query answering. The spatial triples in the RDF graph are loaded in their respective constraint RCC-8 network. The rest of the graph (non-spatial relations) is handled by Pellet's KB, a component that contains assertions about individuals (ABox) and axioms about classes (TBox).

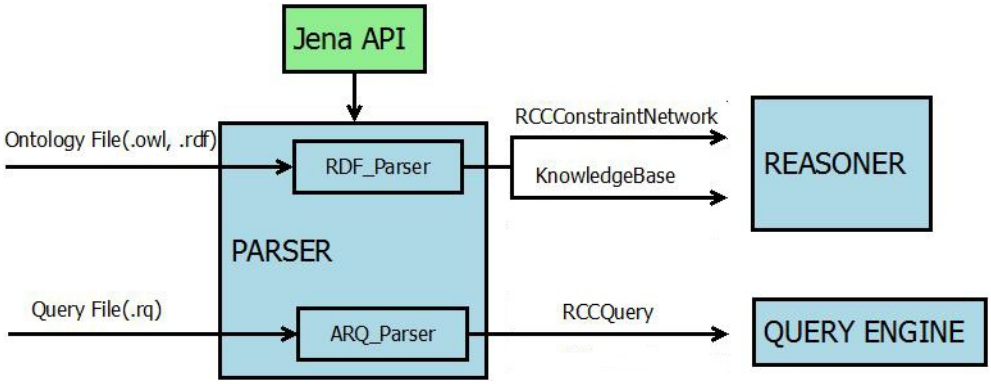

Fig. 3.5: *Parser components*

RDF parsing is implemented using Jena API and internal spatial vocabularies denoting the relations of the RCC-8 formalisms. In query (ARQ) parsing, query atoms are characterized as RCC-8 or non-spatial(Fig. 3.5).

3.2.2 Reasoner

The core of the system is the Reasoner component. ProtegePelletSpatial strictly separates spatial reasoning from semantic OWL-DL reasoning, as it uses one exclusive reasoner for each calculus: RCC relations are managed as a RCC constraint network, while common owl-dl relations are managed by the Knowledge Base of Pellet.

In RCC-8 qualitative formalisms, relations are expressed based on a set of jointly exhaustive and pairwise disjoint basic relations which is closed under several operations. Thus, it is possible to apply constraint based methods for reasoning over these relations. For this, it is necessary to give a composition table either for all relations, or for the basic relations only together with procedures for computing the compositions of complex relations. A composition table is defined using the formal semantics of the relations. Otherwise it is not possible to verify correctness and completeness of the inferences. Formal semantics of the relations are also necessary for finding efficient reasoning algorithms which are essential for most applications. Without formal semantics it is sometimes not even possible to show that reasoning over a system of relations is decidable.

In our implementation, consistency checking for the constraint network is performed by means of a path-consistency algorithm that we saw in algorithm 2.1. This algorithm is based on the corresponding composition table for RCC-8 (see Table 2.2) relations.

3.2.3 Query Engine

As ProtegePelletSpatial processes both spatial and standard semantic OWL relations in RDF/OWL documents, it is natural to support spatial querying. Query Engine component(Fig.3.6) answers conjunctive queries that include spatial and non-spatial patterns, i.e., triple patterns for spatial relations (joined) with triple patterns for semantic RDF relations.

More specifically, this module supports a subset of queries written in SPARQL, that satisfies the following conditions:

- ◇ No variable is used in the predicate position.

- ◇ Each property used in the predicate position is either a property (object or datatype) defined in the ontology or one of the following built-in properties: rdf:type, owl:sameIndividualAs, owl:differentFrom.

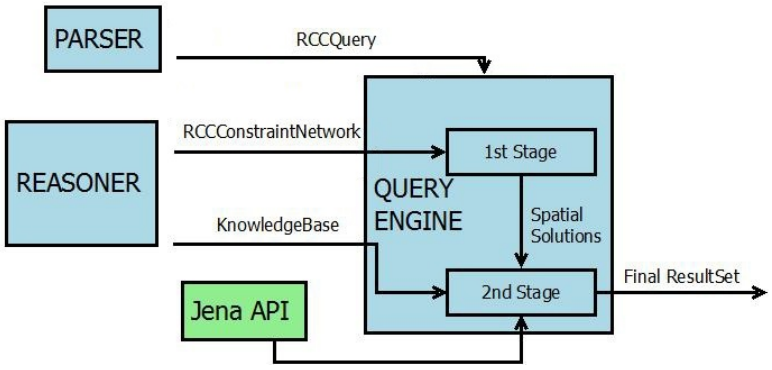

Fig. 3.6: *Query Engine components*

◇ At least one of the triples must contain a "spatial" object property in the predicate position.

Let us illustrate PelletSpatial's query engine function by using an example [70]. Consider a DreamHouse that is located inside a pine forest and borders a lake.

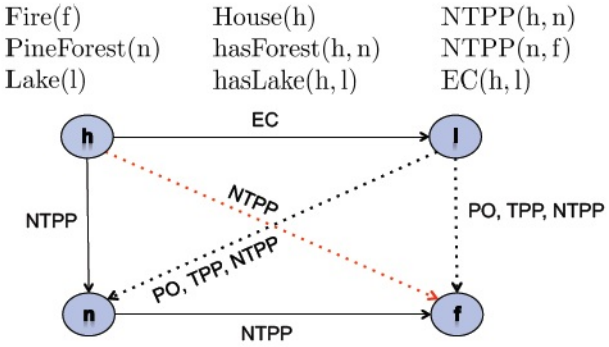

Fig. 3.7: *Query Engine example: The black lines(asserted relations) show the initial state of ABox and the dotted lines show the inferred relations after reasoning in ABox's data. First, the black dotted line was reasoned and then the red dotted.*[70]

An $\mathcal{ALC}$(RCC-8) approach would describe the above DreamHouse as follow:

DreamHouse $\equiv$ House $\sqcap$ $\exists$(loc),(hasLake loc).EC $\sqcap$ $\exists$(loc),(hasForest loc).NTPP $\vee$ TPP

DreamHouse $\sqsubset$ $\forall$ hasForest.PineForest $\sqcap$ $\forall$hasLake.Lake.

So, consider the ABox that follows: Fire(f), PineForest(n), Lake(l), House(h), hasForest(h,n) hasLake(h,l), NTPP(h,n), NTPP(n,f), EC(h,l)

And now the question we want to ask is what are the houses that are threatened?

The answer is h and Fig. 3.7 shows how reasoner came to this result.

3.3 Java Implementation Synopsis

In this section we are going to present some useful information about our implementation of ProtegePelletSpatial reasoner plugin for Protégé. We are going to describe the functionality of the most important classes of the PelletSpatial library of Pellet engine and how we compined that with the Protégé framework. Some useful prior knowledge that could help is that PelletSpatial's library[1] is fully adaptable as it includes a lot of interfaces so that one can use it in whichever way he would like to. PelletSpatial's library is distributed in packages that each of them has a separate functionality. The most functional and useful packages are those that handle the reasoner machine, the parser and the query engine that we presented in the previous section. Withal useful is the translator package whose classes translate the RCC-8 relations to be reasoned as being spatial axioms and not simple OWL-DL axioms. At follow, we will present the functionality of each package of these three.

To begin with, we will describe the packages as shown below:

1. The Reasoner package provides us with *RCCReasoner* interface and some implemented classes and the one that we are going to describe is this of *RCCCompositionTableReasoner*. The functionality of this class is to take all asserted regions of the network and all the asserted relations between them and make the graph that they construct, complete. This is done by adding new relations among the regions. These would be:

   - ◇ Adding the inverse relations to the network. (e.g. for *EC(r1,r2)* add *EC(r2,r1)*)
   - ◇ Adding equal relations for all regions with themselves. (e.g. for *r1* add *EQ(r1,r1)*)
   - ◇ Adding new relations inferred by existing relation (e.g. for existing relations *Rx(a,b), Sy(b,c)* add *Tz(a,c)*). This step is applied recursively by using path-consistency algorithm for every *Rx(a,b), Sy(b,c), Tz(a,c)* relation.

   Finally the inferred network is checked for consistency.

2. The Parser package includes the *RCCParser* interface which is implemented by a variety of other classes. *RCCRDFParser* is one of these and takes as parameters the RCC-8

[1] http://clarkparsia.com/pellet/spatial/pellet-spatial-0.1

network and the ontology model. The functionality of this class is to parse the file with the RDF description of our data and do the following three operations:

- ◇ Find all the RCC-8 relation terms that are described with a specific RDF vocabulary and add them to the spatial RCC graph giving them the proper RCC relation type (e.g. such an RDF relation term is *"disconnectedFrom"* and the given RCC relation type is *"DC"*)
- ◇ Get all the triples from the graph and add all triple relations that are of RCC relation type to asserted network (CN).
- ◇ Get all the remaining triples from the graph and them all to Pellet's Knowledge Base(KB).

3. The *Query Engine* package includes the *RCCQueryEngine* interface which is implemented by a variety of other classes. *RCCQueryEngineFactory* is one of these and takes as parameters the name of the file that contains the query. Then after having parsed the main RDF ontology and having inferred the CN and the KB, Query Engine either uses *"Dual Stage Query Engine"* or *"Pellet Query Engine"* after having translated the RCC network for Pellet. Now a brief description of these two is that *RCCQueryEngineDual* makes both spatial and non-spatial querying on data simultaneously regarding that the two result set are distinct one another. Respectively, *RCCQueryEnginePellet* makes at first the spatial querying and then in the same primary set of nodes makes the non-spatial querying, it being more secure but also more overdue.

