PelletSpatial: A Hybrid RCC-8 and RDF/OWL Reasoning and Query Engine

Markus Stocker[1] and Evren Sirin[2]

[1] University of Kuopio, Kuopio, Finland
markus.stocker@uku.fi
[2] Clark & Parsia, Washington, DC, USA
evren@clarkparsia.com

Abstract. In this paper, we present PelletSpatial, a qualitative spatial reasoning engine implemented on top of Pellet. PelletSpatial provides consistency checking and query answering over spatial data represented with the Region Connection Calculus (RCC). It supports all RCC-8 relations as well as standard RDF/OWL semantic relations, both represented in RDF/OWL. As such, it can answer mixed SPARQL queries over both relation types. PelletSpatial 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. We discuss the details of two implementation approaches and focus on some of their respective advantages and disadvantages.

1 Introduction

It is a common practice use Web Ontology Language (OWL) ontologies to describe spatial regions and relations between these regions such as 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.

In this paper, we present PelletSpatial which supports qualitative spatial reasoning based on the Region Connection Calculus (RCC) logic formalism. PelletSpatial, extends the DL reasoning features of Pellet [10] 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 relations to OWL-DL class axioms presented in [7]. To the best of our knowledge, there has not been an implementation for this translation. In this paper, we present our experimental evaluation of this approach and explain why there are major performance issues. Motivated by these poor performance results, we investigate an alternative spatial reasoning technique based on a path-consistency algorithm and the RCC-8 composition table.

2 Related Work

The Region Connection Calculus (RCC), in the version of the theory we use, was introduced by Randell *et al.* in [11]. 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.

In [5], the authors give a succinct review of spatio-temporal reasoning in Description Logic (DL), introduced in [6, 16]. In [4], they present an architecture for combining RCC and the Web Ontology Language (OWL) by extending the DL Knowledge Base (KB) with RCC specific components, namely a *RCCBox*, a RCC reasoner that uses composition tables for RCC-1, RCC-2, RCC-3, RCC-5 and RCC-8 to check spatial consistency of the ABox, i.e. the consistency of spatial role assertions. In PelletSpatial, we strictly separate spatial from non-spatial relations and manage (defined as well as disjunctive) spatial relations in what we call a RCC constraint network which provides functionality for consistency checking and query answering.

In [7], the authors show how RCC-8 relations can be translated to OWL-DL based on the correspondence between RCC and Modal Logics [9]. In particular, each base RCC-8 relation is translated to a set of class axioms and every RCC region is mapped to a DL concept such that the region follows the regularity condition, which consists of two requirements: (1) to be a non-empty concept and (2) to contain all of the region's interior points. As the authors show, the two requirements can be described in OWL-DL. PelletSpatial includes an implementation of the this translation for which our experiments showed serious performance drawbacks.

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 [12], the authors provide a reference implementation for a path-consistency algorithm applied to consistency checking for RCC.

More recently, Gantner *et al.* proposed a Generic Qualitative Reasoner (GQR) [3] a reasoning service for arbitrary binary qualitative calculi developed for spatial and temporal reasoning based on Mackworth's variant of the path-consistency algorithm [8], backtracking search and an efficient queue data structure [14].

In [17], the authors test the assumption that algorithms tailored for qualitative calculi, generically, and RCC, specifically, computationally outperform reasoning in more expressive formalisms. To support their statement, the authors compare GQR [3] with, among others, the DL reasoner FaCT++ [13]. As we will argue in this paper, with our implementation experience we arrive at similar conclusions.

The authors of [15] propose with *SparQ* a reference implementation and open platform for the development of spatial qualitative calculi for Qualitative Spatial Reasoning (QSR). The authors argue that even though the development of QSR algorithms and techniques has been an active area for decades, practical applications are not very widespread.

