Computational inference of conceptual trajectory model : considering domain temporal and spatial dimensions

Rouaa Wannous

UNIVERSITY OF LA ROCHELLE

DOCTORAL SCHOOL S2IM

LABORATORY : L3I

THESIS presented by :

Rouaa WANNOUS

defense on : **20 October 2014**
for the degree of : **Doctor of University of La Rochelle**
Discipline : **Computer Science and Applications**

Computational inference of conceptual trajectory model.

Considering domain temporal and spatial dimensions

JURY :

Thomas DEVOGELE	Professor, University of Tours
Florence SEDES	Professor, University of Toulouse, Reviewer
Jérôme GENSEL	Professor, University of Grenoble, Reviewer
Alain BOUJU	HDR, University of La Rochelle, Thesis director
Cécile VINCENT	MCF, University of La Rochelle
Jamal MALKI	MCF, University of La Rochelle

Abstract

Spatio-temporal data describing trajectories of moving objects has increased as a consequence of the increasing availability of such data with current sensors techniques. These devices use different technologies like global navigation satellite system (GNSS), wireless communication, radio-frequency identification (RFID), and sensors techniques. Although capturing technologies differ, the captured data has common spatial and temporal features. Relational database management systems (RDBMS) can be used to store and query that captured data. RDBMS define spatial data types and spatial operations. Recent applications show that the solutions based on traditional data models are not sufficient to consider complex use cases that require advanced data models. A complex use case refers not only to data, but also to the domain expert knowledge. An inference mechanism enriches semantic trajectories with this knowledge. Temporal and spatial reasoning are fundamental for the inference mechanism on semantic trajectories. Several research fields are currently focusing on semantic trajectories to discover more information about the behavior of mobile objects.

In this thesis, we propose a modeling approach based on ontologies. We introduce a high-level trajectory ontology. The temporal and spatial parts form an implicit background of the trajectory model. We choose temporal and spatial models to be integrated with our trajectory model. We apply our modeling approach to a particular domain application : marine mammal trajectories. We model this application and integrate it into our ontology. We implement our approach using RDF. Technically, we use Oracle Semantic Data Technologies. To accomplish reasoning over trajectories, we consider the mobile objects, temporal and spatial knowledge in our ontology. Our approach demonstrates how temporal and spatial relationships that are common in natural language expressions (i.e., relations between time intervals like "before", "after", etc.) are represented in the ontology as user-defined rules. To annotate data with this kind of rules, we need an inference mechanism over trajectory ontology. Experiments over our model using the temporal and spatial reasoning address an inference computation complexity. This complexity is time computations and space storage. In order to reduce the inference complexity, we propose optimizations, such as domain constraints, temporal and spatial neighbor refinements. Controlling the repetition of the inference computation is also proposed. We define a refinement specifically for the application domain. Finally, we evaluate our contribution. Results show their positive impact on reducing the complexity of the inference mechanism. These refinements reduce half of the time computation and allow considering bigger size of data.

Keywords: Trajectory data model, Spatial data model, Time data model, Ontology inference, Domain knowledge, Spatial rules, Temporal rules.

Résumé

Le développement de technologies comme les systèmes de positionnement par satellites (GNSS), les communications sans fil, les sytèmes de radio-identifcation (RFID) et des capteurs a augmenté la disponibilité de données spatio-temporelles décrivant des trajectoires d'objets mobiles. Des bases de données relationnelles peuvent être utilisées pour stocker et questionner les données capturées. Des applications récentes montrent l'intéret d'une approche intégrant des trajectoires « sémantiques » pour intégrer des connaissances sur les comportements d'objets mobiles.

Dans cette thèse, nous proposons une approche basée sur des ontologies. Nous présentons une ontologie pour les trajectoires. Nous appliquons notre approche à l'étude des trajectoires de mammifères marins. Pour permettre l'exploitation de nos connaissances sur les trajectoires, nous considérons l'objet mobile, des relations temporelles et spatiales dans notre ontologie. Nous avons évalué la complexité du mécanisme d'inférence et nous proposons des optimisations, comme l'utilisation d'un voisinage temporel et spatial. Nous proposons également une optimisation liée à notre application. Finalement, nous évaluons notre contribution et les résultats montrent l'impact positif de la réduction de la complexité du mécanisme d'inférence. Ces améliorations réduisent de moitié le temps de calcul et permettent de manipuler des données de plus grande dimension.

Mots-clés: Modélisation de trajectoire, Modélisation de temp, Modélisation de space, Règles temporelles, Inférence, Régles.

Table of contents

Chapter 1

Introduction

Contents

1.1 Background

Over the last few years, there has been a huge collection of a real-time data of mobile objects. This data is obtained by GNSS [1] (GPS [2] or ARGOS [3]), phone location or RFID [4] systems. These location and communication technologies have encouraged collecting spatial, temporal and spatio-temporal data of moving objects [60]. Therefore, there is an increasing necessity to understand and to provide solutions for an efficient analysis and knowledge extraction from this data. It opens new perspectives for using these solutions in applications, such as bird migration monitoring [126], daily trips of employees [141], military applications [91] and marine mammal tracking [135].

Captured data records the movement track of a moving object from a departure to a destination point. Generally, this data is spatio-temporal points consisting of longitude, latitude and time of the capture. However, depending on devices' capabilities, additional data can be captured. For example, a device can record the speed and the temperature of a moving object, weather conditions and may provide measurements of a moving object's environment. The captured data is considered as raw data and a commonly called **trajectories**. A trajectory is inherently a spatio-temporal notion. From users' viewpoint,

1. GNSS : Global Navigation Satellite System
2. GPS : Global Positioning System
3. ARGOS : Advanced Research and Global Observation Satellite
4. RFID : Radio Frequency IDentification

the notion of a trajectory is rooted in the evolving position of some object traveling in a space during a given time interval to achieve a goal or to perform activities. This notion can also help to understand the behavior of a moving object.

Spatio-temporal trajectory data should be used in its context. In general, this context is application domain dependent. Therefore, the latter should provide a benchmark or a framework for efficient interpretation of this data. For example, a traffic analysis based on cars' trajectories may derive meaningful results while incorporating information about road networks. Similarly, studies on bird migration patterns may require an understanding of features of the particular bird species (e.g., their body sizes, food sources, and competitors) as well as information about the weather conditions during their flight. So in this case, we are about to associate raw data captured, called trajectories, with additional data bearing well-defined semantics. This is called **semantic trajectories** [7]. Semantic trajectories can be seen as a high-level data layer associated with raw trajectories' layer. This high-level layer may facilitate the discovery of new knowledge [141]. For example, human trajectories are best understood when captured positions can be labeled with activities performed at places related to these positions. Knowledge about these places and all activities can help the system to better interpret the displacement.

This thesis present a research work conducted on trajectories. We present a trajectory analysis and modeling framework to enrich a hight-level data layer. To meet the challenges imposed by the semantic trajectory notion, we present a modeling approach based on an upper trajectory ontology centered on a triple (object, trajectory, activity). Considering this workspace, we show how we can use our modeling approach in a particular domain application : marine mammal trajectories. This thesis provides solutions to several scientific problems to be considered in a trajectory's modeling approach based on ontologies taking into account the spatial, temporal and domain dimensions.

1.2 Problem definition

We base our work on trajectory data of moving objects. This requires a trajectory data model and a moving object model. Moreover, to enrich data with knowledge, a semantic model should be taken into consideration. Therefore, there is a requirement for a generic model to consider the trajectory, moving objects and semantic models simultaneously. This is represented by a semantic trajectory model shown in Fig. 1.1. This model can consume captured data of trajectories and other external data as shown in Fig. 1.1 link (1). This data is related to an application domain. This requires an application domain trajectory model which consists of domain model, as shown in Fig. 1.1 link (2). The latter will support semantics related to users' needs. In the domain model, we also find the necessary semantics related to the real moving object, its trajectories, its activities and others. This semantics is often designed by a domain expert. In general, considering various facets of data, the semantic trajectory model must be extended by other models, such as temporal and spatial, as shown in Fig. 1.1 link (3).

As it can be noted from our general introduction, various scientific issues need to be addressed. The main issue is to build and design the semantic trajectory model with its

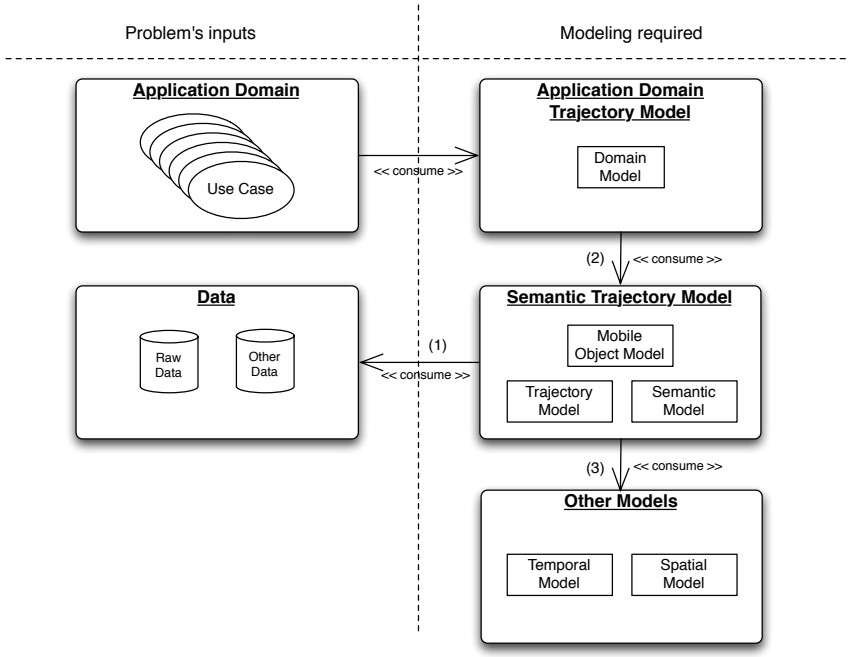

FIGURE 1.1 – Problem and its modeling required

required components. As explained before and shown in Fig. 1.1, our proposed model must take into consideration the following models :

1. Application domain trajectory model : This model has the requirements extracted from the use cases of the application domain. These use cases contain queries and knowledge proposed by domain experts. Therefore, the domain model focuses on mobile object's characteristics and its trajectory's activities ;
2. Other models : An implicit background of the semantic trajectory model could be formed by other models, such as temporal and spatial models.

The semantic trajectory modeling approach is tightly related to the problem of a semantic gap between this model and raw data. Link (1) in Fig. 1.1 presents this gap. Moreover, our model that involves multiple models must establish semantic mappings among them, to ensure interoperability. In Fig. 1.1, links (2) and (3) match the domain, temporal and spatial models with the semantic trajectory model. This matching extends the capabilities of our model. For more efficient semantic capabilities, we want to annotate the data with domain, temporal and spatial knowledge. This knowledge are defined by experts representing users' needs. Annotating data with this knowledge could be done automatically or manually. We cannot use a manual annotation over huge data. Therefore, we choose an automatic annotation which can be accomplished by an ontology inference mechanism. This inference mechanism derives new semantics from existing information using additional knowledge. Figure 1.2 shows this knowledge in a form of rules either supplied by the reasoner or defined by the user. Fundamental axioms are the reasoner's rules, such as RDF, RDFS, OWL or others. User-defined rule is a knowledge related to users' needs. There is a problem of defining these rules in the model in an optimized way.

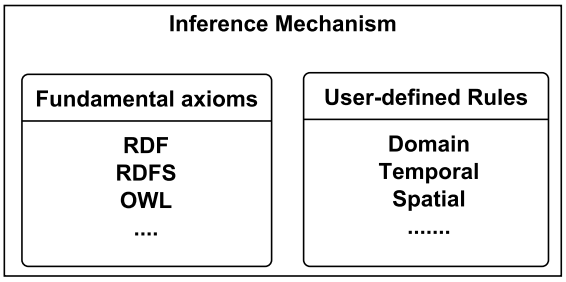

FIGURE 1.2 – Components of the inference mechanism

Oracle evaluates the ontology inference mechanism using fundamental axioms. Oracle inference performance [5] is computed over LUBM ontology and different benchmark data size using RDFS and OWL reasoning. Figure 1.3 presents the results of this evaluation in terms of number of triples inferred and computation time. We notice that the inference is computed over huge data in few hours. However, this is not the case when using user-defined rules. The inference mechanism becomes more complex while using user-defined rules.

Ontology (size)	RDFS		OWLPrime		OWLPrime + Pellet on TBox	
	#Triples inferred (millions)	Time	#Triples inferred (millions)	Time	#Triples inferred (millions)	Time
LUBM50 6.8 million	2.75	12min 14s	3.05	8m 01s	3.25	8min 21s
LUBM1000 133.6 million	55.09	7h 19min	61.25	7hr 0min	65.25	7h 12m
UniProt 20 million	3.4	24min 06s	50.8	3hr 1min	NA	NA

FIGURE 1.3 – Oracle ontology inference performance

1.3 Illustrative examples

This work deals with marine mammals tracking applications, namely seal trajectories. Trajectory data is captured by sensors included in a tag glued to the fur of the animal behind the head. The captured trajectories consist of spatial, temporal and spatio-temporal data. Trajectories data can also contain some meta-data about the marine environment. These datasets are organized into sequences. Each sequence, mapped to a temporal interval, characterizes a defined state of the animal. In our application, we consider three main states of a seal : *hauling out, diving* and *cruising*. Every state is related to a seal's activity. For example, a foraging activity occurs during the state diving. In this application, large datasets need to be analyzed and modeled to tackle the user's requirements. To answer user's queries, we also need to take into account the domain knowledge. Indeed,

domain knowledge is composed both of spatial and temporal knowledge, where and when the moving object activities take place during a given time interval.

In a first step, we assume that the trajectory data is stored and managed in a spatial relational database. Then, we consider the query (Example 1) based on a schema (Code 1.1) of two spatial tables.