Thereafter, having already described the most important classes of PelletSpatial packages, we come to plug them in Protégé reasoner interface so that we produce our ProtegePelletSpatial reasoner plugin. As it has already been cited [58], [67], Protégé consists of a combination of plugins (e.g. Reasoner plugins, Tab plugins, Menu plugins, Frame plugins). And so, what we are called uppon to implement is the interface of the *ProtegeReasoner* plugin, the *ProtegeOWLReasonerInfo*, and to extend the OWLWorkspaceViewsTab and the AbstractOWLViewComponent, so as to see the spatial result after the classification in a seperate view tab of Protégé 4.1. A big segment of our implemented code for the tab plugin is provided below including in comments the functionality that they give to Protégé:

```
1 public class PelletSpatialViewComponent extends AbstractOWLViewComponent {
2     private static final long serialVersionUID = -4515710047558710080L;
3
```

```
4       private static final Logger log = Logger.getLogger(ExampleViewComponent.class);
5
6       private Metrics metricsComponent;
7
8       @Override
9       protected void initialiseOWLView() throws Exception {
10          setLayout(new BorderLayout());
11          add(new JLabel());
12          metricsComponent = new Metrics(getOWLModelManager());    //here we use the class of
                Metrics which internally does all the functionality of the pellet spatial
               reasoning
13         add(metricsComponent, BorderLayout.WEST);
14          log.info("PelletSpatial View Component initialized");
15      }
16
17          @Override
18          protected void disposeOWLView() {
19                  metricsComponent.dispose();
20          }
21
22  }
23
24
25  public class Metrics extends JPanel {
26
27      //DECLARATIONS
28      private static final long serialVersionUID = -2017045836890114258L;
29          private static final Logger log = Logger
30                          .getLogger(Metrics.class.getName());
31      private org.protege.editor.owl.model.OWLModelManager modelManager;
32      ...
33          private ActionListener refreshAction = new ActionListener(){
34          ...
35      };
36      private org.protege.editor.owl.model.event.OWLModelManagerListener modelListener = new
            OWLModelManagerListener(){
37                  ...
38      };
39
40
41      public Metrics(org.protege.editor.owl.model.OWLModelManager modelManager) {
```

```
42          //INITIALISATIONS
43              ...
44          //Creation of the check boxes.
45              ...
46          //Creation of the refresh button
47              ...
48          //Creation of a text areas and text editors
49              ...
50          //Adding all the above in the BorderLayout
51              ...
52          //setting the action listener
53          this.modelManager.addListener(modelListener);
54      }
55
56      public void dispose() {
57          //function used to dispose when terminating the tab plugin
58              ...
59          refreshButton.removeActionListener(refreshAction);
60          modelManager.removeListener(modelListener);
61      }
62
63
64      private void recalculate() {
65          //the functions that does the pellet spatial's reasoning job
66
67          //INITIALIZATIONS
68              ...
69          RCCCompositionTableReasoner spatialreasoner = null;
70
71                  // Show the progress monitor for composition table consistency checking
72                  // A file is added to the jar so that understand the option changing.
73          PelletSpatialOptions.SHOW_COMPOSITION_TABLE_REASONER_PROGRESS = option1;
74                  PelletSpatialOptions.USE_COMPOSITION_TABLE_REASONER = option2;
75                  PelletSpatialOptions.ADD_INTERIOR_POINTS_OF_REGION_AXIOM = option3;
76                  PelletSpatialOptions.USE_DUAL_STAGE_QUERY_ENGINE = option4;
77                  PelletSpatialOptions.USE_REGION_PARTITIONS = option5;
78
79                  RCCConstraintNetwork network = new RCCConstraintNetwork();
80                  OntModel model = ModelFactory.createOntologyModel(PelletReasonerFactory.
                        THE_SPEC);// create an empty ontology model using Pellet spec
81
```

```
82                RCCParser parser = RCCParserFactory.getRDFRelationParser(network, model);
83                parser.parse(data);
84
85                System.out.println("State before consistency checking");
86                textArea1.append("\n\n Number of asserted spatial relations in network: "+
                      network.size());
87                System.out.println("Number of spatial relations in network: " + network.
                      size());
88                if(network.size() < 100){
89                        textArea1.append("\n Network asserted spatial relations :\n");
90                        for(RCCRelation rccRelation : network.getAllRelations()){
91                                textArea1.append(rccRelation.getRegion1()+"\t"+rccRelation
                                      .getRegion2()+"\t\t"+rccRelation.getTypes()+"\n");
92                        }
93                        textArea1.append("\n\n Network asserted spatial definite relations
                             :\n");
94                        for(RCCRelation rccRelation2 : network.getDefiniteRelations()){
95                                textArea1.append(rccRelation2.getRegion1()+"\t"+
                                      rccRelation2.getRegion2()+"\t\t"+rccRelation2.getTypes
                                      ()+"\n");
96                        }
97                        textArea1.append("\n\n Network regions :\n");
98                        for(RCCRegion rccRelation : network.getRegions()){
99                                textArea1.append(rccRelation+"\n");
100                       }
101               }
102               else{
103                       textArea1.append("\n Because of the big network size (>30)
                            relations and regions \nwill not be printed in screen. \n
                            Instead you can try asking a sparql query after network is
                            classified\n");
104               }
105
106
107       long start = System.currentTimeMillis();
108               boolean isConsistent = network.isConsistent();
109       log.info("Pellet Spatial classified in " + (System.currentTimeMillis()-start) + "
              ms");
110               textArea1.append("Pellet Spatial classified in " + (System.
                      currentTimeMillis()-start) + "ms");
111               textArea2.append("Pellet Spatial classified in " + (System.
```

```
                    currentTimeMillis()-start) + "ms");
112
113                 System.out.println("State after consistency checking");
114                 textArea2.append("\n\n Number of inferred spatial relations in network: "+
                        network.size());
115                 if(network.size() < 100){
116                         textArea2.append("\n Network inferred spatial relations :\n");
117                         for(RCCRelation rccRelation : network.getAllRelations()){
118                                 textArea2.append(rccRelation.getRegion1()+"\t"+rccRelation
                                        .getRegion2()+"\t\t"+rccRelation.getTypes()+"\n");
119                         }
120                         textArea2.append("\n\n Network inferred spatial definite relations
                                :\n");
121                         System.out.println("\n\n Network inferred spatial definite
                                relations :\n");
122                         for(RCCRelation rccRelation2 : network.getDefiniteRelations()){
123                                 textArea2.append(rccRelation2.getRegion1()+"\t"+
                                        rccRelation2.getRegion2()+"\t\t"+rccRelation2.getTypes
                                        ()+"\n");
124                         }
125                         textArea2.append("\n\n Network regions :\n");
126                         for(RCCRegion rccRelation : network.getRegions()){
127                                 textArea2.append(rccRelation+"\n");
128                         }
129                 }
130                 else{
131                         textArea2.append("\n Because of the big network size (>30)
                                relations and regions \nwill not be printed in screen. \n
                                Instead you can try asking a sparql query after network is
                                classified\n");
132                 }
133
134                 System.out.println("PelletSpatial Version: " + PelletSpatialVersion.
                        version);
135
136         //query
137                 RCCQueryEngine qe = RCCQueryEngineFactory.create(data);
138                 FileOutputStream queryFile = null;
139                 try {
140                         queryFile = new FileOutputStream("queryFile.rq");
141                 } catch (FileNotFoundException e) {
```

```
142                         // TODO Auto-generated catch block
143                         e.printStackTrace();
144                 }
145                 new PrintStream(queryFile).println(editorPane.getText());
146 
147                 ResultSet rs = qe.query("queryFile.rq");
148                 textPane.setText(ResultSetFormatter.asText(rs));
149                 ResultSet rs1 = qe.query("queryFile.rq");
150                 ResultSetFormatter.out(rs1);
151 
152 
153 
154         int count = modelManager.getActiveOntology().getClassesInSignature().size();
155             ...
156         textArea1.append("\n\n Total classes = "+ count +"\n");
157 
158         int count2 = modelManager.getActiveOntology().getAxiomCount();
159             ...
160         textArea1.append("Total axioms = "+ count2 +"\n");
161 
162         int count3 = modelManager.getActiveOntology().getLogicalAxiomCount();
163             ...
164         textArea1.append("Total logical axioms = "+ count3 +"\n");
165 
166         int count4 = modelManager.getActiveOntologies().size();
167             ...
168         textArea1.append("Number of Active Ontologies = "+ count4 +"\n");
169 
170         int count5 = modelManager.getDirtyOntologies().size();
171             ...
172         textArea1.append("Number of Dirty Ontologies = "+ count5 +"\n\n");
173     }
174 }
```

3.4 Conclusions

To conclude, in this chapter we talked about our thesis-objective implementation. More specifically we show how Qualitative spatial reasoning is handled and translated by a reasoning engine and especially for our reasoner plugin we analysed all its architecture and also sho-

wed some figures of how to use this plugin in Protégé. As a final section to this chapter we provided a code-section where we explained some of PelletSpatial's most useful classes and we presented some code segments of the Protégé classes that we called uppon to implement. In the next chapter we present our experimental evaluation with a case study we conducted.

CHAPTER 4

Experimental Evaluation

In this chapter we present a case study which we decided to elaborate with purpose to check PelletSpatial performance in reasoning and querying while using big spatial ontologies, having as our criterion the time complexity of reasoning. We performed our case study using data taken from an ontology, the "Administrative Geography Ontology"[1], that uses qualitative spatial description of regions in OWL/RDF relations using the RCC-8. Subsequently, we present a user guide for usage of our ProtegePelletSpatial Tab implemented plugin. And for summing up, we come to make a conclusion and we discuss and evaluate possible optimizations of PelletSpatial engine.

4.1 Case Study

Our case study ontology "Administrative Geography Ontology" is a mechanism to describe data related to spatial entities of the administrative and voting area geography of Great Britain[2]. In this ontology, simple models are described in puprose to address administrative geography RDF data using RDF containers [81]. The data used of the basis of this model were OS[3] UK Administrative Geography data containing information resources describing administrative units (such as boroughs, regions, and districts) and their associated properties (such as areas, identification numbers, and topological relationships to other administrative units).