3 Reasoning Engine Architecture

The main goal of PelletSpatial is the support for reasoning and querying for both spatial RCC-8 relations and standard RDF semantic relations. To achieve this goal, we represent every region as an OWL individual. We define an OWL object property for each RCC-8 relation and a spatial relation between two regions is represented as an OWL object property assertion. Non-spatial relations, such as region type, size, etc., are represented as ordinary OWL assertions. For reasoning, PelletSpatial extends Pellet's standard reasoning capabilities to take the semantics of RCC relations into consideration. In the next two subsections we describe two different implementation approaches to support this extension.

3.1 The OWL-DL RCC-8 Reasoning Engine

Due to the close relationship between DLs and Modal Logics as well as between Modal Logics and the RCC-8 formalism, it is possible to create translate RCC-8 relations to DL axioms as described in [7]. First, for each RCC region R we define a corresponding OWL class C_R. Then, an object property assertion like $DC(X, Y)$ stating regions X and Y are disconnected, is translated to an OWL disjointness axiom as in `DisjointClasses`$(C_X\ C_Y)$. We refer the reader to [7] for the details of the translation.

In addition to translation RCC relations to OWL class axioms, one axiom is defined for each region to satisfy the regularity condition of region. This axiom significantly affects non-determinism as well as the number of qualified existential quantifiers in the ontology. This, as we experienced with our implementation, significantly affects the system's performance to the point that, without further investigations and optimizations, the system lacks practicability even for only a few regions.

Given this semantics preserving translation, consistency checking of a set of spatial relations can be reduced to the problem of consistency checking in OWL. Except for the translation itself , Pellet already supports what is required for such a system.

We would like to point out two features of OWL 2 that was crucial for the translation to work. First, is the addition of reflexive properties[3] which is required by the translation algorithm [7]. Second, we use the punning feature of OWL 2[4] so that we can use the same URI to identify both the original OWL individual representing the region and the OWL class generated by the translation simplifying our implementation.

3.2 The Hybrid RCC-8 Reasoning Engine

Because of the performance problems experienced with the implementation described above, we investigated an alternative reasoning technique for RCC-8 constraint networks.

The hybrid implementation in PelletSpatial strictly separates spatial reasoning and semantic OWL-DL reasoning by using a specialized RCC reasoner. Spatial relations are managed as an RCC constraint network that provides – similar to a KB – functionality to check its consistency and querying. Non-spatial relations, i.e. standard OWL assertions, are managed as a Pellet KB as usual.

Consistency checking of a RCC constraint network in our hybrid implementation is performed by means of a path-consistency algorithm based on the RCC-8 composition table. Currently, PelletSpatial supports path-consistency based reasoning for the eight base RCC-8 relations. The details of this algorithm is described in the next section.

4 Path-Consistency Algorithm

In this section, we describe the algorithm used in PelletSpatial for the hybrid architecture (Section 3.2) to check the consistency of a RCC-8 constraint network with defined RCC-8 relations. We compare our implementation with the algorithm provided by the authors in [12] and discuss the differences.

In [12], the authors describe the standard path-consistency algorithm implementation for RCC-8 consistency checking. It is based upon the $n \times n$ matrix M that represents the spatial relationships between n different regions. Here, consistency checking is a process that iteratively performs the path-consistency operation $M_{ij} \leftarrow M_{ij} \cap M_{ik} \circ M_{kj}$ for all regions i, j, k until a fixed point is reached or $M_{ij} = \emptyset$, in which case M is inconsistent.

We implemented a similar algorithm in PelletSpatial using slightly different data structures and queing strategies. Algorithm 1 describes the details of our implemenation. Given a RCC-8 constraint network N, i.e. a set of defined RCC-8 relations, N is consistent if it is empty or if every relation in the network is consistent. Note that, the requirement for the relations in N to be defined, i.e. of the set of eight base relations, is relevant to the tractability of a sound and complete path-consistency procedure. As it is argued in [12], sound and complete path-consistency is tractable for the set of eight defined relations. Maximal

[3] http://www.w3.org/TR/owl2-new-features/#F6:_Reflexive
[4] http://www.w3.org/TR/owl2-new-features/#F12:_Punning

tractable subsets may contain more of the 256 RCC-8 relations but currently we limit our implementation to defined relations as we have not investigated extensions.