Example 1 *Which dives are located within a zone during the time interval : [2007-08-02T00 :00 :00, 2007-08-09T23 :59 :00]*

```
Table Dive (idDive:integer, refSeal:string, maxDepth:real, shape:line(startPoint, endPoint),
            startTime: dateTime, endTime: dataTime);
Table Zone (idZone:integer, name:string, shape:polygon(points[]));
```

Code 1.1 – Spatial schema

To answer the query (Example 1), we need a relational database language supporting spatial data. ISO/IEC 13249-3 SQL/MM [31] is a standardized extensions for multi-media and application-specific packages in SQL. The standard is organized in several parts. The part 3 [127] is the international spatial standard that defines how to store, retrieve and process spatial data using SQL. It also defines how spatial data is represented, and the functions available to convert, compare, and process spatial data in various ways. Code 1.2 gives the SQL/MM expression of the query (Example 1).

```
SELECT D.idDive, D.refSeal
FROM   Dive D, Zone Z
WHERE  Z.shape.ST_Contains(D.shape) AND Z.idZone = 5
       AND startTime => "2007-08-02T00:00:00"
       AND endTime   <= "2007-08-09T23:59:00";
```

Code 1.2 – The SQL/MM query of Example 1

The SQL/MM expression (Code 1.2) is based on a relational model of the trajectory data. This model represents the domain by a set of attributes and their values. Therefore, this model cannot take into account the domain knowledge as given by experts. Let us illustrate the query (Example 2).

Example 2 *In which zones is the seal foraging during the time interval : [2007-08-02T00 :00 :00, 2007-08-09T23 :59 :00]*

Even if the SQL/MM language provides spatial operations to solve the query (Example 1), it is not designed to resolve the query (Example 2). Indeed, the later query combines spatial data (`zone`), spatial operation (`contains`), temporal data (`interval`), temporal operation (`during`) and the semantic domain knowledge related to the seal's activity (`foraging`). We notice that there is a semantic gap between the trajectory data model and the considered operators and activities. We eliminate this gap with an automatic annotation process. This process annotates trajectories or part of the trajectories with semantic annotations. These annotations are domain activities, temporal and spatial operators. The results of this process are semantic trajectories. In the semantic trajectories, each `Sequence` is labeled with one of these activities or these operators. This process is an ontology inference mechanism which goes from low-level data into high-level knowledge.

1.4 Contributions

The work of this thesis is based on a trajectory framework presented in Mefteh thesis [94]. This framework has the following models :

1. Semantic trajectory model : To meet the semantic requirements and fix the semantic gap in this framework, a modeling approach based on ontologies was chosen. The semantic trajectory model consists of :
   (a) Trajectory model : to develop a generic trajectory model, Mefteh et al. [93] consider trajectories of different moving objects : ships and aircrafts. They study the context of each object, collect use cases from required queries, understand and analyze the captured data. Based on these studies, a conceptual model for trajectories of each moving object was presented. Then, they consider common concepts having the same meaning in both models and common features to propose a generic trajectory model.
   (b) Semantic model : an efficient way to take into account a knowledge collected through users' needs. These needs are generally studied by a semantic model. This model can be organized as activities of moving objects contain different base activities ;
   (c) Mobile object model : understanding, analyzing and modeling trajectory data fundamentally leads to define the mobile object model.
2. Application domain trajectory model : The semantic trajectory model is applied over an application domain. Mefteh [94] defined a domain application trajectory ontology. Moreover, she considered domain knowledge experts to formulate semantic requirements or rules. As an example, the `foraging` activity is not a value or a set of values. It is a domain knowledge or a rule in the ontology ;
3. Temporal model : Ontologies of time define concepts that are used in specifying time and temporal elements, such as instant, interval, chronon and etc., and temporal relationships such as after, overlaps and others. We reuse W3C OWL-Time ontology in our framework. We believe that all temporal classes and properties must be considered regardless of the domain model.

In this thesis, we modify this semantic trajectory framework and integrate it with domain, temporal and spatial dimensions. Therefore, the scope of this thesis tackles the following issues :

1. Spatial model : Spatial ontologies define concepts that are used in specifying spatial elements (e.g., point, line, polygon), spatial relationships such as topological relationships, and continuous fields. We reuse a spatial ontology. We believe that all spatial classes and properties must be considered regardless of the domain model. The independent spatial model must contain all the spatial reasoning ;
2. Integration models : We base our approach on the definition of various separated ontologies. Accordingly, we need links between these ontologies. In the data engineering field, this problem is also known as : data integration or mapping. Indeed, the study of this question is not recent and it arose from the need of reusing models ;

3. Data within the model : Our model is an ontological approach based on RDF graph. However, raw trajectories may be provided as files or as relational data. To put raw trajectories into the model, we need a mapping process. This process provides a direct mapping from relational data into RDF graph by D2RQ engine [21]. Then, we can populate data into the ontology. Moreover, we have to resolve the semantic gap between the semantic trajectory model and raw data. Therefore, we map the defined domain knowledge experts with the semantic model's concepts ;
4. Inference mechanism : In our work, an ontology inference mechanism is applied. The knowledge used by this mechanism is the domain experts and the operators of other integrated models. Precisely, the knowledge consists of domain, temporal and spatial rules. The objective of the ontology inference mechanism using these rules is to annotate data with them. For example, to annotate data with a `foraging` activity taking into account `during` temporal operator and `contains` spatial operator, it is necessary to apply the inference mechanism based on this activity and these operators. The results will be a data labeled with semantic annotations ;
5. Inference complexity : We attempt to understand the methodology of the inference mechanism. We compute the inference mechanism over our model with different experiments. We compute the inference using the domain and temporal rules, the domain and spatial rules, and the domain, temporal and spatial rules together. From these experiments, we address the inference computation complexity in terms of time computations and space storage. The size of the data and number of user-defined rules impact the computation of the inference mechanism. This complexity is due to the application of all rules between all pairs of data. We also evaluate the inference mechanism using different engines. Furthermore, to tackle the inference complexity, we define some enhancements. Searching for enchantments, we intervene inside the mechanism of the inference engine which leads to invalidity of the whole database. Then, we modify the definition of rules by limiting their executions with some conditions. These conditions could be temporal and spatial restrictions and constraints. Finally, we visualize the trajectories to view the movement of a moving object and to understand the behavior. From this visualization, we attempt to group interesting data into place-of-interest. Then, we reduce the inference computation by considering place-of-interest instead of the whole data.

In this work, we consider an ontological modeling approach. Detailed later, this model presents a major advantage which lies in the separation between the declarative world, which supports the static aspects, and the imperative world describing the system's dynamics by rules. The global dynamics of the system will be supported by the inference process. Our model is based on a spatio-temporal reasoning and consists of a set of inference rules applying temporal and spatial relationships. Their purpose is to assert additional implied facts in the knowledge base (i.e., determine the spatial or temporal relation between two objects). The formalization frameworks of ontologies also presents various and efficient solutions for data integration.

1.5 Thesis outline

The thesis is organized in four parts :

1. **Extended background and basic concepts** I : In this part, we address the background, foundations needed and some related work. In Chap. 2, we present the basic concepts in the trajectory data modeling approach, time and spatial data models. The semantic technologies for semantic trajectory modeling approach are also illustrated. Chapter3 summarizes related work in trajectory data model in thematic, spatial, temporal and spatio-temporal terms. Work on trajectory ontology modeling approach and ontology inference is also addressed.
2. **Modeling approach** II : This part considers our modeling approach and the way we apply the ontology inference over this model. In Chap. 4, we introduce a generic trajectory model connected to two background models : time and spatial. In Chap. 5, we illustrate an ontology inference mechanism and its necessary parts like spatial and temporal ontology rules.
3. **Research design and implementation** III : In this part, we implement our model and address the application domain. WOur implementation is based on RDF triple store and is illustrated in Chap. 6. Chapter 7 details the implementation of our framework and the computation of the inference mechanism over it. Chapter 8 maps our framework and the case of marine mammal trajectory application. The inference mechanism considers the global framework with the application case. Finally, we propose our contributions to enhance the inference process in general and in the case of the marine mammal application.
4. **Research results** IV : This part evaluates the trajectory ontology inference over our framework in Chap. 9. Finally, experimental results evaluate the impact of the trajectory ontology inference refinements in Chap. 10.

Finally, we conclude the thesis and give some future prospects.

Part I

Extended Background and Basic Concepts

Table of Contents

In this part, we address the background, foundations needed and some related work. In Chap. 2, we present the definition of a trajectory and basic concepts in a trajectory data model. We illustrate time and spatial data models needed for the trajectory data model. In the latter model, a trajectory is considered as data and an ontology modeling approach can be a solution for modeling this data.

In Chap.3, we summarize the related work done on trajectory data model in domain, spatial, temporal and spatio-temporal terms. Specifically, the work on trajectory ontology modeling approach and ontology inference is addressed.

Chapter 2

Foundations

Contents

2.1 Introduction

In this chapter, we address the background and foundations needed. We present the definition of a trajectory, structured trajectory and semantic trajectory. However, there is a need for a generic trajectory data model considering temporal sources, spatial sources or other knowledge. For modeling spatio-temporal trajectory, we illustrate time and spatial data models needed. This generic model should be applicable to different domain applications. An ontology modeling approach can be considered as a solution for modeling the trajectory data. Then, we illustrate the ontology language representation and the reasoning over ontologies to query the extracted knowledge.

2.2 Trajectory data modeling approach

2.2.1 Trajectory definition

A trajectory is considered as a spatio-temporal path covered by a moving object. It represents a semantically meaningful unit of movement for an application. Spaccapietra et al. [126] define a trajectory as follows :

Definition 1 (Trajectory) *[126]. A trajectory is a record of the evolution of the position (perceived as a point) of an object that is moving in space during a given time interval in order to achieve a given goal.*

A trajectory is a spatio-temporal concept where spatial coordinates and time are used to express the position and the time of a moving object, respectively. Most frequently, a moving object is geometrically represented as a set of points. It depends on the domain application. We call `raw trajectory` a captured data from different sensors. Yan et al. [142] define a raw trajectory as follows :

Definition 2 (Raw trajectory $-$ T) *[142]. - A sequence of spatio-temporal points recording the trace of a moving object, i.e. $T = Q_1, ..., Q_m$, where $Q_i = (x, y, t)$ is a triple with the positioning (longitude x, latitude y) at timestamp t.*

However, raw trajectories do not provide any information about the travel, geographical space and background knowledge. Raw trajectories lack semantic information. This semantic information is to understand activities of the moving object. To be useful for the end-user, raw trajectory must undergo a semantic enrichment process that allows the integration of different kinds of knowledge (spatio-temporal, trajectory, geographic, background, domain, etc). The use of semantics was proposed to address these problems [20]. Semantics refer to the meaning of data rather than its syntax or structure. Therefore, there is a need to enrich raw trajectories with semantic information. Semantics here relate to any additional data that is annotated either to a trajectory as a whole or to some of its subparts. Yan et al. [142] define a semantic trajectory as follows :

Definition 3 (Semantic Trajectory) *[142]. A trajectory where spatio-temporal positions are completed with annotations. i.e. $ST = Q'_1, Q'2, ..., Q'm$, where $Q'i = (x, y, t, A)$ is a tuple defining a spatio-temporal point (x, y, t) and its possibly empty set of associated annotations A.*

For example, recording the goal of a person's trip to La Rochelle (e.g., business, tourism) is an annotation at the trajectory level (i.e., holding one value per trajectory). Trajectories should be then enriched with semantic data from the application world. From the user's perspective, trajectory is a semantic spatio-temporal concept because it is viewed as an evolving position of a moving object in some space during a given time interval, having a

specific meaning or goal. Analysis the behavior of moving objects is currently increasing in the research community. Understanding why and how people and animals move, which places they visit, and for which purposes, what are their activities, and which resources they use, is of significant importance for all kinds of decision makers. For example, it is important to observe the "activity" of a moving animal (with activity values "feeding", "resting", "moving", etc.), to infer and record the "means of transportation" used by this moving animal.

For modeling spatio-temporal trajectory, there are two facets of a trajectory [126] : a geometric facet which only considers the point geometry and a semantic facet which gives a meaning or semantic interpretation of application objects. Trajectory components have been enriched by different types of information, then concepts of a semantic trajectory make it possible to extract these information from raw trajectory. Then, any trajectory model must support the characterization of trajectories and their components with attributes, semantic constraints, topological constraints, and links to application objects.

2.2.2 Trajectory and temporal data model

Traveling for achieving a goal takes a limited amount of time (and covers some distance in space), therefore trajectories are inherently defined by the time intervals. Each time interval is delimited by an instant when the object starts a travel and another instant when the travel ends. A trajectory is therefore considered as a temporal concept. This requires a temporal data model to suit the trajectory model.

2.2.2.1 Temporal schema in ISO 19108 standard

The standard ISO 19108:2002 [72] is prepared by technical committee ISO/TC 211, Geographic information/Geomatics. This international standard defines concepts for describing temporal characteristics of geographic information. It provides a basis for defining temporal feature attributes, feature operations, feature relationships, and for defining the temporal aspects of metadata about geographic information. ISO 19103 defines a conceptual schema for describing temporal aspects of geographic information in the Unified Modeling Language (UML). Figure 2.1 shows the UML hierarchy of temporal objects defined by the ISO 19108 standard. These temporal objects are :

- `TM_Object` is an abstract class in the hierarchy ;
- `TM_Primitive` is an abstract class that represents a non-decomposed element of geometry or topology of time ;
- `TM_GeometricPrimitive` provides information about temporal position. The two geometric primitives in the temporal dimension are the instant and the period. `TM_GeometricPrimitive` is an abstract class with two subclasses, `TM_Instant` and `TM_Period`. Figure 2.2 shows these primitives. `TM_GeometricPrimitive` inherits from `TM_Primitive` a dependency on the interface `TM_Order`, and also has a dependency on the interface `TM_Separation` ;
- `TM_TopologicalPrimitive` provides information about connectivity in time ;
- `TM_Complex` is an aggregation of `TM_Primitives` ;

– `TM_TopologicalComplex` is the only subclass of `TM_Complex` that is defined in this standard, and it is an aggregation of connected `TM_TopologicalPrimitives`.