[1] http://www.ordnancesurvey.co.uk/oswebsite/ontology/AdministrativeGeography/v2.0/AdministrativeGeography.rdf

[2] http://data.ordnancesurvey.co.uk/ontology/admingeo/

[3] http://data.ordnancesurvey.co.uk/.html

A description of the hierarchy of administrative units and the supported between RDF individuals relationships follows [82].

The Class hierarchy of the Administrative Geography Ontology is:

- ◇ CivilAdministrativeArea
  - ○ EuropeanRegion
  - ○ Country
  - ○ UnitaryAuthority
  - ○ MetropolitanDistrict
  - ○ GreaterLondonAuthority
  - ○ LondonBorough
  - ○ District
  - ○ CivilParish
  - ○ Community
- ◇ Country

And the distinct topological relations between Administrative Geography Ontology's regions are:

- ◇ Completely Spatially Contains: The interior and boundary of one region is completely contained in the interior of the other region.(corresponds to *NTPPi*)
- ◇ Tangentially Spatially Contains: The interior of one region is completely contained in the interior or the boundary of the other region and their boundaries intersect.(corresponds to *TPPi*)
- ◇ Borders: The boundaries of two regions intersect but their interiors do not intersect.(corresponds to *EC*)
- ◇ Spatially Equivalent: The two regions have the same boundary and interior.(corresponds to *EQ*)

As one can see *PO* does not exist is relations, and so it would be proper to explicate that the topology of the administrative geography of Great Britain contains no overlapping regions resulting that the RCC-8 property *PO* was not required. Consequently, the property hierarchy for topological relationships in the administrative geography RDF is as follows:

- ◇ spatiallyContains
  - ○ compeletelySpatiallyContains
  - ○ tangentiallySpatiallyContains
- ◇ isSpatiallyEqualTo
- ◇ borders

Fig. 4.1 illustrates the map of Great Britain's Administrative Geography[4]. Information in this map is used to acquire the necessary concepts for defining a basic hierarchical structure. This ontology's qualitative spatial relations between regions, which are defined as shown in Fig. 4.2, consist of classes, properties and individuals, all of which we discusses previously.

4.2 ProtegePelletSpatial User Guide

This section deals with the installation of our ProtegePelletSpatial plugin, the activation of the ProtegePelletSpatial Tab, the ontology loading and the description of the tab's layout and the functions that it accords with for spatial reasoning and querying. In this way all the above compose a simple user guide to run into for clarifications of the ProtegePelletSpatial plugin. We used our case study for the examination of our tabs' results given a case of a big ontology loaded in ProtegePelletSpatial and the example ontology "USAMappings", used with the Pellet Spatial reasoner, given a case of a small ontology. All the ProtegePelletSpatial plugin implementation material (source code, executable jar, case study ontology, example queries) is included in the accompanying media (CD) of this thesis deposited to the library.

[4] http://upload.wikimedia.org/wikipedia/commons/d/d6/Map_of_the_administrative_geography_of_the_United_Kingdom.png

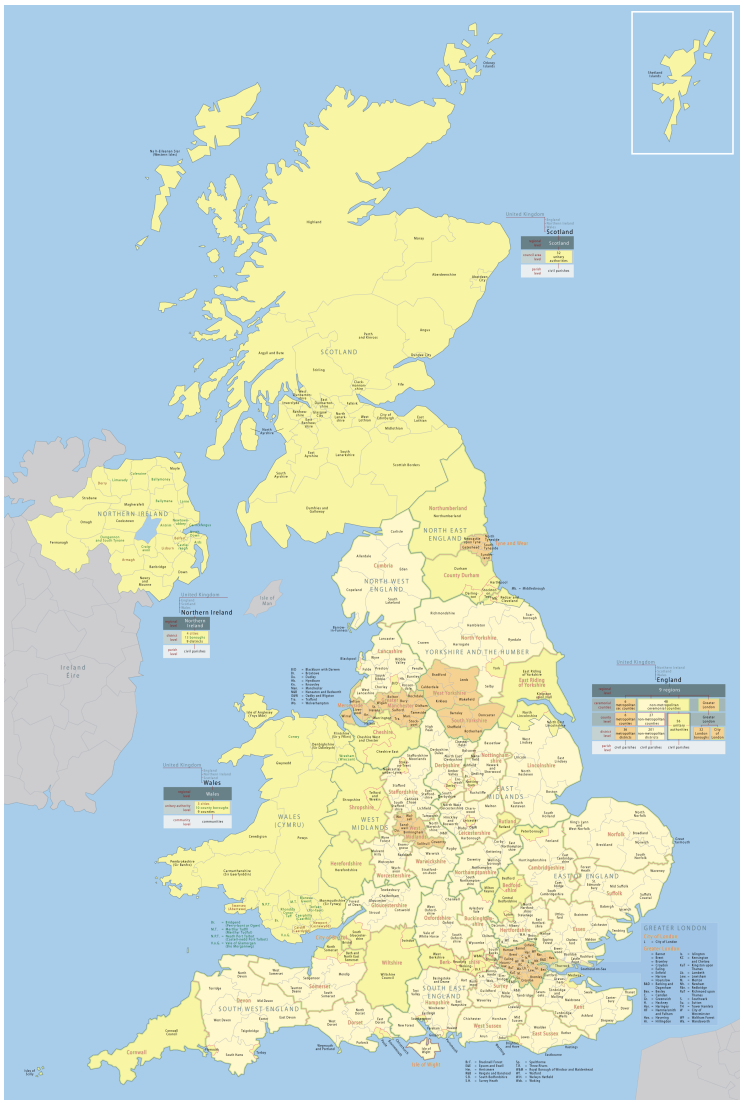

Fig. 4.1: *The map of Great Britain's Administrative Geography*

4.2.1 Plugin Installation

The ProtegePelletSpatial plugin is built over the Protégé 4.1 platform and this documentation will continuously refer to this platform. As a consequence of this it is recommended that you use this platform or any Protégé 4.x platform. In view of the above the only thing that one needs to do is to place the ProtegePelletSpatial packaged file, with name gr.uoa.di.protege.tab_1.0.3.jar (or a newer version probably), in the folder *"plugins"* which is located in the file System directory that Protégé 4.x is installed, e.g. *C:\Program Files\Protege_4.1\plugins*. Having done so, the next time Protégé 4.x is executed the Protege-

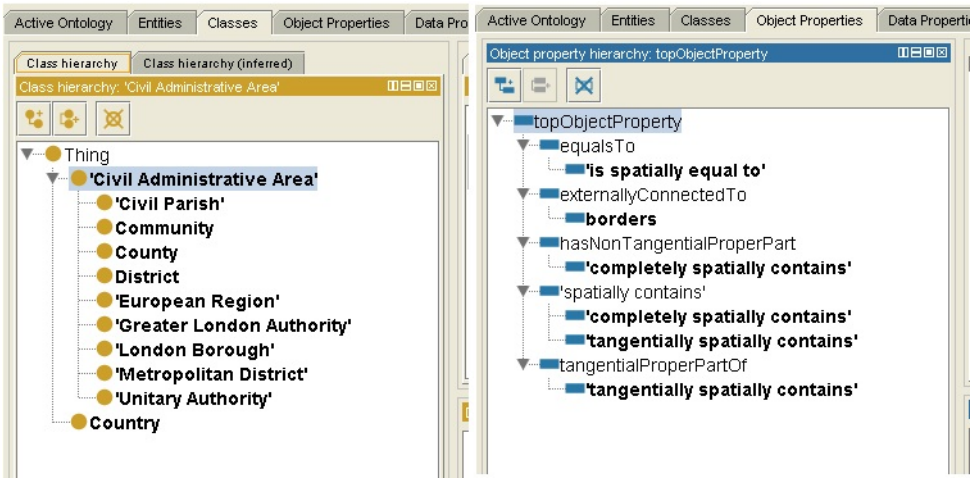

Fig. 4.2: *Administrative Geography Classes and Object Properties*

PelletSpatial plugin will be installed with the name set as "Protege Pellet Spatial Tab"(Fig.4.3). A more informal way to get sure of having properly installed the plugin is when some printing such as "getreasonername", "getreasonerid" are shown in the console(Fig.4.4).

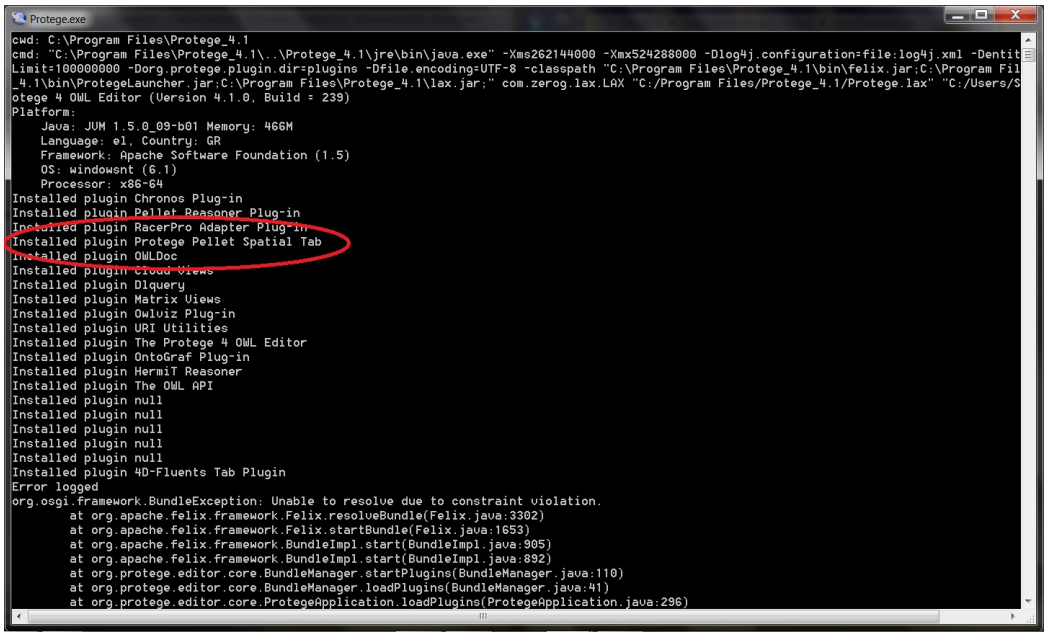

Fig. 4.3: *ProtegePelletSpatial Plugin Installation. The Protege Pellet Spatial Tab plugin is installed in Protégé 4.1.*