The *complete* step at Line 5 processes N to complete inverse and equals relations. For every (defined) relation $R_{ij} \in N$, we ensure that $R^{\smile}_{ji} \in N$ (inverse complete), e.g. for $\{TPP\}(a,b)$ we ensure that $\{TPPi\}(b,a) \in N$, a,b regions $\in N$. For every region $a \in N$, we ensure that $\{EQ\}(a,a) \in N$ (equals complete). Note that, in the notation used for this paper, R_{ij} corresponds to $\{R_1,\ldots,R_n\}(i,j)$, being R the disjunctive set of several relations $\{R_1,\ldots,R_n\}$. According to the notation used in [2], a defined relation $D(i,j)$ can also be written as $\{D\}(i,j)$, i.e. R_{ij}, $R=\{D\}$.

We use a queue Q as a structure to keep track of relations that have to be processed. Hence, the algorithm runs until $Q=\emptyset$ or we found an inconsistency. Q is initialized with all relations $R_{ij} \in N$ (which, currently, are only defined relations).

A relation R_{ab} (Line 15) is path-consistent if the rule for combining a compositional inference with existing information [2],

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

results in a non-empty set $V \neq \emptyset$ for regions a,c; $S_{bc} \in N$ relations with a transitive path with R_{ab} from a through b to c and U_{ac} a relation (possibly $\in N$ as existing information). The compositional inference $T_{ac} \leftarrow R_{ab} \circ S_{bc}$ (Line 17) is computed for regions a,c as the union set T for the composition of each pair (r,s) in the set $R \times S$, $r \in R$, $s \in S$. The composition of a pair (r,s) consists in a lookup for the RCC-8 composition table given that r and s are elements of the set of eight defined relations.

If $U_{ac} \in N$, i.e. there is existing information for the pair (a,c), we complete the rule by computing the intersection $V_{ac} \leftarrow T_{ac} \cap U_{ac}$ (Line 33), where V is the intersection set of relations $v \in T \cap U$. This step does refine the already existing relation $U_{ac} \in N$ and is essential for the path-consistency algorithm as it defines the inconsistent state: if $V=\emptyset$ we have found an inconsistency.

The state at Line 38 is also worth a note. If $U=V$, it means that the step at Line 33 could not refine relation U_{ac}. Hence, combining compositional inference T_{ac} with existing information U_{ac} does not add new information. In this case, we can return. Else, we remove U_{ac} from N, add the refined V_{ac} to N and Q and process the inverse $V^{\smile}_{ca}$.

Our path-consistency implementation (Algorithm 1) presents several differences to the algorithm described in [12]. The first is in the structure used to represent a finite set of RCC-8 constraints, input for the path-consistency algorithm. The algorithm in [12] uses a $n \times n$ *dense* matrix M for n different regions, where M_{ij} represents the relation between the regions i,j. (Note that, there is always at least the universal relation $\top$ between regions i,j.) Instead, our implementation processes a *sparse* matrix with empty cells for $M_{ij}=\top$.

Consequently, the queue Q is initialized differently. While in our implementation Q corresponds to the array of elements $M_{ij} \neq \top$, in [12] Q is an array