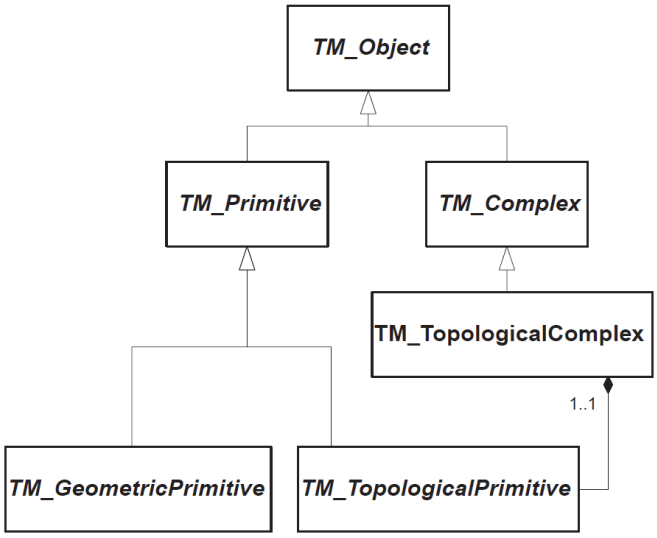

FIGURE 2.1 – The hierarchy of temporal objects defined by ISO 19103 standard

2.2.2.2 Allen temporal interval relationships

To represent actions and events occurring over time, Allen [6] proposed a time model based on time intervals. The author gave an order between two time intervals. In [6], the author introduced the algebra of time intervals and a calculation for temporal reasoning based on 13 temporal relationships or atomic data. Figure 2.3 presents the definition of these relationships.

The standard ISO 19108 [72] uses Allen temporal relationships. Based on them, `TM_Order` provides an operation for determining the position of the `TM_Primitive` relative to another `TM_Primitive`. Values for relative positions are provided by the enumerated data type `TM_RelativePosition`, shown in Figure 2.4.

2.2.3 Trajectory and spatial data model

Most trajectory data models developed so far have considered moving objects with an absolute representation of space [60]. In these trajectory models, a moving object is usually represented using its location as a function of time.

2.2.3.1 Spatial concepts

Formal spatial, and spatio-temporal representations have been studied extensively within a database [58] and recently, the Semantic Web community [109]. Spatial entities (e.g., objects, regions) in classic database systems are typically represented using geometries. A geometry is any geometric shape, such as a point, polygon, or line, and is used as a representation of a

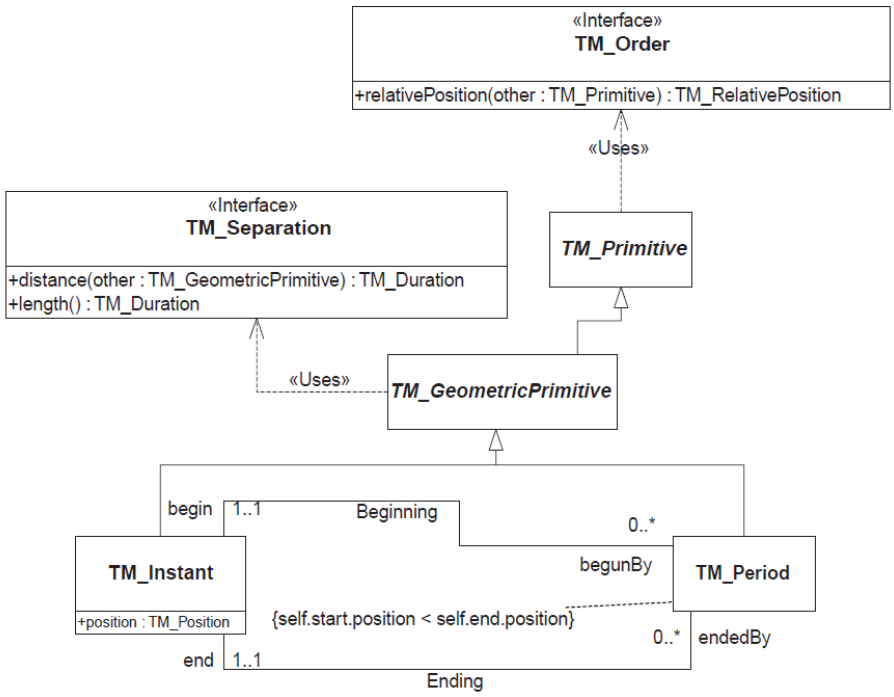

FIGURE 2.2 – The hierarchy of temporal geometric primitives defined by ISO 19103 standard

x RELATION y	KEY	PICTURE
x BEFORE y	B	
x MEETS y	M	
x OVERLAPS y	O	
x FINISHED BY y	Fi	
x CONTAINS y	C	
x STARTS y	S	
x EQUALS y	E	
x STARTED BY y	Si	
x DURING y	D	
x FINISHES y	F	
x OVERLAPPED BY	Oi	
x MET BY y	Mi	
x AFTER y	A	

FIGURE 2.3 – The 13 Allen temporal relationships

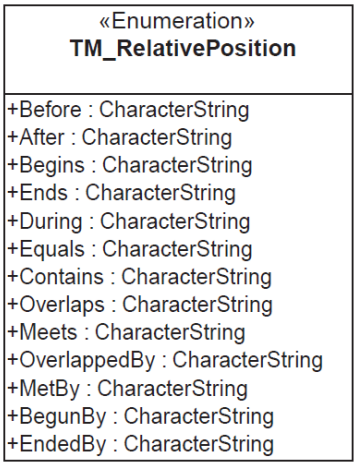

FIGURE 2.4 – The `TM_RelativePosition`

feature's spatial location. A coordinate system describes a location relative to some center. A geocentric coordinate system places the center in the Earth using standard x, y and z ordinates or a Minimum Bounding Rectangle (MBR) enclosing objects or regions and their relationships. Relationships among spatial entities can be topological, orientation or distance based relations.

Any geometry has a coordinate reference system (CRS) (alternatively known as spatial reference system). Elements of a CRS provide context for the coordinates that define a geometry in order to describe accurately their position and establish relationships between sets of coordinates. There are four parts that make up a CRS : a coordinate system, an ellipsoid, a datum, and a projection. An ellipsoid defines an approximation for the center and the shape of the Earth. A datum defines the position of an ellipsoid relative to the center of the earth. WGS84 is a datum that is widely used by GPS devices that approximates the entire world. Geographic coordinate systems use an Earth based datum that transforms an ellipsoid into a representation of the Earth. A geographic (or geodetic) coordinate system uses a spherical surface to determine locations. In such a system, a point is defined by angles measured from the center of the Earth to a point on the surface. These are also known as latitudes and longitudes. The combination of these elements defines a CRS.

2.2.3.2 OGC spatial data model

The OpenGIS Simple Features Interface Standard (SFS) provides a well-defined and common way for applications to store and access feature data in relational or object-relational databases. The spatial data can be used to support other applications through a common feature model, data store and information access interface. The OpenGIS supports simple feature specification for SQL [75]. This specification describes a standard set of the SQL geometry types based on OpenGIS geometry model. Each spatial data is associated with a well-defined spatial reference system (SRID). SRID is a Spatial Reference IDentifier which supports coordinate system to uniquely identify any position on the earth. Latitudes and longitudes can be traced back to arbitrarily exact locations on the surface of the Earth. In this section, we present Open Geospatial Consortium (OGC) spatial data model and its implementation in different database systems. Then, we discuss the implementation of OGC in Oracle Spatial database.

OGC geometry object model. OGC geometry object model is based on extending the Geometry Model specified in the OpenGIS Abstract Specification. It is distributed computing platform neutral and uses Object Modeling Technique (OMT) notation. Figure 2.5 shows the object model for geometry. The base geometry class has subclasses for `Point, Curve, Surface` and `Geometry Collection`. Each geometry object is associated with a Spatial Reference System (SRS) and has a Well-Known Text (WKT) presentation. The geometry model defines two-dimensional collection classes named `MultiPoint, MultiLineString and MultiPolygon` for modeling geometries corresponding to collections of `Points, LineStrings and Polygons` respectively. `MultiCurve and MultiSurface` are introduced as abstract super classes that generalize the collection interfaces to handle `Curves and Surfaces`. Figure 2.5 shows aggregation lines between the leaf collection classes and their element classes.

OGC model in database systems. Nowadays, database management systems (DBMS) handle spatial data by supporting spatial data types. DBMS are extended to handle spatial data capabilities and each type uses a set of properties and methods that correspond to the

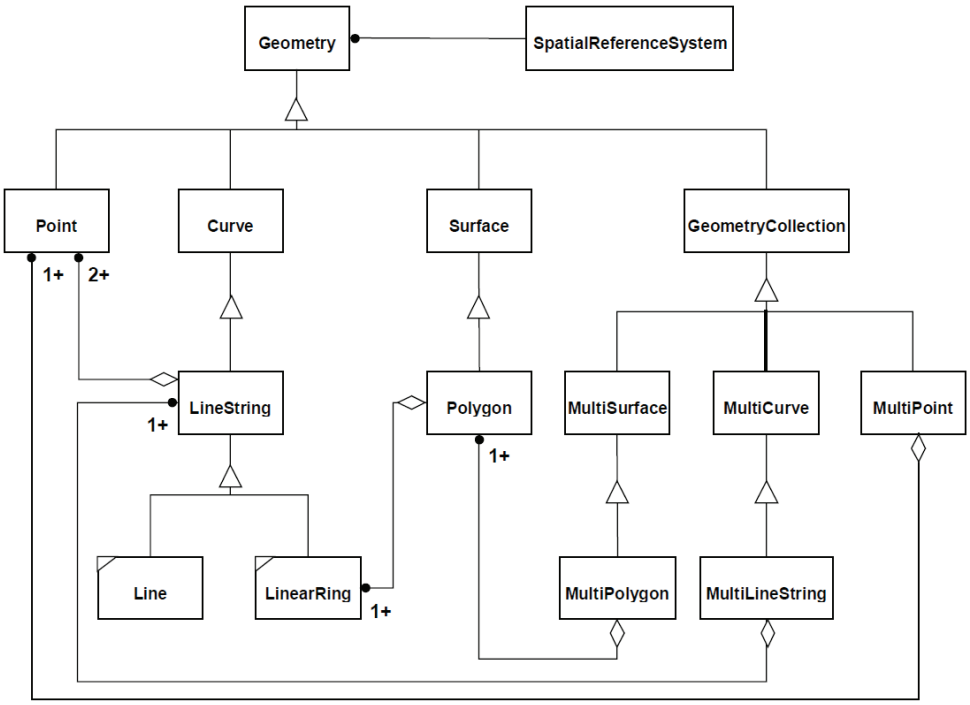

FIGURE 2.5 – The OGC geometry object model hierarchy

OGC functionality. In relational database systems, the OGC geometry model and its collection classes are defined as tables with geometry columns and its features are stored as rows in these tables. In fact, OGC geometry data types are implemented differently in DBMS and these implementations are used in many applications. We give a glimpse of a few examples systems, without being exhaustive.

Common Language Runtime (CLR) types package in SQL Server System contains components implementing geometry, geography, and hierarchy types in SQL Server 2008 R2 [89]. SQL Server implements OGC methods to support spatial operators defining spatial objects relationships. This implementation also supports index over spatial data types. Similarly, MySQL implements a subset of SQL with Geometry Types environment proposed by OGC [137]. This term refers to an SQL environment that has been extended with a set of geometry types as well as functions to create and analyze geometry values. Moreover, PostGIS [112] is a spatial database technology research project. It is based on PostgreSQL DBMS for defining and manipulating spatial data. Other DBMS, like ALTIBASE HDB or DB2 also define spatial extensions to support spatial data.

OGC model in Oracle Spatial.

Oracle DBMS supports the spatial data since version 8 (1997). Spatial data definition and manipulation is done according to OGC specifications. Oracle Spatial [102] is designed to make spatial data management easier and more natural to users of location-enabled applications and GIS applications. Oracle Spatial provides an SQL schema and functions to enable the storage, retrieval, update, and query of spatial features in Oracle database. Oracle Spatial consists of the following :

– A schema (MDSYS) that prescribes the storage, syntax, and semantics of supported geometric data types ;
– A spatial indexing mechanism ;
– Operators, functions, and procedures for performing area-of-interest queries, spatial join queries, and other spatial analysis operations ;
– Topology data model for working with data about nodes, edges, and faces in a topology ;
– Network data model for representing capabilities or objects that are modeled as nodes and links in a network ;
– GeoRaster, a feature to index, query, analyze, and deliver GeoRaster data, that is, raster image and gridded data and its associated metadata.

Oracle Spatial supports the object-relational model for representing geometries. This model corresponds to an "SQL with Geometry Types" implementation of OpenGIS simple feature specification for SQL [75]. Oracle Spatial stores a geometry in a spatial data type `SDO_GEOMETRY`. An Oracle table can contain one or more `SDO_GEOMETRY` columns. A geometry is an ordered sequence of vertices that are connected by straight line segments or circular arcs. The semantics of the geometry are determined by its type. Oracle Spatial supports several primitive types, and geometries composed of collections of these types, including two-dimensional : `Points, Lines, Polygons`, etc.

The spatial relationships are based on geometry locations. The most common spatial relationships are based on topology and distance. To determine spatial relationships between entities in the database, Oracle Spatial has several secondary filter methods :
– The `SDO_RELATE` operator evaluates topological criteria ;
– The `SDO_WITHIN_DISTANCE` operator determines if two spatial objects are within a specified distance of each other ;
– The `SDO_NN` operator identifies the nearest neighbors for a spatial object.

For example, the `SDO_RELATE` operator implements a nine intersection model, which are developed by Clementini and others [26, 27]. This implementation is for categorizing binary topological relationships between `Points, Lines, Polygons`. This yields to the set of spatial relationships as shown in 2.1 :

```
SDO_covers, SDO_coveredby, SDO_contains, SDO_equal, SDO_touch, SDO_inside, SDO_anyinteract, SDO_overlaps
```

Code 2.1 – Topological relationships in Oracle Spatial

Oracle Spatial uses a two-tier query model to resolve spatial queries and spatial joins, in other words, two distinct operations are performed to resolve queries : primary and secondary filter operations. The output of the two combined operations yield the exact result set. The primary filter permits fast selection of candidate records to pass along to the secondary filter. The latter applies exact computations to geometries which result from the primary filter. The secondary filter yields an accurate answer to a spatial query.

2.3 Trajectory and ontology engineering

We've seen in the previous section that there is a need for a generic trajectory data model considering temporal sources, spatial sources or other knowledge. This model should be applicable to different domain applications. Therefore, an ontology modeling approach can be considered as a solution for trajectory data model. In this section, we start with a history of structuring data. Then, we illustrate the representation for an ontology.