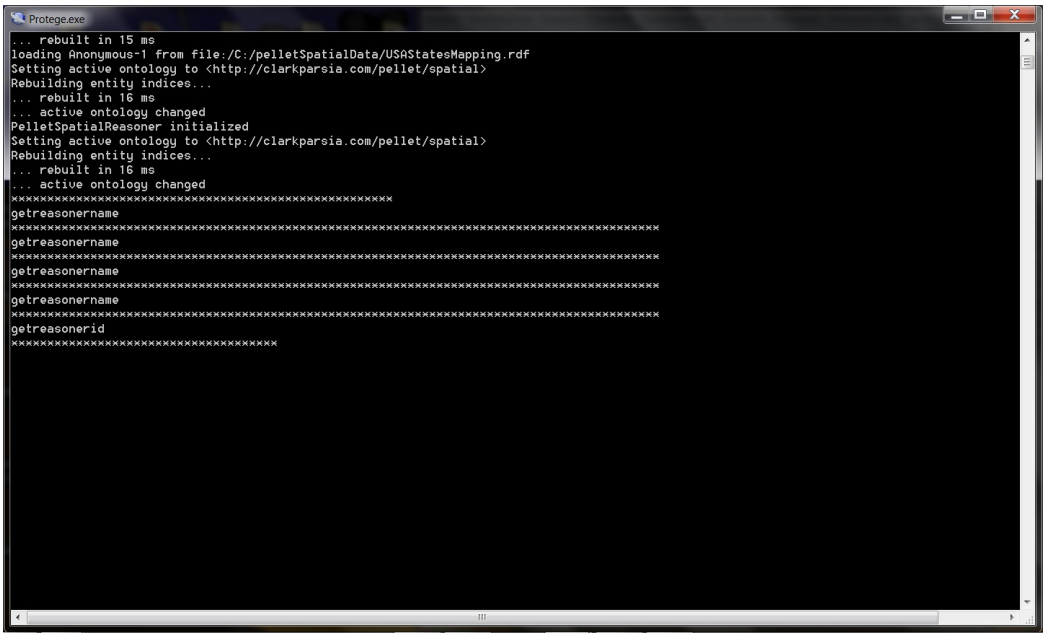

Fig. 4.4: *ProtegePelletSpatial Plugin Installation Verification. If "getreasonername" and "getreasonerid" printings are shown in console then that is an uninformal way to verify that plugin installation completed successfully*

4.2.2 ProtegePelletSpatial Tab Activation

Once ProtegePelletSpatial plugin is installed in Protégé then when the main window of Protégé is loaded a tab with name "ProtegePelletSpatial Tab" must be added to the bar of tabs(Fig.4.5). In case that the tab is not shown in the bar of tabs then one has to set the tab visible manually by clicking on the menu *"Window → Tabs → ProtegePelletSpatial Tab"* (Fig.4.6).

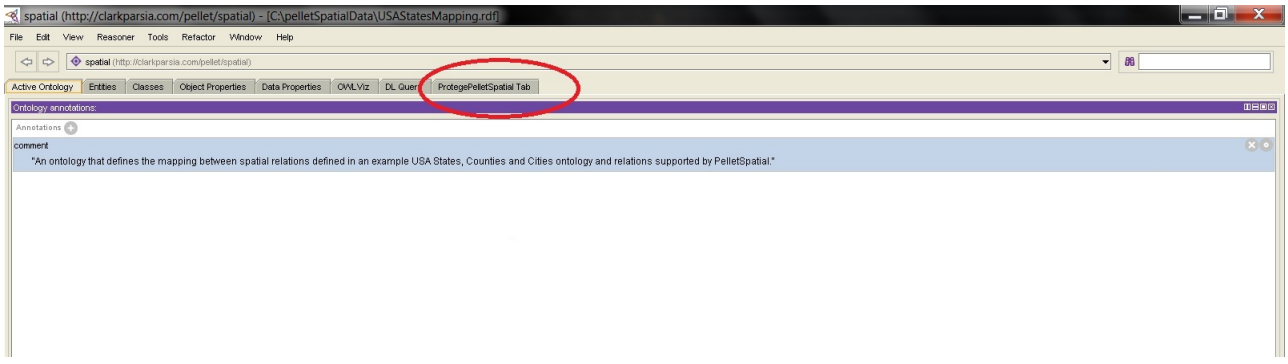

Fig. 4.5: *Protégé Active Ontology Window*

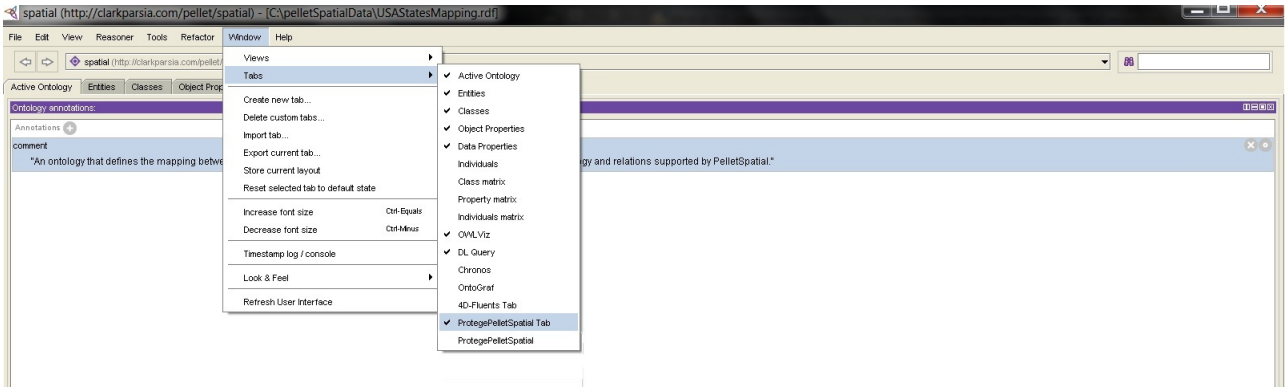

Fig. 4.6: *ProtegePelletSpatial Tab Set Visible*

4.2.3 Spatial Ontology Loading

Having all the above working fine it is time to load a spatial ontology in Protégé so that to classify it, to check its spatial consistency and ask sparql queries. Ontology loading is done with the classic way of Protégé. By clicking on the menu *"File → Open..."* and selecting the place in the disk where your spatial ontology is saved. In this version of our plugin we have implemented only methods that take the file system path of the ontology to be classified, so the option of loading a spatial ontology from internet(*"File → Open from URL..."*) is yet not implemented but in the next version of the plugin this inconsistency will be surpassed.

It is important to mention in this subsection that as we will see later in the next section one who uses the ProtegePelletSpatial Tab plugin, is given some option which allow him to classify not only spatial ontologies, but also simple OWL-DL ontologies. It is also important to take under consideration that if your ontology depends from other ontologies, to set properly those ontologies in the right directories so that the Pellet Spatial Reasoner finds them. For instance, the example ontology "USAMappings.rdf" depends from the ontology "USAStates.rdf" which must also be set in the same directory with the first.

4.2.4 ProtegePelletSpatial Tab Layout and Functionality

Having everything fine so far it is time we click on the ProtegePelletSpatial tab. As shown in Fig.4.7 ProtegePelletSpatial layout consists of five boxes, which are divided in two groups. The group on the left relates with Pellet Spatial Reasoning components, while the group on the right relates with Pellet Spatial Querying components.