Algorithm 1 PathConsistency

```
procedure PathConsistency($N$)
    if $N = \emptyset$ then
        return true
    end if
    $complete(N)$
    $Q \leftarrow \{R_{ij} | R_{ij} \in N\}$
    while $Q \neq \emptyset$ do
        $R_{ab} \leftarrow remove(Q)$
        if $!isConsistent(N, Q, R_{ab})$ then
            return false
        end if
    end while
    return true
end procedure

procedure IsConsistent($N, Q, R_{ab}$)
    for $S_{bc} \in N$ do
        $T_{ac} \leftarrow R_{ab} \circ S_{bc}$
        $add(N, Q, T_{ac})$
        if $!isConsistent$ then
            return false
        end if
    end for
    return true
end procedure

procedure Add($N, Q, T_{ac}$)
    if $T = \top$ then
        return
    end if
    $U_{ac} \leftarrow \{R_{ij} | i = a, j = c, R_{ij} \in N\}$
    if $\nexists U_{ac}$ then
        $V_{ac} \leftarrow T_{ac}$
    else
        $V_{ac} \leftarrow T_{ac} \cap U_{ac}$
        if $V = \emptyset$ then
            $isConsistent = false$
            return
        end if
        if $U = V$ then
            return
        end if
        $N \leftarrow N \setminus \{U_{ac}\}$
    end if
    $N \leftarrow N \cup \{V_{ac}\}$
    $Q \leftarrow Q \cup \{V_{ac}\}$
    $add(N, Q, V_{\breve{ca}})$
end procedure
```

of triple pairs (i, j, k), (k, i, j) for regions that represent a path that needs to be revised. Note that, an element M_{ij} describes an edge M between two nodes i, j while a triple (i, j, k) represents two edges M_{ij} and M_{jk} between the nodes i, j and j, k, holding, thus, more information.

As we process a sparse matrix (ignoring elements $M_{jk} = \top$) in our algorithm we iterate over fewer k nodes. Further, in our Q we don't keep track of triples (k, i, j) which correspond to a path with incoming edge for i. Given an edge i, j, we look for nodes k and update i, k (which corresponds to triple (i, j, k)). In our implementation, incoming edges for i are processed by keeping both i, j and j, i in Q. On revising j, i, we look for nodes k (assuming edges k, i and i, k are in Q) and update j, k as well as k, j (which corresponds to triple (k, i, j)).

The triple structure allows to be more compact in the rule for combining compositional inference and existing information, as the triple already contains all the required information to compute the rule. The algorithm in [12], thus, computes the rule in one step while in our implementation we split the compositional inference from its combination with existing information. This allows us to discard the compositional inference whenever it returns the universal relation $\top$ from further processing as it cannot add any new information to the network.

The two implementations also differ in how they process inverses $M_{ji}^{\smile}$. The algorithm in [12] assigns the inverse directly to the corresponding matrix index, not adding it, thus, to Q. Instead, in our implementation, we recurse for $M_{ji}^{\smile}$.

5 Answering Spatial Queries

PelletSpatial supports a subset of SPARQL queries with Basic Graph Patterns (BGP) that include spatial and non-spatial patterns, i.e. triple patterns for spatial relations (joined) with triple patterns for semantic RDF relations. This type of querying is supported for both reasoning architectures, i.e. SPARQL querying is independent of the underlying reasoning engine. In the following, we refer to a query that may have both spatial and non-spatial query patterns simply with *spatial query*.

As PelletSpatial processes both spatial and standard semantic OWL relations in RDF/OWL documents, it is natural to support spatial querying. (Note that a subset of the semantic OWL relations may be metadata about the regions of spatial relations.) It is, thus, possible to query for regions that are involved in a specific spatial relation with another region and have certain characteristics that are described by semantic RDF relations. For instance, we could query a hypothetical ontology for all region names that are externally connected to a given region (the spatial subset query) that have an area and population greater than a given lower bound (the non-spatial subset query).

The algorithm used in PelletSpatial to answer spatial queries depends on the underlying architecture. For the architecture in which we translate defined RCC-8 relations to OWL-DL class axioms (Section 3.1) spatial and semantic RDF relations are stored in a single Pellet KB (RDF graph). Hence, spatial queries can be translated to SPARQL-DL queries that are answered by Pellet.

There is an important issue to note with respect to this query answering algorithm. Some of the RCC-8 relations are translated to axioms with universal role restrictions. Pellet does not support SPARQL-DL query atoms with variable filler for role restrictions. However, it is possible, in spatial queries, that a region translated to a role restriction filler may be variable. For instance, the query atom $EC(?x, Wisconsin)$ is translated, among others, to the axiom $\forall R.x \sqsubseteq \neg\forall R.Wisconsin$, where the left hand side has a variable filler in role restriction.