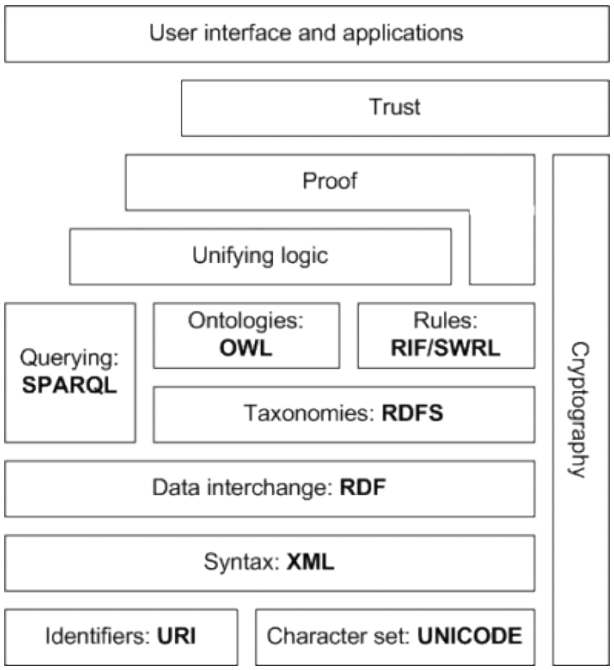

FIGURE 2.6 – Hierarchy for structuring data on the Web

2.3.1 Introduction

Tim Berners-Lee [20], one inventor of the World Wide Web, envisioned in the year 2000 the future of the Internet. He coined the Semantic Web which gives semantics to structured data and responds to complex human requests based on their meaning. Figure 2.6 shows the hierarchy of structuring data on the Web. Data is represented as a Uniform Resource Identifier (URI) and their values as UNICODE. The World Wide Web Consortium (W3C) has adopted Resource Description Framework (RDF) [114] as a standard for representing data. RDF can be represented as a directed graph (edges and nodes). In RDF graph, a directed edge is labeled with a Property which connects two nodes (a Subject to an Object). A Subject, a Predicate and an Object compose an RDF statement commonly called a triple. These triples are stored in a store known as `triplestore`. RDF Schema (RDFS) [115] provides a standard vocabulary for schema-level constructs such as Class, SubClassOf, Domain, and Range. Ontologies [32] provide logical models of concepts, entities and relationships in a specific domain. They are considered as the most common forms of the knowledge representation on the Web [56]. Web Ontology Language (OWL) [92] is a well known standard for structuring data in an ontology. OWL extends RDF/RDFS by defining additional vocabulary for describing classes as logical combinations (e.g., intersection, union, complement) and allows additional assertions about property types (e.g., transitive, symmetric or functional). Rules restrict semantics of concepts of an ontology and conceptual relationships in a specific conceptualization of a particular application domain. Many semantic stores (including Jena [79], Oracle [103], Vituoso RDF Triple Store [104], Sesame [24] and C-Store [1]) use databases to store and process RDF data. Finally, queries can be applied over ontologies to extract information.

2.3.2 Languages for ontology representation

Ontology languages provide essential vocabularies to describe domain knowledge, the underlying common model for data aggregation and integration. We briefly present a list of relevant formal languages used to represent knowledge. The most important ones are referred as OWL-family [92]. The World Wide Web Consortium has proposed ontology representation languages (RDFS and OWL) to capture domain knowledge. OWL is the main language used to describe and share ontologies over the Web, developed as a follow-on from RDF and RDFS [68]. Another important capability of OWL is the ability to define restrictions on the behavior of a property with respect to a given class. In 2009, a new version of OWL (OWL 2) [53] was proposed by the W3C, which aimed to be both an extension and revision of OWL. The motivation for the development of this new version came from the limited expressiveness of OWL, an overly complex syntax and the inability to annotate axioms.

Description logics [9] is a family of knowledge representation languages that seem to be specifically tailored for ontology description. DLs describe knowledge in terms of concepts and role restrictions that are used to automatically derive classification taxonomies. The main effort of the research in knowledge representation is in providing theories and systems for expressing structured knowledge and for accessing and reasoning with it in a principled way. DL languages [10] are in expressive power, depending on the building-operators that are retained for the language. Various studies have examined extensions of the expressive power for such languages and the trade-off in computational complexity for deriving `Is-A` relationships between concepts in such a logic.

2.3.3 Ontology formalization syntax

$SHIQ$ [67] is an expressive DL language, implemented in the FaCT and RACER reasoning [61] systems. Its concepts can be expressed using boolean constructors (conjunction ($\cap$), disjunction ($\cup$) and negation ($\neg$)). $SHIQ$ supports the existential and universal restriction constructors ($\exists R.C$ and $\in R.C$) that define the set of instances which are linked by role (R) to at least (or only to) instances of the concept (C). $SHIQ$ also supports the number restriction constructors ($\leq$nR, $\geq$nR, $= nR$), and the qualified number restriction constructors ($\leq$nR.C, $\geq$nR.C, =nR.C) that define the set of instances that are linked by at least (at most, exactly) n instances of the role R (to C instances).

Based on $SHIQ$, an ontology is represented as a graph $\mathcal{G}$ for a domain D. Common components of an ontology $\mathcal{G}$ are shown in Table2.1. With consideration of two sets : $\mathcal{B}$ is the set of values associated with each basic domain (i.e., integer, string) and Ω is the abstract entity domain such that $\mathcal{B} \cap \Omega = \emptyset$. Defining constraints in an ontology are shown in Table 2.2. Therefore, an ontology $\mathcal{G}$ is a 4-tuple $\langle E, \mathcal{A}, \mathcal{R}, \mathcal{X} \rangle$.

$$\forall (e, a, r, x) \in \mathcal{G} \text{ for a domain } D \text{ where : } (e, a, r, x) \in (E \cup \mathcal{A} \cup \mathcal{R} \cup \mathcal{X}).$$

Over all of these components and constraints, we can represent the size of a graph G as $\#(G)$. This size is defined as the cardinality of the entities, properties, relationships and axioms in the ontology representation : $\#(G) = \#(E) + \#(\mathcal{A}) + \#(\mathcal{R}) + \#(\mathcal{X})$.

TABLE 2.1 – Syntax and semantics of common components of an ontology

Construct name	**Syntax**	**Semantics**	**Definition**
Entities	E	$E_i \subseteq \Omega$	concepts, classes (C), kinds of things, instances (I) from concepts or classes. $E = C \cup I$
Attributes	$\mathcal{A}$	$A_i \subseteq \Omega \times \mathcal{B}$	properties of concepts
Relationships	$\mathcal{R}$	$R_i \subseteq \Omega \times ... \times \Omega = \Omega^n$	ways in which (classes, classes) and (individuals, individuals) can be related to one another
Axioms	$\mathcal{X}$	-	assertions related to the ontology language, in a logical form

TABLE 2.2 – Semantics of constraints of an ontology

Constraint	**Semantics**	**Example**
Attributes	$E \subseteq \{e \in \Omega \vert \#(A \cap (\{e\} \times \mathcal{B}_D)) \geq 1\}$	$\forall x.E(x) \rightarrow \exists y.A(x,y) \wedge D(y)$
Relationships	$R \subseteq E_1 \times ... \times E_n$	$\forall x_1, ..., x_n.R(x_1, ..., x_n) \rightarrow E_1(x_1) \wedge ... \wedge E_n(x_n)$
Cardinality	$E_i \subseteq \{e_i \in \Omega \vert p \leq \#(R \cap (\Omega \times \{e_i\} \times \Omega)) \leq q\}$	$\forall x_i.E(x_i) \rightarrow \exists^{\geq p} x_1, ..., x_{i-1}, x_{i+1}, ...x_n.R(x_1, ..., x_n) \wedge \exists^{\leq q} x_1, ..., x_{i-1}, x_{i+1}, ...x_n.R(x_1, ..., x_n)$
IS_A	$E_i \subseteq E, \forall i = 1, ..., n$	The subsumption of $E_1 is\_a E_2$ means that E_1 is a subconcept of E_2

2.3.4 Ontology reasoning

An ontology language provides different expressive power for an interpretation of the world based on formal semantics and also computational complexity for reasoning. They allow the encoding of knowledge about specific domains and often include reasoning facilities that support the processing of these knowledge. `Inference engines`, also called reasoners, can reason ontology's instances and schema definition for advanced information access and navigation. Supporting an inferencing capability is challenging for several reasons :

- `Expressiveness vs. Tractability tradeoffs` : On one hand, the expressive constructs allow users to compactly capture domain knowledge. On the other hand, the reasoning should not be only tractable but also efficient ;
- `Pre-known semantics vs. user-defined semantics expressed as rules` : User-defined semantics that are expressed as domain rules pose additional challenges. Then, the reasoning should support user-defined rules ;
- `Scalability` : The inference engine should be able to do inferencing on arbitrarily large knowledge bases. Moreover, the inference engine should be able to adapt itself as the amount of derived triples increases.

2.3.5 Querying data through ontologies

Specifying queries for massive volumes of RDF data is becoming every day more and more important. Ontology query languages are developed to query the information defined by these ontology languages and reasoning systems. These languages can be broadly classified into three categories : RDF-based, applying subgraph matching of RDF triples against the ontology graph ; DL-based, supporting directly the DL semantics ; and mixed approaches that combine DL expressivity with DL query conjunction.

Most query languages used for querying RDF documents are SPARQL [40], SeRQL [3] and RQL [54]. The SPARQL query language has become a standard from W3C. It is based on matching graph patterns. The simplest graph pattern is a triple pattern, which corresponds to an RDF triple. SeRQL (Sesame RDF Query Language) is a new RDF/RDFS query language that was developed to address practical requirements from the Sesame user community [5] that was not sufficiently met by other query languages. SeRQL combines features of other languages and adds some of its own. SeRQL handles optional matches, reification and schema queries. SeRQL has recently become the default query language of the Sesame system [24]. RQL is an RDF Schema query language designed by ICS-FORTH [124]. RQL uses a path expression syntax and has a functional approach to query language design. In fact, SeRQL's syntax was in a large part based on RQL. The SPARQL, SeRQL and RQL offer support of features such as Graph transformation [119] or the specification of simple rules. Graph transformation is a powerful tool in application scenarios ; where mapping between different vocabularies is needed.

Finally, Strabon [17] is a semantic spatio-temporal RDF store. It can be used to store linked geospatial data that changes over time and pose queries using two extensions of SPARQL, stSPARQL and stRDF. The stSPARQL can be used to query data represented in an extension of RDF called stRDF. Strabon supports spatial datatypes enabling the serialization of geometric objects in OGC standards WKT and GML. The recent OGC standard is GeoSPARQL [16] which consists of the core, geometry extension and geometry topology extension. Strabon is built by

5. www.OpenRDF.fr

extending the well-known RDF store Sesame [24] and extends Sesame's components to manage domain, spatial and temporal data that is stored in the backend RDBMS.

2.4 Conclusion

In this chapter, we presented the basic concepts related to the notion of trajectory and all the associated definitions. A trajectory is considered as a spatio-temporal concepts. The notion of being a temporal concept, proposed by the ISO 19108 standard model, and temporal relationships are based on Allen's algebra were illustrated. The notion of being a spatial, the OGC spatial model, specifically its implementation in the management system Oracle databases was described. Modeling spatio-temporal trajectory could be based on an ontology approach. Finally, a brief review of the representation, reasoning and querying ontologies aspects was performed.

Chapter 3

Related work

Contents

3.1 Introduction

In this chapter, we summarize some related work done on trajectory data model from conceptual models to semantic models considering domain, spatial, temporal or spatio-temporal terms. Based on an ontology approach for modeling trajectory data, we present some spatio-temporal ontology for semantic trajectories. This leads to reuse online ontologies and to match separated ontologies. So, we present ontology matching approach. Finally, we present some work based on an inference mechanism over ontology models and their complexity.

3.2 Trajectory data modeling approach

3.2.1 Conceptual data model and data mining on trajectories

Mefteh et al. [93] introduced a generic trajectory data model. To develop this model, they considered trajectories of different application domains. They studied the domain context and

analyzed the captured data to present a conceptual model for trajectories of each application domain. Then, they considered the common concepts and common features to propose this generic trajectory model. Our work [136] is based on this generic model. However, Parent et al. [126] also introduced a conceptual model for trajectories. Their model is an evolution of the spatio-temporal MADS model [106] to support trajectories. They addressed trajectories as movements corresponding to semantically meaningful travels (of humans, animals, objects, and phenomena). Components of a trajectory include the definition of when a trajectory starts, when it ends, and when it pauses. These components are fixed by the application, based on semantics given to the trajectories. Therefore, the conceptual model for trajectories supports these components and provides constructs and rules to use movement data as sets of identifiable trajectories traveled by application objects.

Parent et al. [126] considered a trajectory as a segment of spatio-temporal path where the moving object alternatively changes position or stays fixed. They called the former `moves` and the latter `stops`. A `stop` is an interesting place in which a moving entity has stopped or reduced significantly its speed for a sufficient amount of time, likely to accomplish some activity. A `move` is any subset of an object trajectory between consecutive stops. Thus, a trajectory is a sequence of moves going from one stop to the next one (or as a sequence of stops separating the moves). Identifying stops (and moves) within a trajectory is the responsibility of the application. In Parent et al. [126], the stops and moves are semantically defined as follows.