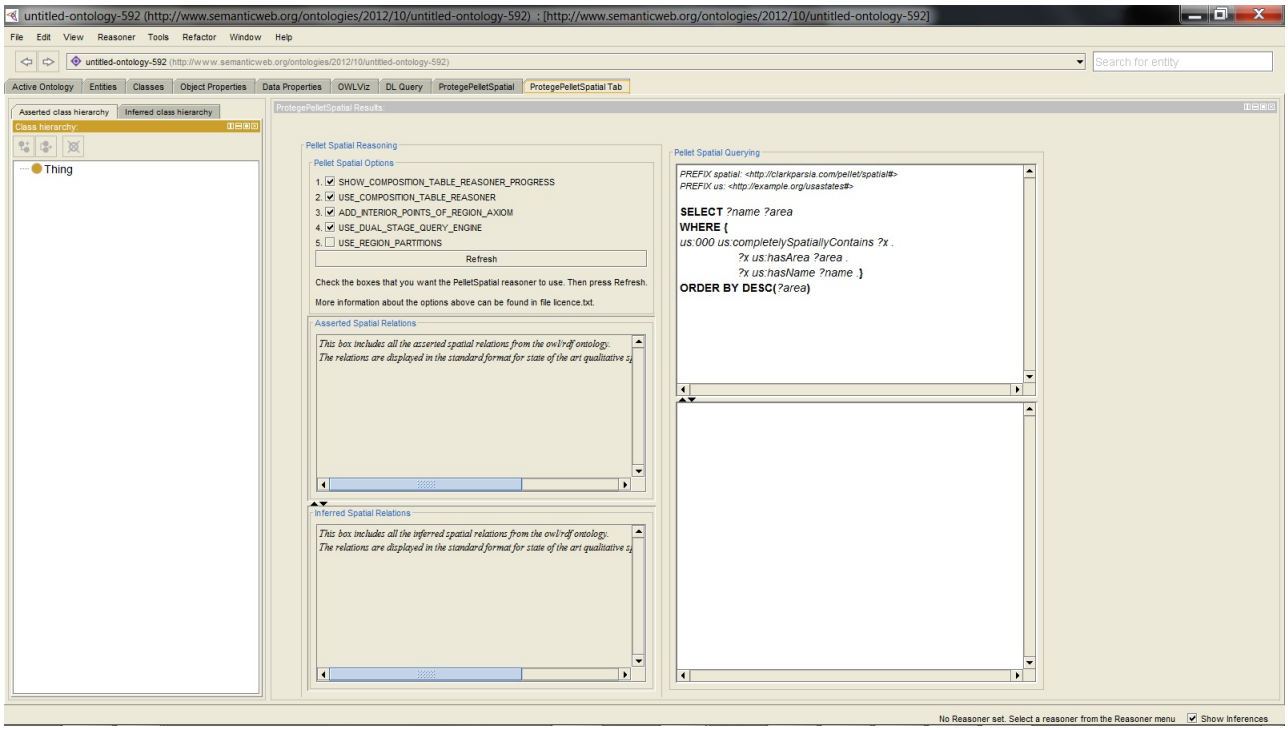

Fig. 4.7: *ProtegePelletSpatial Tab Layout*

4.2.4.1 ProtegePelletSpatial Tab Reasoning Layout and Functionality

To begin with, in the left group of boxes of the layout the three boxes show respectively, the Pellet Spatial Options, the Asserted Spatial Relations and the Inferred Spatial Relations.

The Pellet Spatial Options contain four reasoner affiliated option and one query affiliated. The use of these options is implemented by the use of checkboxes so that one can educe the results of his preference with any combination he likes. The most important of these options is the *USE_COMPSITION_TABLE_REASONER* which initially is given a *"true"* value. This option substantially tell the reasoner whether to use the RCC Reasoner with Pellet Reasoner or use only the Pellet reasoner. In other words, this option tells the reasoner whether it has to classify a spatial ontology or a non-spatial ontology. So be aware to change this option for non-spatial ontologies. The rest four options are explained briefly in the file *"spatialOptions.txt"* which is also included in the implementation material of the accompanying media (CD) of this thesis, which is deposited to the library.

To continue, having selected the options for classification the only thing that remains before classifying is press the button *"Refresh"*. *"Refresh"* button has been placed with an OWLActionListener which "listens" for every alteration in the current ontology. As a result,

every time we change the ontology with the way described in previous section if subsequently we press the *"Refresh"* button classifying should be done in the new ontology. *"Refresh"* button's main functionality after having been pressed, is to call Pellet Spatial Reasoner and Query Engine for doing consistency checking and spatial querying respectively.

The boxes of Asserted and Inferred Spatial Relations initially contain nothing and solely inform the ProtegePelletSpatial Tab user that the format of the results follows the standard format for state of the art qualitative spatial reasoners, like the Renz reasoner for RCC8.[83]

After pressing *"Refresh"* button each of the two boxes is filled in with the fundamental results that show respectively the asserted/inferred spatial relations, the asserted/inferred spatial definite relations and the asserted/inferred network regions. Complementary to the above, some additional useful information presented is the ontology DocumentIRI and name, the values set in the spatial options used in each progress, the time elapsed for the network classification consistency checking and some other ontology statistics. (Fig.4.8, Fig.4.9)

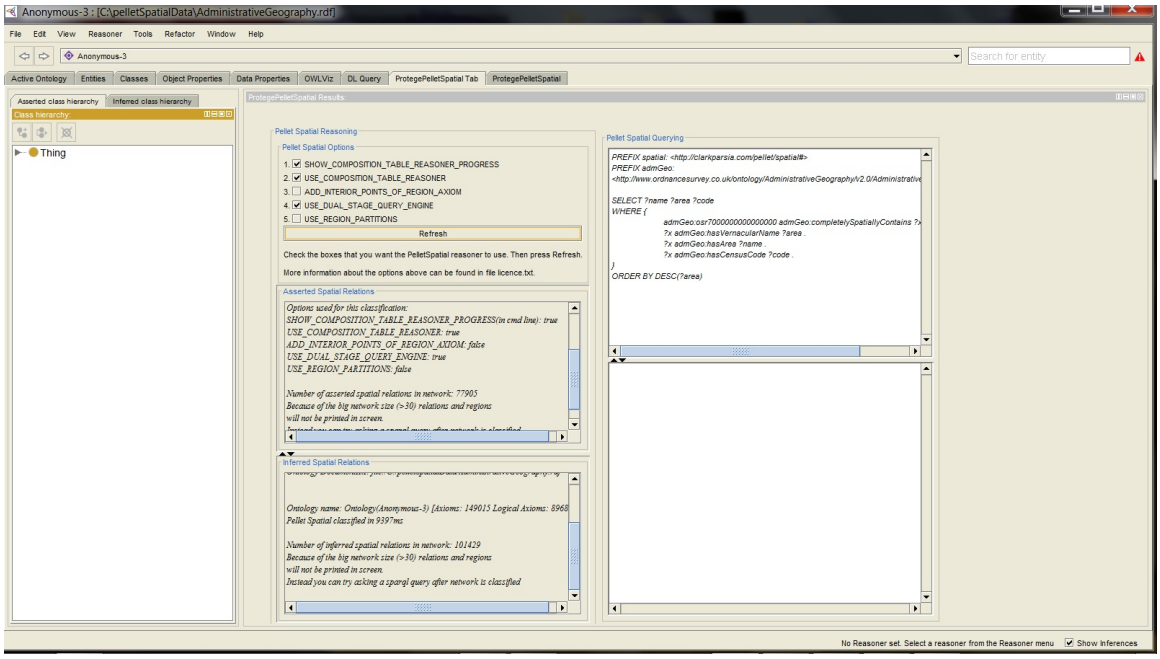

Fig. 4.8: *ProtegePelletSpatial Tab Layout: AdministrativeGeography ontology. The output format we used follows the standard format for state of the art qualitative spatial reasoners, like the Renz reasoner for RCC8.*[83]

Finally in this subsection it is important to mention that ontologies which have over one hundred relations in their network are not printed in detail because system may crash. As opposed to this we recommend to ask a sparql query to find the relations of a region.

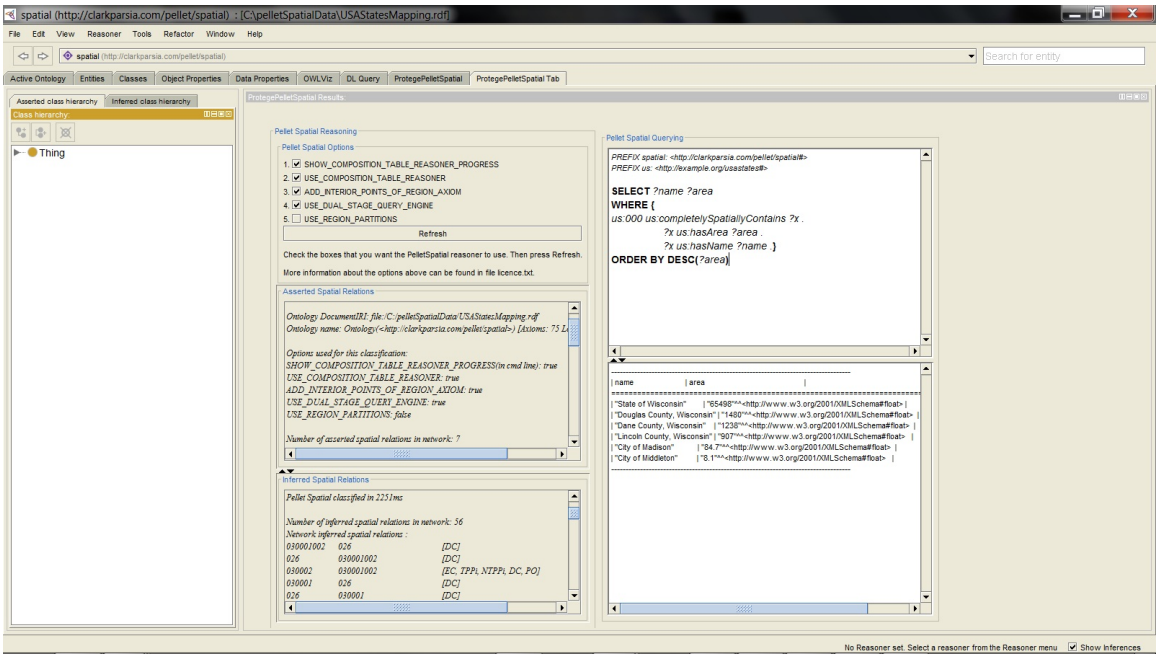

Fig. 4.9: *ProtegePelletSpatial Tab Layout: USAMappings ontology. The output format we used follows the standard format for state of the art qualitative spatial reasoners, like the Renz reasoner for RCC8.*[83]

In addition to that, it would be beneficial to inform you that some big ontologies, such like our "AdministrativeGeography" ontology because of their relations' size and number may not fit in Protégé's default heap size. This must be changed manually by following the protegewiki's[5] instructions and set the *"Protege.lax"* as shown in Fig.4.10, having taken into account not to set heap size parameter larger than about 80% of your computer free physical memory. It may also be needed to set the user permissions if the directory you have Protégé 4.x installed has a direct relationship with System's operating files(Fig.4.11).