A workaround for variable fillers in role restrictions is to substitute them with regions in a reformulation step during translation of RCC query atoms to SPARQL-DL query atoms. In our example, provided a RCC-8 constraint network with regions *Iowa*, *Wisconsin*, and *Minnesota*, we would reformulate the spatial query $EC(?x, Wisconsin)$ to a set of two elements, each containing, among others, the axiom $\forall R.Iowa \sqsubseteq \neg\forall R.Wisconsin$ and $\forall R.Minnesota \sqsubseteq \neg\forall R.Wisconsin$ respectively. (Note that for EC we can avoid reformulating the variable with the region *Wisconsin*.) It is not surprising that this query atom reformulation affects query performance. In fact, depending on the number of regions, a single spatial query may be reformulated into a prohibitive set size.

The reformulation is avoided with the algorithm used for spatial query answering in the architecture based on the path-consistency algorithm (Section 3.2). Here we use a dual stage query answering technique by extending the constraint network with a dedicated query handler that, given a RCC query atom and – for conjunctive queries – a set of prepared bindings, returns a set of RCC query solutions, i.e. variable to region bindings. The set of query solutions returned by the first stage is given as input to the second stage which consists of further constraining the set of bindings such that the non-spatial query subset is satisfied.

6 Experiments

In our preliminary experiments, we have been testing both reasoner architectures with small manually created datasets containing only spatial relations or combining spatial and non-spatial relations.

Our results show that, without further optimizations, the reasoner based on the translation of RCC relations to OWL-DL class axioms (Section 3.1) lacks practicability even for our small datasets. The axiom that is required for every region to follow regularity condition significantly affects the performance of PelletSpatial. This is caused by the interplay of qualified existential and universal quantifiers, one of the source of complexity (AND-branching) in DL reasoning [1]. In [7], the authors ask whether DL optimizations work well for their translation. With PelletSpatial we show that, without further optimizations, the translation lacks practicability even for just a few defined RCC-8 relations.

As we mentioned earlier in Section 5, this architecture is furthermore problematic for querying of RCC relations that need to be reformulated because of a variable in the filler position of role restrictions. Instead of investigating possible solutions to improve the performance of this architecture, we opted for another spatial reasoning technique (Section 3.2).

Our preliminary experiments show that the alternative spatial reasoner performs much better on our manually created datasets as well as on the significantly bigger Ordnance Survey[5] ontology that describes administrative areas in England, Wales and Scotland.

We have used a sample of the Ordnance Survey ontology with a mapping for the terms used in PelletSpatial for RCC-8 relations and those used by the Ordnance Survey ontology. The sample contains 223 (defined) RCC-8 regions. Consistency checking for the sample runs in roughly 15 seconds on a Lenovo X60. After consistency checking, the network contains 17,652 RCC relations, of which 7,042 are defined.

Testing with the full Ordnance Survey ontology returned a few inconsistency which have been forwarded and confirmed by Ordnance Survey. Results show that consistency checking for the corresponding network with 72,688 relations and 11,756 regions currently performs too slow for practical use.

We did plan to include some experimental results in collaboration with the Swiss Federal Institute for Forest, Snow and Landscape Research[6]. Because of time constraints we need to defer our experiments.

7 Conclusion and Future Work

With PelletSpatial, we provide a proof-of-concept for a system implemented on top of Pellet for qualitative spatial reasoning and querying for data represented with RCC and RDF/OWL. The experience has shown that the translation of RCC relations to OWL-DL class axioms presented by the authors in [7] lacks practicability even for very small networks (without further optimizations).

A spatial reasoner based on a path-consistency algorithm and the RCC-8 composition table has been more promising w.r.t reasoning and query performance. There are a number of directions for future work. As the experiments have shown, PelletSpatial does not scale to more than a few thousand relations. We are planning to investigate the impact of heuristics as described in [12] and newer approaches adopted in GQR [3] and SparQ [15] to improve the scalability of PelletSpatial.