Definition 4 (Stop) *A stop is a part of a trajectory, such that :*
- *The user has explicitly defined this part of the trajectory ($[t_{beginstopx}, t_{endstopx}]$) to represent a stop.*
- *The temporal extent ($[t_{beginstopx}, t_{endstopx}]$) is a non-empty time interval, and*
- *The traveling object does not move, i.e. the spatial range of the trajectory for the ($[t_{beginstopx}, t_{endstopx}]$) interval is a single point. All stops are temporally disjoint, i.e. the temporal extents of two stops are always disjoint.*

Definition 5 (Move) *A move is a part of a trajectory, such that :*
- *The part is delimited by two extremities that represent either two consecutive stops, or t_{begin} and the first stop, or the last stop and t_{end}, or ($[t_{tbegin}, t_{end}]$).*
- *The temporal extent($[t_{beginmovex}, t_{endmovex}]$) is a non-empty time interval, and*
- *The spatial range of the trajectory for the ($[t_{beginmovex}, t_{endmovex}]$) interval is the spatio-temporal line (not a point) defined by the trajectory function (in fact, it is the polyline built upon the sample points in the($[t_{beginmovex}, t_{endmovex}]$) interval).*

Definition 6 (Begin-End) *A beginning, ending of a trajectory is a stop where their temporal extent is a single chronon.*

A trajectory design pattern, Figure 3.1, holds object types for representing trajectories and their `Begin, End, Stops` (`B.E.S`) and `Moves`, where each part contains a set of semantic data. An object type `B.E.S` groups begin, end, and stop objects as instances of the same type, because of their similar features. Each `B.E.S` object is a simple time interval and a geometry point. A relationship type `TrajComp` relates trajectories to their components. Its cardinalities imply that each component belongs to a single trajectory, and each trajectory has at least two components `Begin` and `End`. The two object types `B.E.S` and `Move` are related by two relationship types,

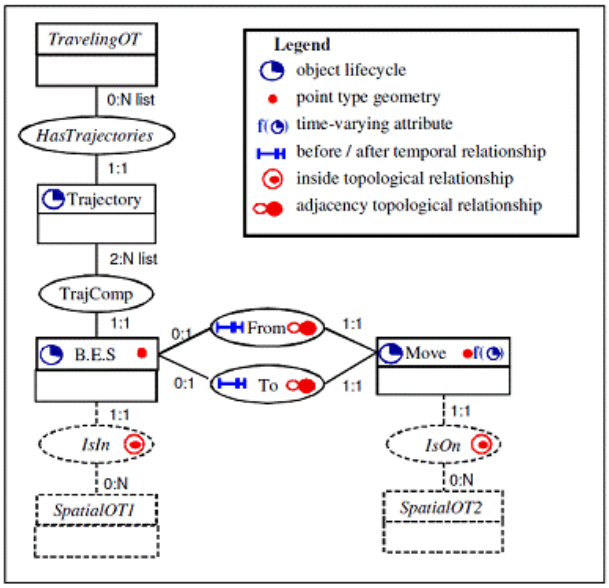

FIGURE 3.1 – Trajectory design pattern from Parent et al. [126]

`From` and `To`, materializing the fact that each move starts and ends in a stop. In addition to expressing the internal structure of a trajectory, trajectories are linked to a hook object type `TravelingOT` that represents the traveling objects covering the trajectories. `B.E.S` may be linked to a hook spatial object type `SpatialOT1` that represents the corresponding location in terms of application objects. The `IsIn` relationship bears a topological inside constraint. Similarly, `moves` may be related to a hook object type `SpatialOT2` by another topological inside relationship, called `IsOn`, that may be used to model network-constrained trajectories. To evaluate Parent et al. [126] approach, bird migration monitoring was analyzed to get better understanding of the bird's behavior. Scientists tried to answer queries such as : where, why and how long birds stop along their travel, what activities did they perform during their stops and which weather conditions did the birds face during their flight.

Based on this conceptual model of trajectories, several studies have been proposed such as Yan [141], Bogorny [22] and Alvares [7]. Bogorny et al. [22] transformed the conceptual data model into a trajectory data mining. They implemented three data mining tasks supported by their model, as shown in Figure 3.2 : trajectory frequent patterns, trajectory sequential patterns and trajectory association rules. A sequential pattern is a list of items ordered in time. A frequent pattern is a set of items, i.e., a set of stops or moves in any order. A trajectory association rule is an association pattern that is composed by a set of items that correspond to the antecedent of a rule and a set of disjoint items corresponding to the consequent of the same rule. Bogorny et al. [22] applied their model on trajectories tourism application.

In Parent et al. [126], patterns had drawbacks when using a relational trajectory model, such as complex maintenance when upgrading the application (a simple update of the application to a given point can have an impact in cascade on other functions of the application), and the application will then be edited in its entirety. Moreover, the functional approach is not suitable to develop applications managing complex phenomena that are constantly changing.

Etienne [41] proposed an approach to data mining of pairing homogeneous groups of trajectories to be analyzed by statistical method. This approach extracts trajectories' spatio-temporal

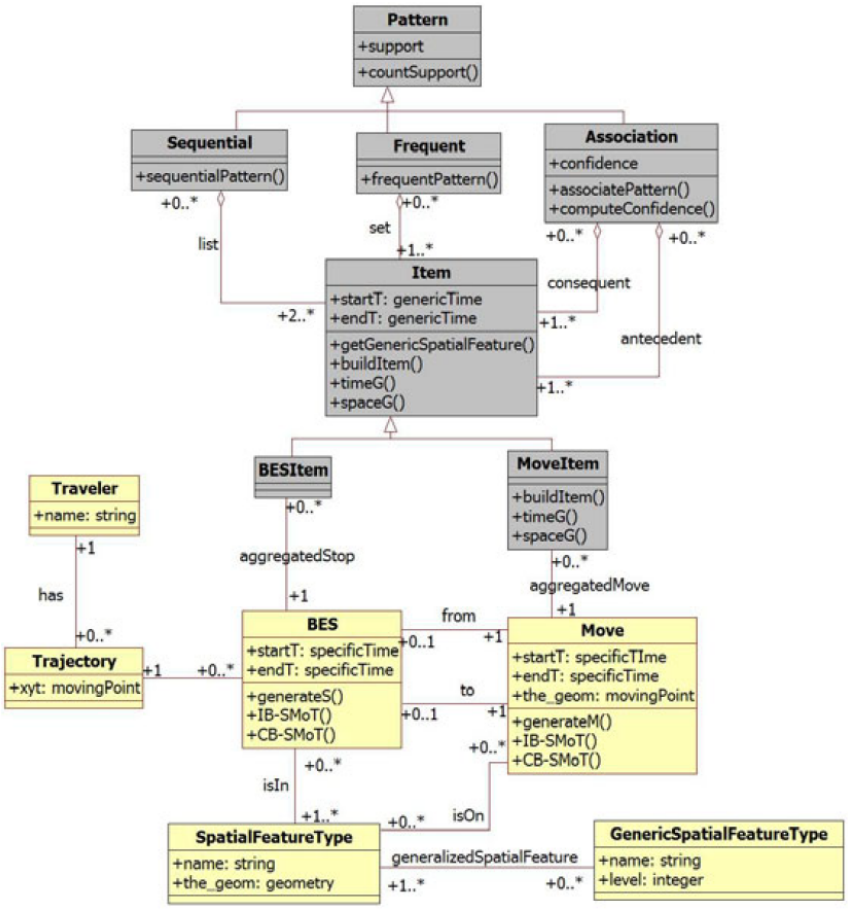

FIGURE 3.2 – Trajectory conceptual modeling to data mining

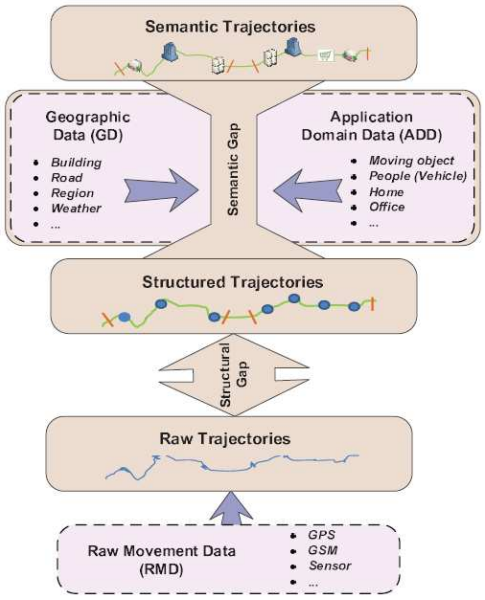

FIGURE 3.3 – Multiple perspective on trajectories

patterns in order to detect abnormal behavior. This research defines tools based on spatial and temporal measures and fuzzy logic to qualify trajectories. These tools are used over this application of maritime traffic. This work is validated using a set of real data from this application. The developed research allows an operator in maritime surveillance to focus its attention on the movement of vessels trajectories qualified as unusual.

3.2.2 Semantic trajectory data model

Most of the research on semantic trajectory started from GeoPKDD project [49]. The GeoPKDD project emphasized the need to address and to use semantic data about moving objects for an efficient analysis of the trajectory. To continue the investigation on the discovery of knowledge and exploitation of moving object data, GeoPKDD was followed first by MODAP [98] and more recently by SEEK [118]. The same community recently presented a survey of the research in the same area [107]. Among the active initiatives aiming at boosting the research on moving object modeling, analysis and visualization, a notable contribution includes the COST action MOVE [99].

Yan [144] presented a semantic trajectory data analysis. He provided a multi-layer trajectory model with multiple perspectives on trajectories, which integrates three kinds of trajectory data : raw movement, geographic data and application domain data, as shown in Figure 3.3.

Alvares et al. [7] adopted the conceptual model [126] for enriching trajectories with semantic geographical information. They applied spatial joins between trajectories and a given set of regions of interest (ROIs), computing frequent moves between stops and two important trajectory episodes. `Episode` is a maximal sub-sequence of a trajectory such that all its spatio-temporal positions comply with a given predicate that bears on the spatio-temporal positions and/or their annotations. Thus, a trajectory is segmented into a list of episodes. The scope of Alvares et al. [7] research is limited to the formal definition of semantic trajectories. Based on the notion

of episodes model, Guc et al. [57] introduced an extensible trajectory annotation model. Their model provided two kinds of annotation elements : `episodes` and `trips`. `Episodes` partitioned a trajectory and described semantically homogeneous sections of the trajectory. `Trips` are defined as groups of sequences of episodes on a higher semantic level which pertain to a common aim.

Parent et al. [107] presented a survey focused on semantic trajectories about mobility. They extended this initial task to cover moving objects' behaviors, whose discovery represents one of the most popular uses of mobility data and possibly the ultimate goal of trajectory analyses.

3.2.3 Semantic spatio-temporal trajectory data model

Recently, much research has focused on moving objects databases (MOD) technology, such as a discrete spatio-temporal trajectory based on moving object database system (DSTTMOD) [95] and moving objects spatio-temporal (MOST) model [138, 105, 123]. The database representation of spatio-temporal trajectories still requires integration of semantic-based approaches necessary for studying of moving objects' activities in space and time. For the representation and modeling of semantic trajectories, we can distinguish two different approaches : a traditional one that includes moving object semantics since the phase of data design, and an a-posteriori approach in which trajectories are annotated by analyzing their raw features.

The first approach, which adopted by Zheni et al. [146], is based on an algebraic spatio-temporal trajectory data type (STT) endowed with a set of operations designed as a way to cover syntax and semantics of a trajectory. The modeling approach represents the concept of space-time trajectories by a series of connected trips and activities that are the usual primitives used in the conceptual apprehension of space-time trajectories. Close to this approach is the work of Pfoser et al. [33], that generates synthetic datasets of semantic trajectories.

The second approach, which can be also referred to semantic of trajectories, is more frequent in the literature. The resulting representation is compliant to the episodes model defined by Alvares et al. [7]. Based on that, Yan et al. [141] presented a hybrid spatio-semantic trajectory model and a computing platform for developing and transforming raw mobility data (GPS) to meaningful semantic trajectories. For example, they represented a spatio-temporal raw data feed at different semantically abstracted levels, starting from basic abstractions (e.g. stop, moves) to enriched higher-level abstractions (e.g. office, shop). Their application concerned daily trips of employees from home to work and back. Their hybrid model, as shown in Figure 3.4, consisted of :

– Data model encapsulating the trajectory definitions available from a raw data perspective.
– Conceptual model as a key mid-level abstraction of a trajectory that allows a progressive abstraction of the raw mobility data.
– Semantic model encapsulating spatio-semantic behavior of the trajectory.

Their trajectory computing platform [141] exploited the spatio-semantic trajectory model and built trajectory instances of the models at every level (spatio-temporal, structural, semantic) from large-scale real-life GPS feed. Figure 3.5 shows four layers in their platform, each containing several techniques for progressive computation of the trajectory instances : data preprocessing layer, trajectory identification layer, trajectory structure layer and semantic enrichment layer.

The two approaches can be combined by trajectory features processing with contextual information, like Yan et al. [142, 143]. The two former steps defined a conceptual semantic trajectory model to describe a trajectory as a sequence of `semantic episodes`. This model corresponds