[5]http://protegewiki.stanford.edu/wiki/Setting_Heap_Size

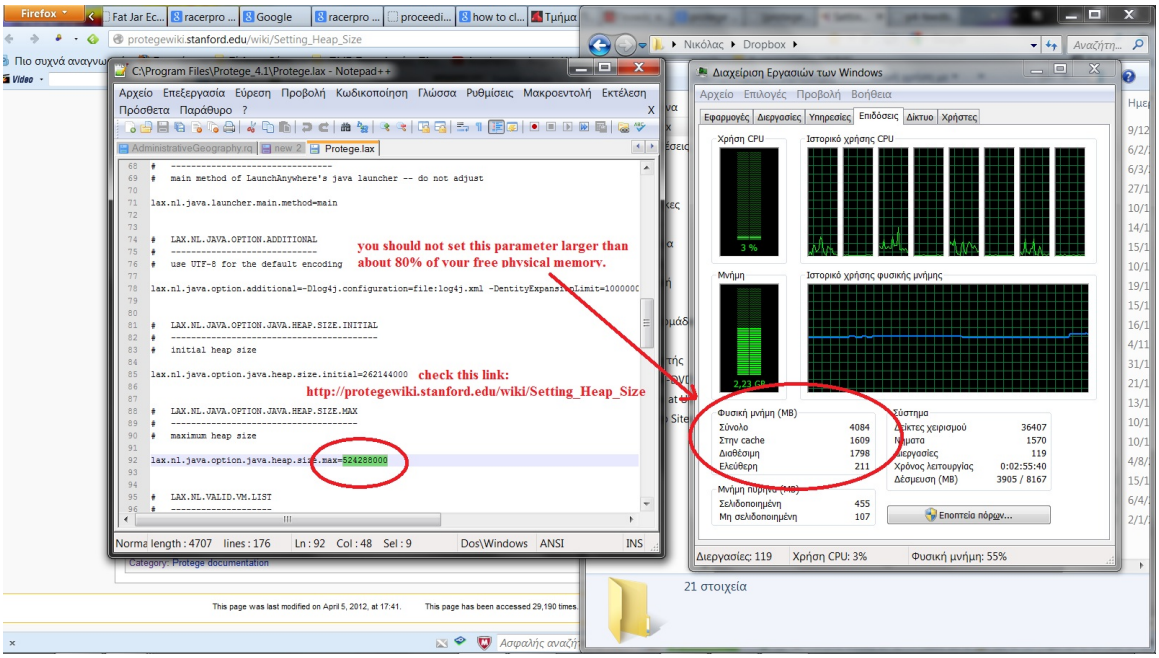

Fig. 4.10: *Setting Protégé's Heap Size*

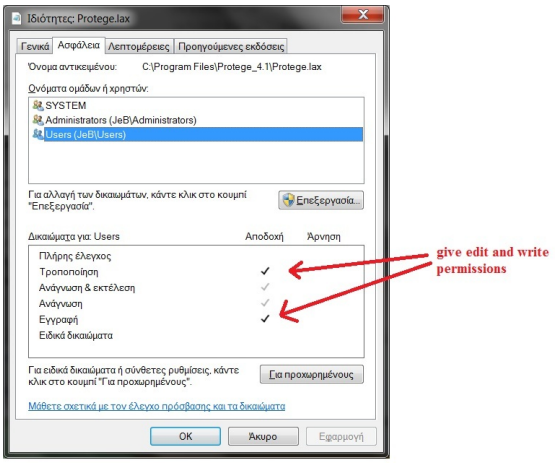

Fig. 4.11: *Setting Protege.lax permissions*

4.2.4.2 ProtegePelletSpatial Tab Querying Layout and Functionality

The Querying group of boxes in ProtegePelletSpatial's Tab Layout contains two boxes. The top of them is editable and has been initialized with an example sparql query while opposed to this the bottom box is uneditable and is initialized as vacant. As easily one can imply from these in the top box each time we ask the sparql query we want to make and the a-

fter pressing the *"Refresh"* button the bottom box print to screen the results educed from the Query Engine of Pellet Spatial Reasoner. The important advantage of this ProtegePelletSpatial Tab's functionality is that one does not have to load the ontology one time for each sparwl query, but to press only the *"Refresh"* button and the query will be performed on the spot. Unfortunately, the query editor we have performed has no error checking to prevent syntax errors, but it is planned in a latter version to be incorporated one parser for handling syntax errors. What the more, an example answer to the query is also shown in Fig.4.9. Finally, not to forget to mention that we have also included a video demo tutorial where the usage of ProtegePelletSpatial Tab is presented and some sparql queries are asked on the fly while capturing. This video tutorial is also included in the implementation material of the accompanying media (CD) of this thesis, which is deposited to the library.

4.3 Conclusions

To conclude with, in this chapter we elaborated our case study of the "Administrative Geography Ontology", and also used the example spatial ontology "USAMappings" given with pellet spatial, so that to find out at first that our reasoner plugin educes the correct result spatial relations, and also to corroborate the time complexity of the path-consistency algorithm.

In the end, our induction from the evaluation presented, and taking under consideration other studies[6] [8], [77], [76], [72], is that we can infer that PelletSpatial is not as efficient as RacerPro for classification especially because of his hybridity retardation and the use of a high time-complexity algorithm in big ontologies like "Administrative Geography", opposite to the reasoning rules of RacerPro· but still has an acceptable performance. The difference in performance seen when they both infer on non-spatial data might not be significant for applications because the processing times are generally couple of seconds, but when having to infer in spatial data it is more clear. On the other hand, PelletSpatial answers rapidly in mean time a SPARQL query, while RacerPro lingers.

[6] http://www.mindswap.org/2003/pellet/performance.shtml

CHAPTER 5

Epilogue

ProtegePelletSpatial, is a qualitative spatial reasoning engine plugin for Protégé implemented in Java, using the PelletSpatial extension of the Pellet reasoner. ProtegePelletSpatial plugin provides consistency checking and query answering over spatial data represented with the Region Connection Calculus (RCC-8) represented in RDF/OWL. As such, it can answer mixed SPARQL queries over spatial and non-spatial relation types. ProtegePelletSpatial extends PelletSpatial's hybrid architecture implementing the Protégé reasoner interface and classifies the given data using a path-consistency algorithm.

As a case study we presented the "Administrative Geography Ontology". Time complexity of O(n^3) worst case was verified and other investigations uppon PelletSpatial's and RacerPro's performance comparison were confirmed.

5.1 Conclusion and Future Work

In conclusion, we would like to suggest a number of directions for future work. Firstly, there exist other tools that in reasoning, they use SOWL, adding except for the RCC-8 calculus, the CSD-9 calculi as well. Such an addition in PelletSpatial reasoning engine would be very promising for a lot of ambiguities. To explain that, when for example some regions partially overlap, then there may exist some "holes" within their overlapping which cannot be calculated from RCC-8 and may result in false reasoning. Such cases will be solved. Furthermore, most tools support disjunctive relations between two regions to be asserted and a maximal tractable set of disjunctive relations has been shown to ensure the soundness and the completeness of the path-consistency algorithm. Scalability issues for large scale applications

are also important issues for future research. Even, OWL 2 restrictions on spatial relations (e.g.,a country A borders with exactly 3 other countries) could be a good idea to consider in reasoning. Finally, we support the suggestion of George Christodoulou, Euripides G.M. Petrakis, Sotirios Batsakis [78] in reducing the 9-CSD relations to 8 by encoding the identical to relation using the sameAs OWL axiom, as in their experimental results, it was shown that this optimization speed-up the reasoner by at least 10%.

TERMINOLOGY

Foreign Term	Greek Term
Plugin	Πρόσθετο
Semantic Web Framework	Πλαίσιο Σημασιολογικού Ιστού
Region Connection Calculus	Λογισμός Περιοχής Σύνδεσης
Ontology	Οντολογία
Reasoning	Συμπερασμός
Querying	Επερώτηση
Spatial	Χωρικός

ABBREVIATIONS - ACRONYMS

Abbreviations/Acronyms	Full Evolvent
OWL-DL	Web Ontology Language - Description Logics
OWL	Web Ontology Language
DL	Description Logics
RDF	Resource Description Framework
RDFS	Resource Description Framework Scheme
SWRL	Semantic Web Rule Language
SPARQL	SPARQL Protocol And RDF Query Language
SOWL	Spatio-temporal information in OWL
RCC	Region Connection Calculus
RCC-8	Region Connection Calculus using 8 relations
CSD	Cone-shaped Directional Calculus
CSD-9	Cone-shaped Directional Calculus using 9 relations
Jena	Jena Semantic Web Framework
OWLAPI	OWLAPI Semantic Web Framework
Protégé	Protégé Semantic Web Framework
ARQ	a Jena query engine that supports the SPARQL
KB	Knowledge Base
CN	Constraint Network
CT	Composition Table
URI	Uniform Resource Identifier
URL	Uniform Resource Locator
RACER	Renamed ABox and Concept Expression Reasoner

Continued on next page

Continued from previous page

Abbreviations/Acronyms	**Full Evolvent**
EC	Externaly Connented
EQ	Equals
DC	Disconnented
PO	Partially Overlap
TPP	Tangential Proper Part of
NTPP	Non-Tangential Proper Part of
TPPi	has Tangential Proper Part
NTPPi	has Non-Tangential Proper Part
N	North
NE	North-east
E	East
SE	South-east
S	South
SW	South-west
W	West
NW	North-west
O	Identical
MRBs	Minimum Bounding Rectangles
WWW	World Wide Web
HTML	Hyper-Text MarkUp Language
W3C	World Wide Web Consortium
BBC	British Broadcasting Corporation
SQL	Structured Query Language
RQL	RDF Query Language
RDQL	RDF Data Query Language
SeRQL	SWI-Prolog Semantic Web Server
TRIPLE	RDF Rule Language with Context and Use Cases
nSPARQL	Navigational SPARQL Protocol And RDF Query Language
DAML+OIL	DARPA Agent Markup Language + Ontology Inference Layer
JEPD	Jointly Exhaustive and Pairwise Disjoint