We are also planning to extend PelletSpatial support beyond eight base relations. 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 [12]. With this extension, we can then support more general relations like "proper part of" (PP) as a disjunction of $\{TPP, NTPP\}$ both in the data and in queries.

[5] http://www.ordnancesurvey.co.uk
[6] http://www.wsl.ch

References

1. F. Baader, D. Calvanese, D. McGuinness, D. Nardi, and P. Patel-Schneider. *The Description Logic Handbook: Theory, Implementation and Applications.* Cambridge University Press, 2nd Edition, 2007.
2. B. Bennet. Knowledge Representation and Reasoning: Compositional Reasoning, 2007. Lecture notes, School of Computing, University of Leeds.
3. Z. Gantner, M. Westphal, and S. Wölfl. GQR - A Fast Reasoner for Binary Qualitative Constraint Calculi. In *Proc. of AAAI'08 Workshop on Spatial and Temporal Reasoning*, Chicago, 2008.
4. R. Grütter and B. Bauer-Messmer. Combining OWL with RCC for Spatioterminological Reasoning on Environmental Data. In *Proc. of 3rd OWL: Experiences and Directions (OWLED2007)*, 2007.
5. R. Grütter and B. Bauer-Messmer. Towards Spatial Reasoning in the Semantic Web: A Hybrid Knowledge Representation System Architecture. In *Lecture Notes in Geoinformation and Cartography*, Springer, Berlin Heidelberg, 2007.
6. V. Haarslev, C. Lutz, and R. Möller. Foundations of Spatioterminological Reasoning with Description Logics. In *Proc. of 6th Int. Conf. on Principles of Knowledge Representation and Reasoning (KR98)*, pages 112–123, 1998.
7. Y. Katz and B. Cuenca Grau. Representing Qualitative Spatial Information in OWL-DL. In *Proc. of 1st OWL: Experiences and Directions Workshop (OWLED2005)*, Galway, Ireland, 2005.
8. A. Mackworth. Consistency in Networks of Relations. *Artificial Intelligence*, 8(1):99–118, 1977.
9. W. Nutt. On the translation of qualitative spatial reasoning problems into modal logics. In *Proc. of the 23rd Annual German Conf. on AI (KI99)*, 1999.
10. B. Parsia and E. Sirin. Pellet: An OWL DL Reasoner. In *Proc. of Int. Workshop on Description Logics (DL2004)*, 2004.
11. D. Randell, Z. Cui, and A. Cohn. A Spatial Logic based on Regions and Connections. In *Proc. of 3rd Principles of Knowledge Representation and Reasoning (KR92)*, pages 165–176, San Mateo, CA, USA, 1992.
12. J. Renz and B. Nebel. Efficient Methods for Qualitative Spatial Reasoning. In *Proc. of the 13th European Conference on Artificial Intelligence (ECAI98)*, 1998.
13. D. Tsarkov and I. Horrocks. FaCT++ description logic reasoner: System description. In *Proc. of the Int. Joint Conf. on Automated Reasoning (IJCAR06)*, 2006.
14. P. van Beek and D. W. Manchak. The Design and Experimental Analysis of Algorithms for Temporal Reasoning. *JAIR*, 4:1–18, 1996.
15. J. O. Wallgrün, L. Frommberger, D. Wolter, F. Dylla, and C. Freksa. Qualitative Spatial Representation and Reasoning in the SparQ-Toolbox.
16. M. Wessel. On Spatial Reasoning with Description Logics. In *Proc. of Int. Workshop on Description Logics (DL2002)*, pages 156–163, 2002.
17. M. Westphal and S. Wölfl. Confirming the QSR Promise. In *AAAI Spring Symposium on Benchmarking of Qualitative Spatial and Temporal Reasoning Systems.* AAAI Press, 2009.