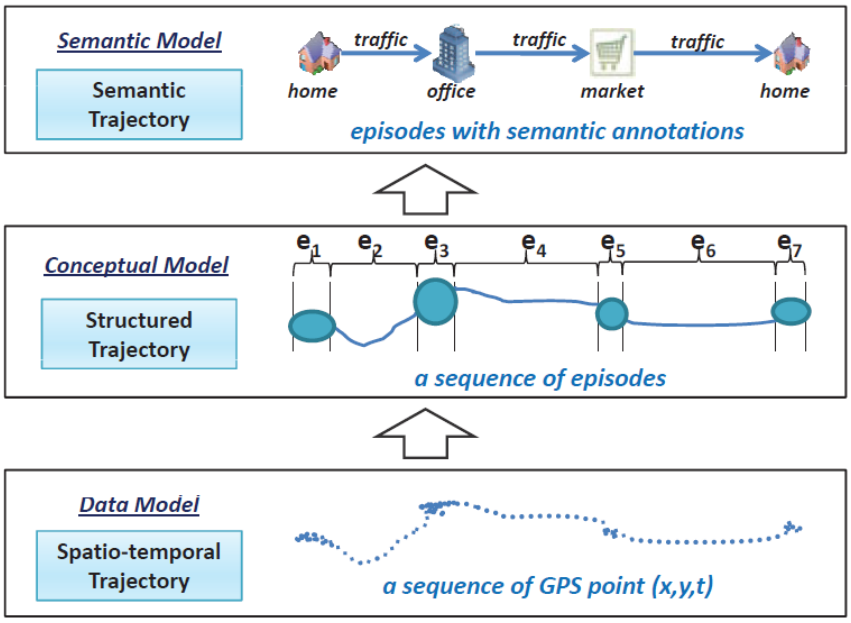

FIGURE 3.4 – Hybrid spatio-semantic trajectory model

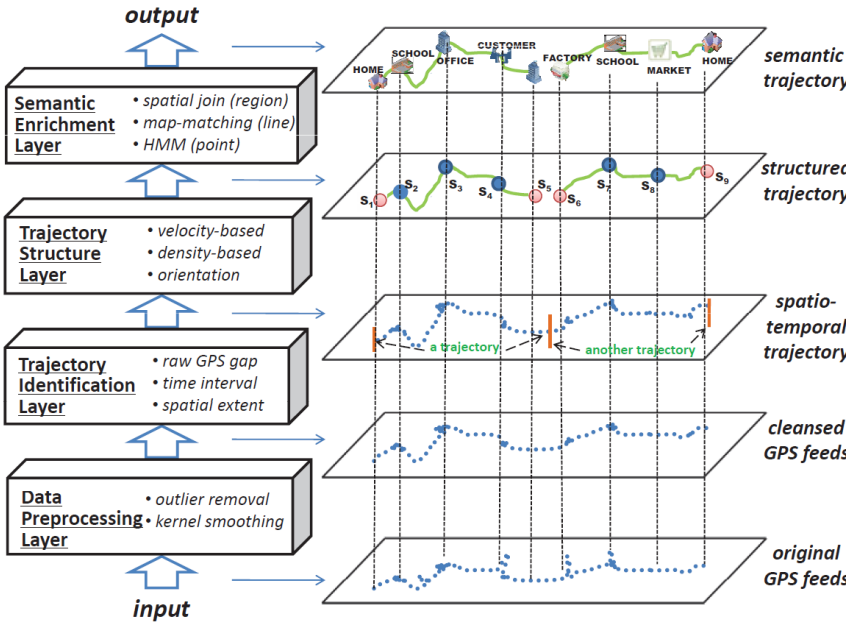

FIGURE 3.5 – Spatio-semantic trajectory computing platform

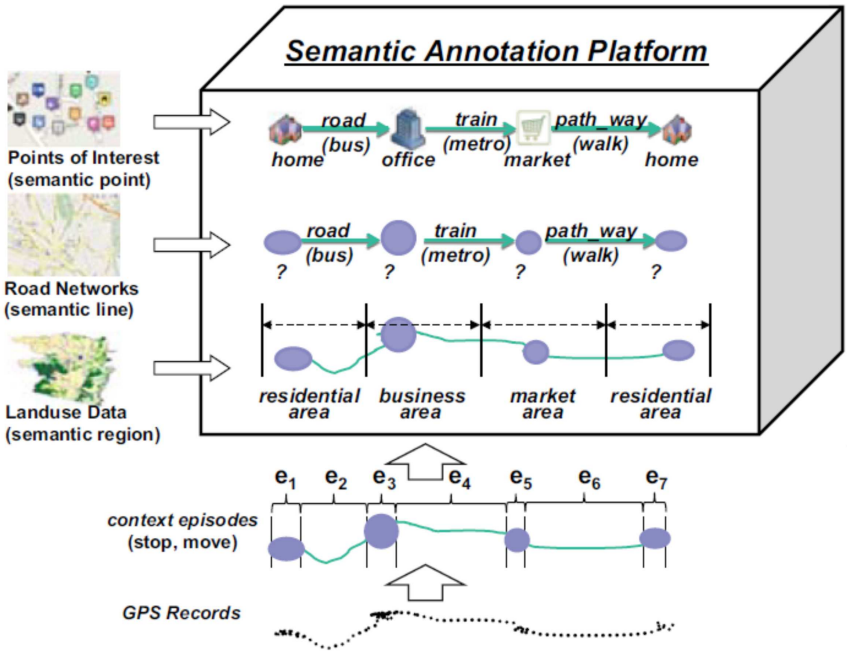

FIGURE 3.6 – Logical view of SeMiTri

to an application's interpretation of trajectories. They developed a framework, called `SeMiTri` (Semantic Middleware for Trajectories). Figure 3.6 shows the logical view of this framework. SeMiTri supports semantic enrichment of raw trajectories, exploiting both geometric properties and background geographic of a stream of spatio-temporal points. They introduced `semantic places` as a semantic counterpart of spatio-temporal positions. Thus, they annotated each spatio-temporal position of a trajectory with links to semantic place objects that the moving object visited. SeMiTri progressively annotates trajectory's episodes with three categories of geographic objects : regions of interest (ROIs) such as park, administrative region and landuse cells (residential, industrial) ; lines of interest (LOIs) such as jogging path, highway and other roads ; and points of interest (POIs) such as bar, restaurant or even a big shopping mall. For example, stop episodes need to be annotated with POIs while move episodes can be integrated with road networks (LOIs). Finally, both moves and stops are annotated as activities, such as walking, driving, cycling, and transportation modes, like bus, car, taxi, etc.

Most advances in the research on trajectories and semantics may be broadly classified among three research areas : spatio-temporal data modeling for the representation of semantic trajectories ; knowledge discovery from data (KDD) for semantic trajectory mining ; and geographic visualization and visual analytics for semantic trajectory visualization.

3.3 Trajectory ontology approach

3.3.1 Trajectory ontology

To exploit raw trajectories, ontology modeling approach is a well known candidate. Yan et al. [140] described a framework for modeling and querying of trajectory data. This framework

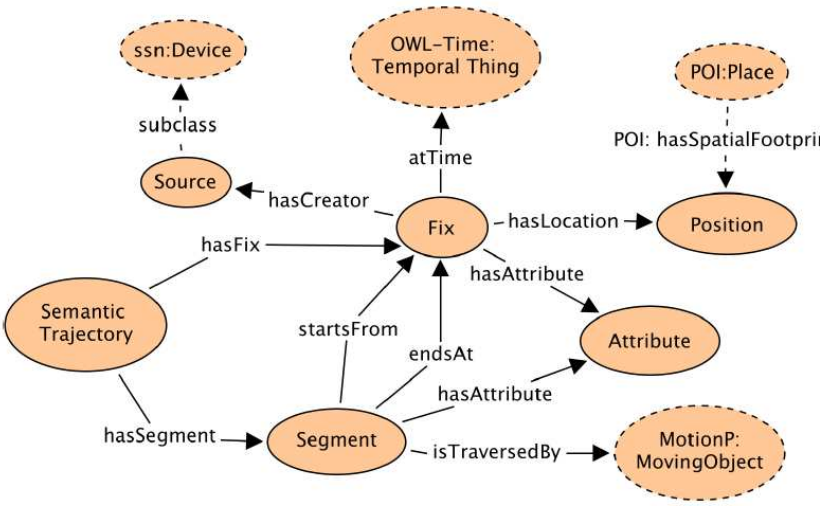

FIGURE 3.7 – Geo-Ontology design pattern for semantic trajectory

relies on the definition of trajectory-related ontologies. Trajectory ontologies are developed for knowledge discovering with aim to understand moving object's behaviors and other phenomena. The work of Yin [147] is focused on activity recognition from trajectory data.

Camossi et al. [25] described a methodology for semantic pattern discovery. This methodology relies on a top-level ontology for modeling moving object trajectories, namely moving object ontology (MOO). The authors studied scenario of maritime surveillance in order to detect suspicious containerized transportations. In this scenario, they defined a knowledge base for the domain of maritime containers, namely the maritime container ontology (MCO) [134], and modeled anomalous container patterns that describe suspicious movement behaviors.

3.3.2 Spatio-temporal ontology for semantic trajectory

Using ontologies for spatio-temporal data modeling is a new research field. Several studies integrate spatial domain, however there is no standard representation of spatial and temporal data in ontologies. In 2007, the Geospatial Incubator Group (GeoXG), a W3C working group, tried to provide an overview of geospatial foundation ontology to represent geospatial concepts [86]. The result of this work is GeoRSS inspired by GML. The use of GeoRSS in the domain of ontologies offers the advantage of a simplified spatial representation, nevertheless it tends to suffer from an overly semantic limitation. Moreover, beyond the formalization of spatial entities, very little work has been carried out concerning the implementation of spatial reasoning procedures in Geospatial ontology [82]. Figure 3.7 shows schematic description of a geo-ontology design pattern for semantic trajectory. This ontology represents a trajectory as a set of `segments` which has a starting and ending spatio-temporal point as `fix`. The authors reused OWL-Time ontology to express the temporal information associated with a fix and point of interest (POI) ontology to include geographic data. Moreover, a geo-ontology design pattern for semantic trajectories [70] presented a GeoSPARQL as a common query language for the Geospatial semantic web to handle and index linked spatio-temporal data. GeoSPARQL is based on OGC [78] with some adaptations for RDF.

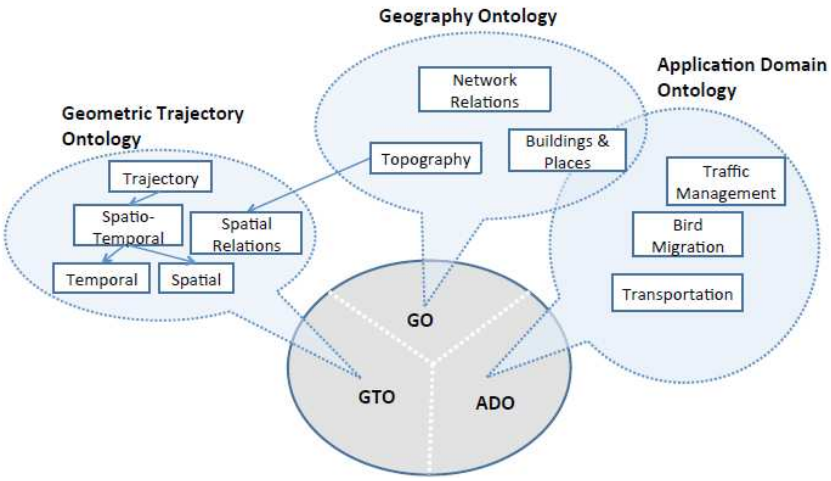

FIGURE 3.8 – Ontological infrastructure for trajectories

For spatio-temporal trajectory, Yan et al. [140] used an ontological approach for representing semantic trajectory. They defined three different ontology modules for representing geometry, geography and the requirements of the application domain. In their model, the three following ontologies were combined to provide a semantic infrastructure necessary to describe the semantic trajectory, as shown in Figure 3.8 :

– The geometric trajectory module describes spatio-temporal features, as a set of structured trajectories compliant with the model they defined in [126].
– The geography module is linked to a geometric trajectory and application modules, because each geometric element has a geographical corresponding according to the application domain. This module constituted of building and places, topography and network relations.
– Application domain module describes all concepts related to the application.

These three component ontologies are integrated into a unique ontology by setting up links between them. These links are shown in Figure 3.9 which presents the semantic trajectory ontology for a traffic management application. For example, the concept `Car` in the traffic domain ontology is connected to the concept `Trajectory` in geometric trajectory ontology by a `hasTrajectory` property.

The work of Vandecasteele et al. [132, 133] is based on these modules and who adopted this approach to detect abnormal ship behavior. Based on these modules, Battle et al. [16] built an ontology design patterns on semantic trajectories, called Geo-Ontology. In the Geo-Ontology, the representation of a semantic trajectory is integrating W3C OWL-Time ontology, the point-of-interest ontology (POI)), domain knowledge and semantic sensor information [96] (W3C SSN-XG ontology) [6]. The authors used their semantic trajectory pattern to annotate two kinds of databases : trajectories generated by human travelers and by animals. This work lacks semantics and does not support inference over domain rules to enhance the semantic trajectories.

In [110], Perry et al. presented an upper-level ontology defining a general hierarchy of thematic and spatial entity classes and associating relationships to connect these entity classes. They combined concepts and relationships from both, the thematic and spatial dimensions, and showed how to deal with temporal semantics in their ontology. To represent two-dimensional space and

6. http ://www.w3.org/2005/Incubator/ssn/wiki/Main_Page

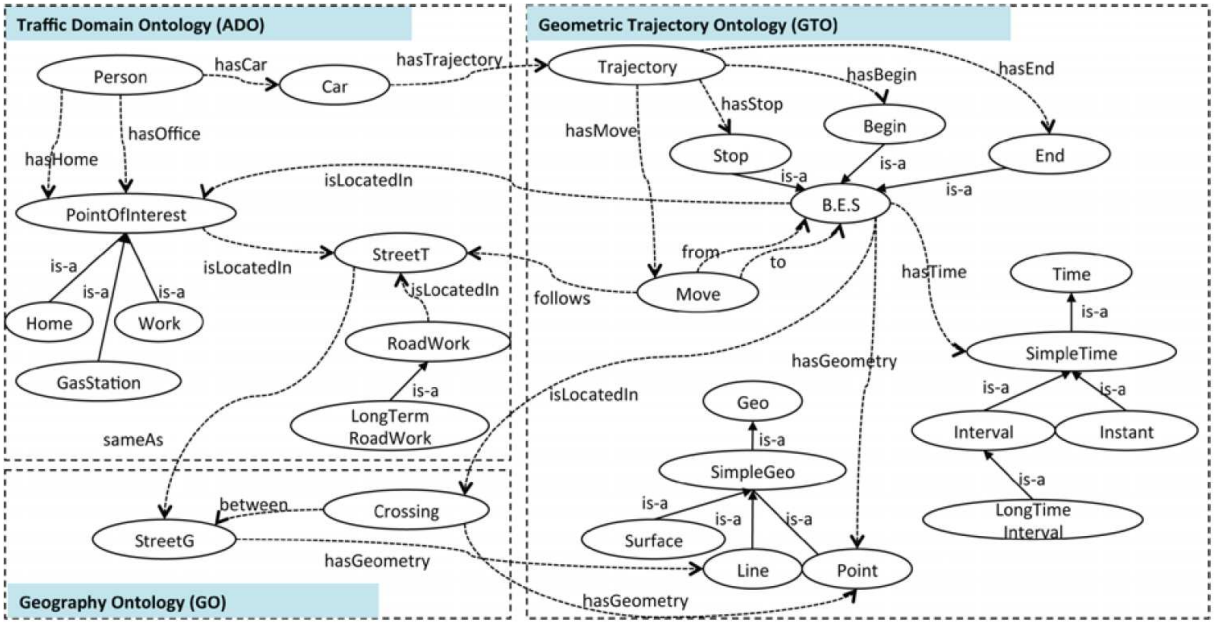

FIGURE 3.9 – Semantic trajectory ontology for a traffic management application

time information in the model, they distinguished entities which persist over time and maintain their identity through change named continuants, and events that happen and then no longer take place, called occurrents. The spatial dimension is presented by a spatial region, a coordinate, and a coordinate system entities. Temporal information is integrated into their ontology by labeling relationship instances with their valid times. Figure 3.10 shows the ontology-based model of space, time, and theme. Moreover, the ontology is formalized by RDFS vocabulary and implemented in a relational database. The main concepts presented in their ontology [109] are :

1. Geographical place representing a geographic feature or a named place.
2. Named place linked to a footprint spatial element to geo reference point, line, or polygon.
3. Dynamic entities representing undefined spatial properties.
4. Entities with static spatial properties referring to buildings, administrations, markets, universities or a city which are presented by named places entities.
5. Events that are special types of entities, representing occurrences in space and time e.g. workshop, inauguration of an institute, etc.