Continued on next page

Continued from previous page

Abbreviations/Acronyms	**Full Evolvent**
$\mathcal{ALC}$	Attribute Language with Complex concept nagation
$\mathcal{ALCQHIR}+$ / $\mathcal{SHIQ}$	Attribute Language with Complex concept nagation, Qualified cardinality restrictions, role Hierarchy, Inverse properties, Reflexity and irreflexity and more
GSIs	General Concept Inclusions
CLIPS	C Language Integrated Production System
UML	Unified Modeling Language
XMI	XML Metadata Interchange
XML	EXtensive Markup Language
PAL	Protégé Axiom Language
LDAP	Lightweight Directory Access Protocol
HTTP	Hypertext Transfer Protocol
JDK	Java Development Kit

BIBLIOGRAPHY

BIBLIOGRAPHY

[1] M. Stocker and E. Sirin, *"PelletSpatial: A Hybrid RCC-8 and RDF/OWL Reasoning and Query Engine." in 6th Intern. Workshop on OWL: Experiences and Directions (OWLED 2009)*. Springer-Verlag New York, Inc., Aug. 2009, pp. 2-31. [Online]. Available: http://www.webont.org/owled/2009/papers/owled2009_submission_20.pdf conclusions3

[2] R. H. Güting, *"An Introduction to Spatial Database Systems."* The VLDB Journal, vol. 3, no. 4, pp. 357-399, Oct. 1994. [Online]. Available: http://dl.acm.org/citation.cfm?id=615204.615206

[3] 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. 551-575, 2006. [Online]. Available: http://dblp.uni-trier.de/db/journals/tgis/tgis10.html#ArpinarSRUAK06

[4] D. A. Randell, Z. Cui, and A. Cohn, *"A Spatial Logic Based on Regions and Connection"*, in KR'92. 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. [Online]. Available: http://citeseer.ist.psu.edu/randell92spatial.html

[5] 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. [Online]. Available: http://dblp.uni-trier.de/db/reference/spatial/spatial2007.html#RenzN07

[6] 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. [Online]. Available: http://dblp.uni-trier.de/db/journals/fuin/fuin46.html#CohnH01

[7] B. Nebel and H.-J. Brckert, *''Reasoning about Temporal Relations: A Maximal Tractable Subclass of Allen''s Interval Algebra"*. in AAAI, B. Hayes-Roth and R. E. Korf, Eds. AAAI Press / The MIT Press, 1994, pp. 356-361. [Online]. Available: http://dblp.uni-trier.de/db/conf/aaai/aaai94-1.html#NebelB94

[8] Evren Sirin, Bijan Parsia, Bernardo Cuenca Grau, Aditya Kalyanpur, Yarden Katz. *"Pellet: A Practical OWL DL Reasoner"*. In Proc. of Int. Workshop on Description Logics (DL2004), 2004. Department of Computer Science Georgia State University Atlanta,Georgia 30303. http://tinman.cs.gsu.edu/řaj/8711/sp11/presentations/pelletReport.pdf, http://pellet.owldl.com/papers/sirin05pellet.pdf

[9] A. Mackworth.*"Consistency in Networks of Relations"*. Artificial Intelligence, 8(1):99-118, 1977.

[10] C. Welty and R. Fikes: *"A Reusable Ontology for Fluents in OWL"*. Frontiers in Artificial Intelligence and Applications, 150:226236, 2006.

[11] S. Batsakis, *"SOWL: A Framework for Handling Spatio-Temporal Information in OWL,"* Ph.D. dissertation, Technical Univ. of Crete(TUC), Dept. of Electronics and Comp. Engineering, Chania, Crete, Greece, Dec. 2011. [Online]. Available: http://www.intelligence.tuc.gr/publications.php?pub author=All&pub type=10&pub subject=All

[12] E. G. M. Petrakis, *"Design and Evaluation of Spatial Similarity Approaches for Image Retrieval."* Image Vision Comput., vol. 20, no. 1, pp. 59-76, 2002. [Online]. Available: http://dblp.uni-trier.de/db/journals/ivc/ivc20.html#Petrakis02

[13] 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:4366, 1995.

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

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

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

[17] T. Berners-Lee, J. Hendler, O. Lassila, et al. *"The Semantic Web"*. Scientific American, 284[5]:2837, 2001. 10

[18] F. Baader. *"The Description Logic Handbook: Theory, Implementation, and Applications*. Cambridge Univ. Pr., 2003. 10

[19] D.L. McGuinness, F. Van Harmelen, et al. *"OWL Web Ontology Language Overview"*. W3C Recommendation, 10:200403, 2004. 10, 16

[20] I. Horrocks, P.F. Patel-Schneider, H. Boley, S. Tabet, B. Grosof, and M. Dean. *"SWRL: A Semantic Web Rule Language Combining OWL and RuleML"*. W3C Member submission, 21, 2004. 10

[21] Ora Lassila and Ralph R. Swick. *"Resource Description Framework (RDF) Model and Syntax Specification"*. Technical report, W3C Recommendation, 1999.

[22] Georgi Kobilarov, Tom Scott, Yves Raimond, Silver Oliver, Chris Sizemore, Michael Smethurst, Christian Bizer, and Robert Lee. *"Media Meets Semantic Web How the BBC Uses DBpedia and Linked Data to Make Connections"*. In Proceedings of the 6th European Semantic Web Conference (ESWC), Heraklion, Crete, Greece, 2009.

[23] Tim BernersLee, Roy Fielding, and Larry Masinter. RFC 2396 *"Uniform Resource Identifiers (URI): Generic Syntax"*. http://www.isi.edu/innotes/rfc2396.txt, Augoust 1998.

[24] Greg Karvounarakis, Sofia Alexaki, Vassilis Christophides, Dimitris Plexousakis, and Michel Scholl. *"RQL: A Declarative Query Language for RDF"*. In Proceedings of the 11th World Wide Web Conference (WWW 2002), Honolulu, Hawaii, U.S.A., 2002

[25] Andy Seaborne. *"RDQL A query Language for RDF"*, W3C Member Submission 9 January 2004. http://www.w3.org/Submission/2004/SUBMRDQL20040109/, Jan 2004.

[26] Jeen Broekstra and Arjohn Kampman. *"SeRQL: An RDF Query and Transformation Language"*. http://www.cs.vu.nl/ jbroeks/papers/SeRQL.pdf, 2004.

[27] Michael Sintek and Stefan Decker. *"TRIPLE - A Query, Inference and Transformation Language for the Semantic Web"*. In Proceedings of Deductive Databases and Knowledge Management (DDLP'2001), 2001.

[28] Eric Prud'hommeaux and Andy Seaborn. *"SPARQL Query Language for RDF"*. W3C Recommendation http://www.w3.org/TR/rdf-sparql-query/, 2008.

[29] Jorge Perez, Marcelo Arenas, and Claudio Gutierrez. nSPARQL: *"A Navigational Language for RDF"*. In Proceedings of the 7th International Conference on The Semantic Web (ISWC 2008), Karlsruhe, Germany, 2008.

[30] Marcelo Arenas, Claudio Gutierrez, Jorge Perez, and Catolica Chile. *"An Extension of SPARQL for RDFS"*. In Proceedings of Semantic Web, Ontologies and Databases, VLDB Workshop (SWDBODBIS 2007), Vienna, Austria, 2007.

[31] Bechhofer, S., Harmelen, F.V., Hendler, J., Horrocks, I., McGuinness, D.L., Patel-Schneider P.F., et al.: *"OWL Web Ontology Language"*. Retrieved November 27, from W3C Recommendation: http://www.w3.org/TR/owl-ref/ (2009)

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

[33] Manolis Koubarakis, *"Lectures of Lessons in Artificial Intelligence"*, NKUA, 2012, http://cgi.di.uoa.gr/ artint2/STORAGE/lectures/2011-2012/introduction-to-owl2.pdf

[34] B.C. Grau, I. Horrocks, B. Motik, B. Parsia, P. Patel Schneider, and U. Sattler. OWL 2: *"The Next Step for OWL. Web Semantics: Science, Services and Agents on the World Wide Web"*, 6[4]:309-322, 2008. 17

[35] B. Motik, P.F. Patel-Schneider, B. Parsia, C. Bock, A. Fokoue, P. Haase, R. Hoekstra, I. Horrocks, A. Ruttenberg, U. Sattler, et al. *"OWL 2 Web Ontology Language: Structural Specification and Functional-Style Syntax"*. W3C Recommendation, 27, 2009. 17

[36] I. Horrocks, O. Kutz, and U. Sattler. *"The Even More Irresistible SROIQ"*. In Proc. of the 10th Int. Conf. on Principles of Knowledge Representation and Reasoning (KR 2006), pages 5767, 2006. 17, 56

[37] J. Renz *"Maximal tractable fragments of the Region Connection Calculus: A complete analysis."* Int. Joint Conference on Artificial Intelligence, vol.16, pp. 448-455, 1999.

[38] P. van Beek, and R. Cohen *"Exact and approximate reasoning about temporal relations."* Computational intelligence, Vol 6(3), pp. 132147, 1990.

[39] Bennett, B.: *"Logics for Topological Reasoning"*. In: 12th European Summer School in Logic, Language and Information (ESSLLI-MM). Birmingham, UK (2000, August, 6•18)

[40] Egenhofer, M., Franzosa, R.: *"Point-Set Topological Spatial Relations"*. International Journal of Geographical Information Systems, 5(2) (1991) 161-174

[41] Grütter, R., Bauer-Messmer, B.: *"Towards Spatial Reasoning in the Semantic Web: A Hybrid Knowledge Representation System Architecture"*. Lecture Notes in Geoinformation and Cartography. Springer, Berlin (2007) (to appear)

[42] Baader, F., Nutt, W.: Basic Description Logics. In: Baader, F., Calvanese, D., McGuinness, D., Nardi, D., Patel-Schneider, P. (eds.): *"The Description Logic Handbook"*. Cambridge University Press (2003) 47-100

[43] 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.