However, Boulmakoul [23] proposed a general moving object meta-model in space-time ontology and event models to structure tremendous data collected data. Their meta-model includes the hybrid spatio-semantic model proposed by Yan [141] and others like spatial model according to OGC spatial data model. Their approach was inspired by ontologies, but the resulting system was database-based. However, in extracting information from the instantiated model during the evaluation phase, they relied on a pure SQL-based approach.

In [136], we work on the marine mammal tracking with the objective of understanding the behavior of the animal. To model semantic trajectories, an ontological approach was defined to represent trajectory concepts integrated with OWL-Time ontology. In [135], we mapped the trajectory ontology to a spatial ontology based on OGC geometry model. The ontologies constructed are formalized in RDF and OWL languages.

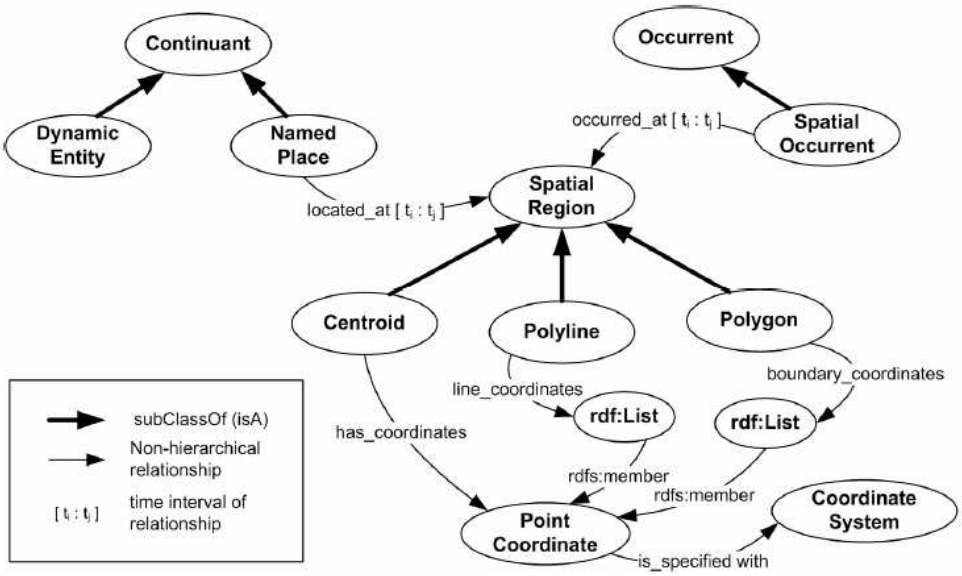

FIGURE 3.10 – Ontology-based model of space, time, and theme

3.4 Reused ontology approach

Ontologies tend to be for many applications, such as information integration, peer-to-peer systems, electronic commerce, semantic web services, social networks, and so on. However, different systems may use different ontologies, thus, the use of ontologies does not avoid heterogeneity. Ontology matching has been used as a solution for the ontology heterogeneity problem. It addresses the combination of multiple ontologies to enable data reuse and integration. Euzenat et al. in [42] provided a repository of information devoted to ontology matching. Ontology matching can be defined as a process of discovering similarities between two ontologies. This process takes two ontologies as input and returns, as output, a set of correspondences (i.e. an alignment) between the entities of these ontologies. It reconciles the differences between ontology entities, in order to facilitate tasks such as information retrieval, information integration, query answering and navigation on the semantic web. Thus, matching ontologies enables the knowledge and data expressed in the matched ontologies to interoperate.

Ontology matching can be processed by exploiting a number of different techniques. To provide a common conceptual basis, researchers have started to identify different types of ontology matching techniques and to propose classifications to distinguish them. Ontology matching techniques, as shown in Figure 3.11, have generally been developed to operate on database schemas [19], XML schemas [8], taxonomies [111], formal languages, entity-relationship models [34], dictionaries, and other label frameworks. For example, Abels et al. [2] proposed a classification that consists of nine matching techniques based on existing literature studies. Another example is the classification developed by Shvaiko and Euzenat [120]. The categories proposed in [39] are the following :

– Element-level techniques : consider ontology entities, such as class or properties, and ignore their relations with other entities. They are usually based on string-based methods, i.e., name similarity or edit distance[85], (for example “title” and “title” are aligned because they are equal strings). They are also based on language-based methods, (i.e., tokenisation, lemmatisation, elimination) and constraint-based methods, (i.e., type similarity, key properties). Furthermore,

external resources such as WordNet [7] database can be used to find the similarity between ontology entities. For example, "author" and "creator" are aligned because they are synonyms in WordNet.

– Structure-level techniques : find alignments by analyzing how ontology entities appear in the structure by comparing their relation with other entities. Consider combinations of elements, such as complex schema structure, model-based method (SAT solvers, DL reasoners), graph-based method (graph homomorphism, path, levels), and taxonomy-based method (taxonomy structure). For example, "Book" and "Essay" are aligned because they have super classes that are aligned.

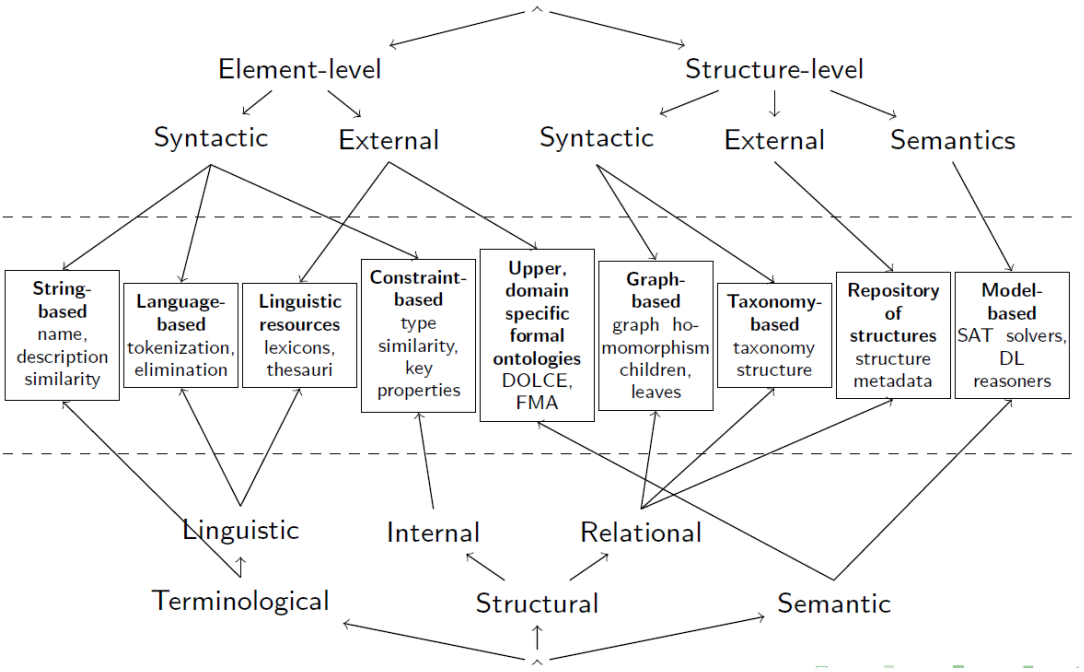

FIGURE 3.11 – Classification of matching techniques

3.5 Ontology inference

In order to exploit a trajectory ontology, it must be integrated with a reasoning engine. The purpose of the inference engine is to infer new facts from existing data and defined rules. Then, the behavior of the moving object can be determined according to the outcome of these rules and the inference engine.

The work conducted by Baglioni et al. [11, 12] is based on the conceptual model on trajectories [126]. They represented annotated trajectories in an ontology encompassing the geographical knowledge and the application domain knowledge. They considered different kinds of stops and temporal knowledge to distinguish between them. Afterwards, they used ontology axioms to infer behavior of patterns using Oracle and OWLPrime to test the axioms. Moreover, Perry et al. in [91] applied an inference mechanism over their ontology. This inference is based on several

7. http ://wordnet.princeton.edu/

domain specific table functions and only on RDFS rules indexes. They used a military application domain and applied complex queries requiring sophisticated inference methods. In their implementation, they used Oracle DBMS and demonstrate the scalability of their approach with a performance study using both synthetic and real-world RDF datasets.

To implement the ontological framework of Yan et al. [140] cited in Sect. 3.3.2, the authors relied on database technology enriched with reasoning capabilities, Oracle Semantics with OWLPrime for ontology representation, querying and inferencing. The ABox of the ontology is containing the ontology instances. However, Vandecasteele et al. [132] proposed an ontology with a spatial dimension. The prototype system aims to analyze ship positions and characterize their behavior according to rules defined by experts. These rules are defined using SWRL language and executed by an inference mechanism. They provided a graphical interface to display the results of the inferences obtained.

Van Hage et al. [131] presented an approach for modeling and analyzing ship trajectories for early time awareness for maritime surveillance and security. This approach takes into account semantics of trajectories and takes as inputs the marine automatic identification system (AIS) as messages sent by ships. They built trajectories and segment them by detecting the significant events that represent changes in ship behavior, such as speeding up, anchored and stopped. Reasoning rules for event labeling are specified in SWI-Prolog, and the geographical knowledge relies on the GeoNames [8] ontology. Moreover, Randell et al. [113] described an interval logic for reasoning about space using a simple ontology that defines functions and relations for expressing and reasoning over spatial regions.

In [135], our presented framework contains domain, trajectory, spatial and time ontologies. We combined these ontology by creating links between their concepts. Focusing on the inference, we defined domain, temporal and spatial rules. We built an inference methodology to execute these rules over the raw trajectories.

3.6 Complexity of the ontological inference

Ontology languages such as OWL are being widely used as the ontology data engineering. With the proliferation of the Semantic Web, more and more large-scale ontologies are being developed in real-world applications to represent and integrate knowledge and data. There is an increasing need for measuring the complexity of these ontologies in order to better understand, maintain, reuse and integrate them. In general, the file size, the number of classes and properties, used in the ontology, effect the inference complexity. Zhang et al. [145] proposed ontology metrics to measure the design complexity of ontologies which also effect the inference complexity. They classified the metrics into two sets : one for measuring the overall design complexity of an ontology (ontology-level metrics), and the other for measuring the complexity of internal structure (class-level metrics).

A key issue in semantic reasoning is the computational complexity of inference tasks on expressive ontology languages such as OWL DL and OWL 2 DL. Theoretical works have established worst-case complexity results for reasoning tasks for these languages. Kang et al. [77] identified a number of metrics that can be used to effectively predict reasoning performance. A set of 8 ontology metrics are defined with the aim of measuring different aspects of the design complexity

8. www.geonames.org

of OWL ontologies. These metrics provide additional insights into ontology engineering and maintenance. These metrics are defined on a graph representation of an ontology and are used as a set of features for predicting reasoner performance. They are divided into the two categories defined by Zhang et al. [145]. In addition to these 8 metrics, Kang et al. [77] defined some other metrics that measure different aspects : an ontology's size and structural characteristics. The metrics are defined on the asserted logical axioms in an ontology and they are divided into two more categories : anonymous class expressions and properties. The authors hypothesized that different metrics may have different effects on ontology classification performance. Feature selection is a very widely-used techniques in data pre-processing to remove irrelevant features. A number of feature selection algorithms are applied to identify and quantitatively study the ontology metrics that have a positive impact on the performance.

3.7 Conclusion

In this chapter, we discussed related work. In reality, there is also the opportunity to complete all the definitions associated with the notion of trajectory. We presented the generic trajectory model proposed by Wafa Mefteh. A trajectory is seen as a set of spatio-temporal sequences. Literature semantic trajectory models was reviewed, and the spatio-temporal semantic models. We present the formalism of ontologies. Again, we presented work related to an ontological representation and spatio-temporal trajectory semantics. The last three sections discuss issues related to integration separated ontologies, possible modes of inference, as well as metrics for assessing the complexity of an ontology.

Part II

Modeling Approach

Table of Contents

In this part, we explain our framework, modeling and reasoning aspects. We represent a general spatio-temporal trajectory data model. Moreover for efficient interpretation of this model, we associate it with an ontology inference mechanism to consider its high-level layers.

Firstly in Chap. 4, we introduce a generic trajectory model. This model contains of three separated models : a mobile object domain, a trajectory domain model and a semantic annotation model. We connect these models to have a semantic trajectory domain model. Enriching trajectories with semantic annotations is to enable the desired interpretations of movements. From these semantic trajectories, we lock forward to extract knowledge about movements' characteristics, in particular the behavioral patterns of moving objects. Moreover, a trajectory is by definition a set of spatio-temporal sequences which trace movements of a moving object in a specific place during a given time interval. Then, space and time models can be used to enrich the description of the concepts in the trajectory modeling approach, to represent their spatial and temporal localization. We reuse on-line time and space models and connect them with our model.

Our main objective is to extract the possible knowledge from our framework to answer user queries. Therefore secondly in Chap. 5, we introduce an inference mechanism as a software application derives new knowledge based on an existing one. This mechanism is based on data and additional aspects called rules. We detail the inference layers, inference rules and the mechanism of the inference engine. Meanwhile, we give examples of some rule languages and examples of inference engine systems. Then to apply the inference mechanism over our trajectory model, we introduce domain, spatial and temporal rules and integrate with with our modeling approach.

Chapter 4

Trajectory ontology

Contents

4.1 Introduction

In this chapter, we introduce a generic trajectory model on which this work is based. This model contains of three separated models : a mobile object domain, a trajectory domain model and a semantic annotation model. We connect these models to have a semantic trajectory domain model. Enriching trajectories with semantic annotations is to interpret these movements. From these semantic trajectories, we lock forward to extract knowledge about movements' characteristics, in particular the behavioral patterns of moving objects. Moreover, spatial and temporal parts form an implicit background of trajectory model. Then, space and time models can be used to enrich the description of the concepts in the trajectory modeling approach and to represent

their spatial and temporal localizations. We reuse online time and space models and connect them with our model.