[44] 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. [Online]. Available: http://dblp.uni-trier.de/db/conf/pricai/pricai2004.html#RenzM04

[45] D. Peuquet and C.-X. Zhan. 1987. *"An algorithm to determine the directional relationship between arbitrarily-shaped polygons in a plane"*. Pattern Recognition 20 : 65-74.

[46] I. Horrocks, U. Sattler, *"A tableaux decision procedure for SHOIQ"*, in: Proc. of the 19th Int. Joint Conf. on Artificial Intelligence (IJCAI 2005), Morgan Kaufman, 2005.

[47] F. Baader, U. Sattler, *"An overview of tableau algorithms for description logics"*, Studia Logica 69 (2001) 540.

[48] Boris Motik, Peter F. Patel-Schneider, and Bernardo Cuenca Grau. *"SWRL: A semantic web rule language combining OWL and RuleML"*. URL, 2009. http://www.w3.org/Submission/SWRL/.

[49] Boris Motik, Ian Horrocks, and Ulrike Sattler. *"Representing Ontologies Using Description Logics, Description Graphs, and Rules"*. Artificial Intelligence, 7(2):7489, 2009.

[50] Wessel, M.: *"On Spatial Reasoning with Description Logics"*. Position Paper. In: Horrocks, I., Tessaris, S. (eds.): Proceedings of the International Workshop on Description Logics (DL2002), Toulouse, France. CEUR Workshop Proceedings 53 (2002, April 19-21) 156-163

[51] Rolf Grütter and Bettina Bauer-Messmer: *"Combining OWL with RCC for Spatioterminological Reasoning on Environmental Data"*. Swiss Federal Institute WSL, An Institute of the ETH Board, Zurcherstrasse 111, 8903 Birmensdorf, Switzerland rolf.gruetter, bettina.bauer@wsl.ch

[52] http://www.racer-systems.com/products/racerpro/index.phtml

[53] Michael Wessel, Ralf Möller Hamburg University of Technology (TUHH), Germany-mi.wessel, r.f.moeller@tuhh.de: *"A Flexible DL-based Architecture for Deductive Information Systems"*. http://ceur-ws.org/Vol-192/paper07.pdf

[54] Lora Aroyo, Darina Dicheva (Workshop co-chairs). *"International Workshop on Applications of Semantic Web technologies for E-Learning*(SW-EL), http://www.win.tue.nl/SW-EL/

[55] Enrico_Motta, Seminars, http://sisinflab.poliba.it/dinoia/bag/seminars/Enrico_Motta/Semantic_Web_Technologies_and_Applications_part2.pdf

[56] Manolis Koubarakis, *"Lectures of Lessons in Artificial Intelligence-"*, NKUA, 2012, http://cgi.di.uoa.gr/ artint2/STORAGE/lectures/2011-2012/Introduction%20to%20the%20Semantic%20Web.pdf

[57] *"Introduction to the Semantic Web Vision and Technologies"*: http://www.slideshare.net/nimonika/pelagios-2011

[58] Holger Knublauch, Ray W. Fergerson, Natalya F. Noy and Mark A. Musen. *"The Protégé OWL Plugin: An Open Development Environment for Semantic Web Applications"*. Stanford Medical Informatics, Stanford School of Medicine 251 Campus Drive, Stanford, CA 94305-5479, holger@smi.stanford.edu. http://protege.stanford.edu/plugins/owl/publications/ISWC2004-protege-owl.pdf

[59] Michael K. Smith, Chris Welty, Deborah L. McGuinness: *"OWL WebOntology Language Guide"*. W3C Recommendation, February 2004. http://www.w3.org/TR/owl-guide/

[60] http://jena.apache.org/about_jena/architecture.html

[61] Matthew Horridge a, Sean Bechhofer . *"The OWL API: A Java API for OWL Ontologies"*. School of Computer Science, Kilburn Building, Oxford Road, University of Manchester, M13 9PL, United Kingdom, email: first.second@manchester.ac.uk, http://www.semantic-web-journal.net/sites/default/files/swj107_2.pdf

[62] W3C DOMWorking Group. *"Document Object Model"*. http://www.w3.org/DOM/.

[63] Sun Microsystems, Inc. JavaTMPlatform. http://java.sun.com/j2se/.

[64] Graham Klyne and Jeremy J. Carroll. *"Resource Description Framework (RDF): Concepts and Abstract Syntax"*. W3C Recommendation, World Wide Web Consortium, 2004. http://www.w3.org/TR/owl-guide/.

[65] Motik, Boris and Patel-Schneider, Peter F. and Parsia, Bijan. OWL 2 Web Ontology Language Structural Specification and Functional-Style Syntax. W3C Recommendation, World Wide Web Consortium, 2009. http://www.w3.org/TR/owl2-syntax/.

[66] Jeremy J. Carroll, Ian Dickinson, Chris Dollin, Dave Reynolds, Andy Seaborne, and Kevin Wilkinson. *"Jena: implementing the semantic web recommendations"*. In Stuart Feldman, Mike Uretsky, Mark Najork, and Craig Wills, editors, Proceedings of the 13th international World Wide Web conference on Alternate track papers & posters, pages 74-83, New York, NY, USA, May 2004. ACM.

[67] Matthew Horridge. *"A Practical Guide To Building OWL Ontologies Using Protégé 4 and CO-ODE Tools Edition 1.3"*. http://owl.cs.manchester.ac.uk/tutorials/protegeowltutorial/resources/ProtegeOWLTutorialP4_v1_3.pdf

[68] Y. Katz, and B. Grau. *"Representing Qualitative Spatial Information in OWL-DL"*. In Proc. of Int.Workshop: OWL Experiences and Directions, Galway, Ireland, 2005.

[69] Manolis Koubarakis, Kostis Kyzirakos and Charalampos Nikolaou. *"Data Models and Query Languages for Linked Geospatial Data[RW2012]"*. Dept. of Informatics and Telecommunications National and Kapodistrian University of Athens. http://www.strabon.di.uoa.gr/files/Session1-Introduction.pdf

[70] Charalampos Nikolaou. *"Geospatial Information with Description Logics, OWL, and Rules[RW2012]"*. Dept. of Informatics and Telecommunications National and Kapodistrian University of Athens. http://www.kr.tuwien.ac.at/events/rw2012/teaching-material/RW12-Tutorial-Koubarakis-III.pdf

[71] Kostis Kyzirakos. *"Implemented RDF Stores with Geospatial Support[RW2012]"*. Dept. of Informatics and Telecommunications National and Kapodistrian University of Athens. http://www.strabon.di.uoa.gr/files/Session5-Implemented_systems.pdf

[72] Manolis Koubarakis. *"Conclusions[RW2012]"*. Dept. of Informatics and Telecommunications National and Kapodistrian University of Athens. http://www.strabon.di.uoa.gr/files/Session7-Conclusions.pdf

[73] Kostis Kyzirakos. *"Geospatial data in RDF - GeoSPARQL[RW2012]"*. Dept. of Informatics and Telecommunications National and Kapodistrian University of Athens. http://www.strabon.di.uoa.gr/files/Session4-GeoSPARQL.pdf

[74] Kostis Kyzirakos. *"Geospatial data in RDF - stSPARQL[RW2012]"*. Dept. of Informatics and Telecommunications National and Kapodistrian University of Athens. http://www.strabon.di.uoa.gr/files/Session3-stSPARQL.pdf

[75] Manolis Koubarakis, Manos Karpathiotakis, Konstantina Bereta and Consortium members. *"TELEIOS"*. http://www.earthobservatory.eu/deliverables/FP7-257662-TELEIOS-D10.4.2.pdf

[76] Michael Wessel, Ralf Möller Hamburg University of Technology (TUHH), Germany-mi.wessel, r.f.moeller@tuhh.de: *"Flexible Software Architectures for Ontology-Based Information Systems"*. http://www.michael-wessel.info/papers/jal.pdf

[77] Volker Haarslev, Kay Hidde, Ralf MÏler and Michael Wessel, *"The RacerPro Knowledge Representation and Reasoning System"*, http://www.semantic-web-journal.net/sites/default/files/swj109_3.pdf

[78] George Christodoulou, Euripides G.M. Petrakis, Sotirios Batsakis, *"Qualitative Spatial Reasoning using Topological and Directional Information in OWL"*. Dept. of Electronic and Computer Engineering Technical University of Crete (TUC) Chania, Crete, Greece, GR-73100. Email: {fchristodoulou, petrakis, batsakisg}@intelligence.tuc.gr

[79] Charalampos Nikolaou, Manolis Koubarakis: *"Incomplete Information in RDF"*. CoRR abs/1209.3756 (2012)

[80] Zoi Kaoudi, Manolis Koubarakis, Kostis Kyzirakos, Iris Miliaraki, Matoula Magiridou, Antonios Papadakis-Pesaresi: *"Atlas: Storing, updating and querying RDF(S) data on top of DHTs"*. J. Web Sem. 8(4): 271-277 (2010)

[81] Alex Lohfink, Duncan McPheeQ: *"Resource-Level Versioning in Administrative Geography RDF Data"*. University of Glamorgan, faculty of Advanced Technology, Dept. of Computing and Maths, Pontypridd, CF37 1DL, Tel. +44443 482950, (alohfink, dmcphee)@glam.ac.uk, http://www.semantic-web-journal.net/sites/default/files/swj214.pdf

[82] John Goodwin, Catherine Dolbear, Glen Hart: *"Geographical Linked Data: The Administrative Geography of Great Britain on the Semantic Web"*, research article, http://onlinelibrary.wiley.com/doi/10.1111/j.1467-9671.2008.01133.x/abstract

[83] Jochen Renz and Bernhard Nebel: *Efficient Methods for Qualitative Spatial Reasoning"*, J. Artif. Intell. Res.(JAIR), 2001, pp.289-318, http://dx.doi.org/10.1613/jair.872, http://www.dblp.org/rec/bibtex/journals/jair/RenzN01