4.2 Trajectory model

Trajectory data are captured as a series of points coming from different sensors related to moving objects. Understanding, analysis and modeling this data is fundamental for studying moving objects trajectories. Trajectory modeling is proposed for querying moving objects.

In our previous work, we proposed a methodology for modeling trajectory data [94]. This methodology focused on two real cases : boats and plans trajectories. For each case, we define a context, data capture, an analysis process of these data, and then we proposed a UML domain model. From these models, we show that it is possible to define a trajectory pattern in a form of a generic trajectory model.

4.2.1 Mobile object domain model

Mobile objects are considered as a series of moving points in 3D dimensions. Figure 4.1 presents the mobile object domain model. Every `Mobile Object` is equipped with one or more `Sensor` objects. The `Deployment` describes the spatio-temporal relationships between the sensor and the mobile object.

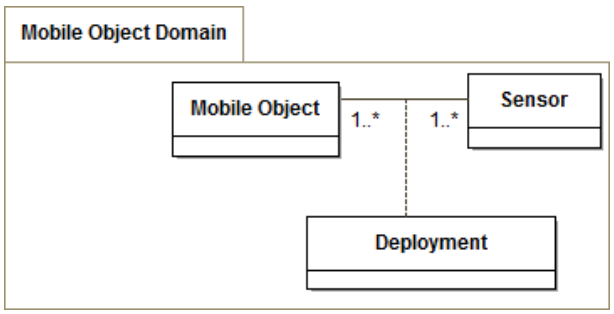

FIGURE 4.1 – Mobile object domain model

4.2.2 Trajectory domain model

The trajectory domain model defines trajectory components independently of any application domain. This high-level data model can be adapted to maintain additional requirements and can be used in many moving object applications. A trajectory is therefore formed by spatial, temporal and spatio-temporal components. More precisely, a trajectory consists of a series of spatio-temporal points representing movements of a moving object.

The class diagram 4.2 presents the trajectory domain model. Table 4.1 provides descriptions and logical forms of the classes of this model. Coordinates of each class are represented in the logical forms, like in Alvares et al. work [7].

TABLE 4.1 – Trajectory domain model dictionary

Concept	**Description**	**Logical form**
Position	spatio-temporal point supports an exploration of the positioning of a moving object at an instant of time. It is characterized by its spatial coordinates (x_i, y_i, z_i) denoting a position in the 3D space, and (t_i) holds a time interval of a capture	$p_i = (x_i, y_i, z_i, t_i)$
Sequence	spatio-temporal interval representing a capture between starting and ending points	$S = \langle p_i, p_j \rangle$
GeoSequence	spatial part of a sequence	$S = \langle p_i, p_j \rangle$
Specific Sequence	metadata associated of a capture	$S = \langle p_i, p_j \rangle$
Trajectory	a set of sequences of the spatio-temporal path covered by a moving object	$T = \langle S_1, ..., S_n \rangle$

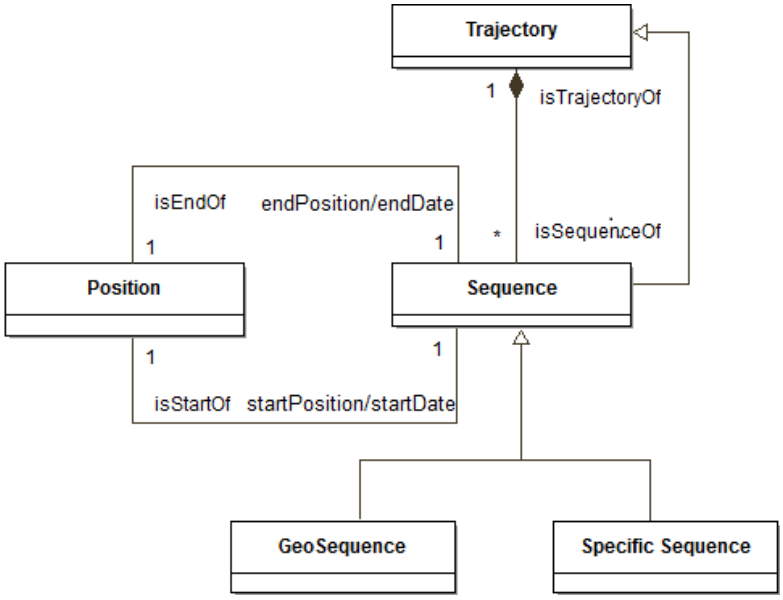

FIGURE 4.2 – Trajectory domain model

4.2.3 Semantic annotation model

Semantic annotation model considers users' needs into the designed model. Figure 4.3 represent the components of the semantic annotation model. It is organized as general activities and a hierarchy of basic activities.

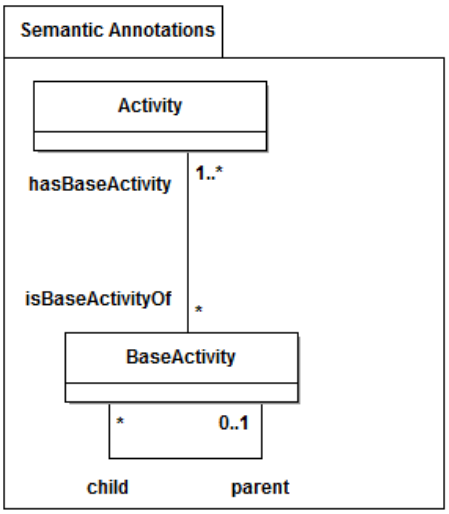

FIGURE 4.3 – Semantic annotation model

4.2.4 Semantic trajectory domain model

The goal of applications using trajectories is to meet as much as possible the needs of users. These requires :

1. A thorough knowledge of the application domain ;
2. Formalization of this knowledge ;
3. Integration of the knowledge in the application.

Semantic integration is an active area of research in databases, information-integration or ontologies. A brief survey by Natalya [101] provided approaches to semantic integration produced by various projects between models. This survey discussed techniques for finding correspondences between models, declarative ways of representing these correspondences, and use of these correspondences in various semantic-integration tasks. Other approaches in semantic integration as statistical methods [62] or probabilistic [88] are also interesting. A semantic trajectory model is proposed by Parent et al. in [107].

A semantic data model is a result of integrating semantic layers to a data model. In this work, we suppose that semantic layers can be defined using semantic annotations. These requires :

1. Constructing trajectories from movements tracks ;
2. Enriching trajectories with semantic annotations to enable the desired interpretations of movements ;
3. Analyzing semantic trajectories and extracting knowledge about movements' characteristics, in particular the behavioral patterns of the moving objects.

The logical form of a semantic trajectory is $ST = \langle S'_1, S'_2, ..., S'_m \rangle$, where $S'_i = (S, A)$ is a tuple defining a sequence (S) and its possibly empty associated activity (A). Figure 4.4 represents the semantic trajectory domain model where the trajectory and sequence concepts have, respectively, activity and base activity.

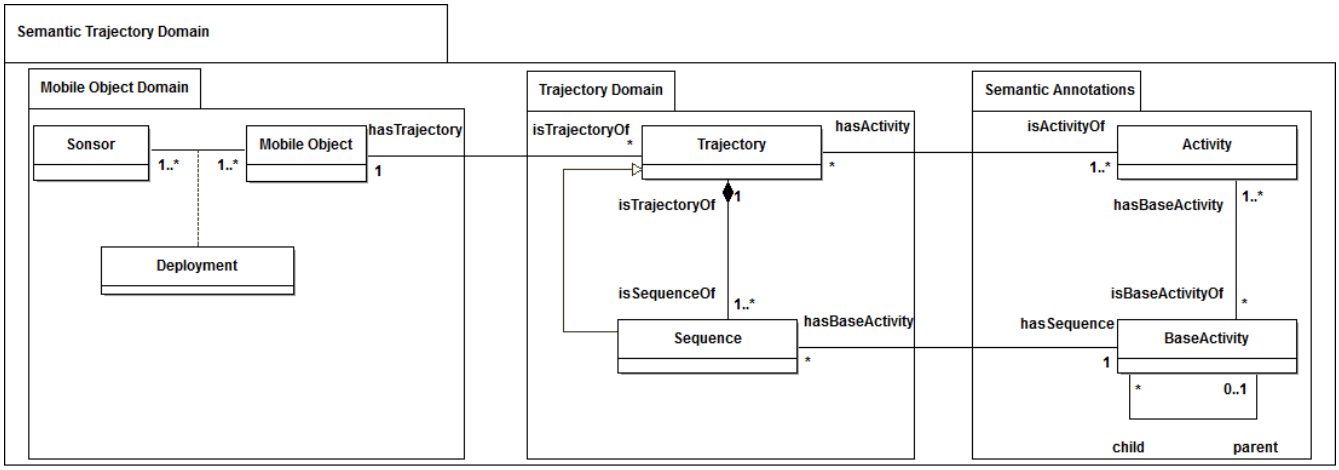

FIGURE 4.4 – Semantic trajectory domain model

4.3 Trajectory ontology

UML class diagram has limitations related to additional requirement constraints needed and imposed by the semantic data definitions. There are several approaches that fully meet the semantic integration problem [101, 18]. In this work, we propose a modeling approach based on ontologies. The class diagram of the trajectory domain model, Figure 4.4, is the starting point for the construction of the declarative part of the trajectory ontology. The concepts of this ontology keep the same definition as the trajectory domain model classes. In the following sections, we detail the different parts of the global trajectory domain model.

4.3.1 Mobile object domain ontology

In `Mobile Object Domain` ontology (MOD), each mobile object has a sensor to capture data. Axioms 4.1 - 4.2 define the properties of the MOD ontology. Figure 4.5 presents the MOD ontology as an upper layer describes the mobile object. The concepts of this ontology keep the same definition as the mobile object domain model classes. Table 4.2 details the relationships between the concepts of the MOD ontology. Later, we can extend the concepts and relationships of the MOD to take into account more information.

$$hasSensor \subseteq MobileObject \times Sensor \tag{4.1}$$

$$hasDeploy \subseteq MobileObject \times Deployment \tag{4.2}$$

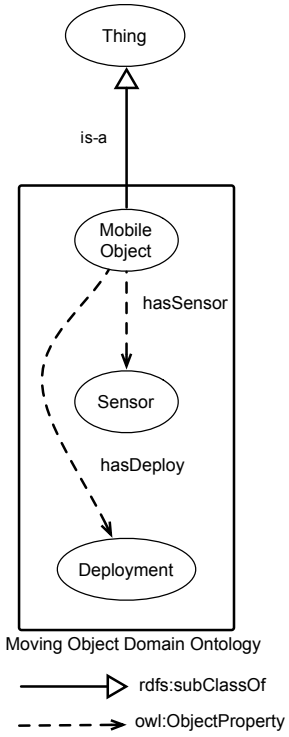

FIGURE 4.5 – Overview of the mobile object domain ontology

TABLE 4.2 – Object properties of the mobile object domain ontology

Object property	**Description**
hasSensor	equips every moving object with a sensor
hasDeploy	describes the spatio-temporal relationships between the sensor and the moving object

4.3.2 Trajectory domain ontology

To define the trajectory domain ontology, we transform this diagram to an OWL ontology called `Trajectory Domain` ontology (TD). Figure 4.6 presents an extract of it. Besides these concepts, Table 4.3 details the object properties between the concepts of the TD ontology.

An encoded formalization for the concepts and relationships of the `TD` ontology is given by Axioms 4.3 - 4.10. A sequence is a spatio-temporal interval. Each sequence could be a `GeoSequence` or a `Specific Sequence`, as forced by Axiom 4.3. Axioms 4.4 - 4.6 enforce that every `GeoSequence` has a `startPosition` and `endPosition`. Axioms 4.7 - 4.8 enforce that every `Sequence` has a `startDate` and `endDate` related to `Instant`. Axioms 4.9 - 4.10 enforce that every sequence belongs to a trajectory and every trajectory contains at least one sequence.

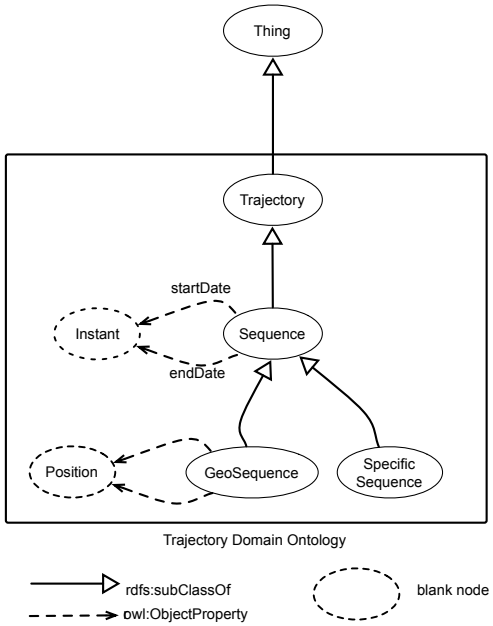

FIGURE 4.6 – Overview of the trajectory domain ontology

TABLE 4.3 – Object properties of the trajectory domain ontology

Object property	**Description**
startPosition, endPosition	represent the starting and ending coordinate of a GeoSequence
startDate, endDate	represent the starting and ending date of a sequence

$$Sequence \subseteq GeoSequence \cup Specific\ Sequence \tag{4.3}$$
$$GeoSequence \subseteq \exists startPosition \cap \exists endPosition \tag{4.4}$$
$$startPosition \subseteq GeoSequence \times Position \tag{4.5}$$
$$endPosition \subseteq GeoSequence \times Position \tag{4.6}$$
$$startDate \subseteq Sequence \times Instant \tag{4.7}$$

$$endDate \subseteq Sequence \times Instant \tag{4.8}$$
$$isSequnceOf \subseteq Sequence \times Trajectory \tag{4.9}$$
$$isTrajectoryOf \subseteq Trajectory \times Sequence \tag{4.10}$$

4.3.3 Semantic domain ontology

Figure 4.7 displays the `Semantic Domain` (SD) ontology. An `Activity` has one or different `BaseActivity`. An encoded formalization for the concepts of the `SD` ontology is given by Axioms 4.11 - 4.12.

$$BaseActivity \subseteq Activity \tag{4.11}$$
$$\forall b.BaseActicity(b) \rightarrow \exists^{>j} a.isBaseActivity(b,a) \wedge Activity(a) \tag{4.12}$$

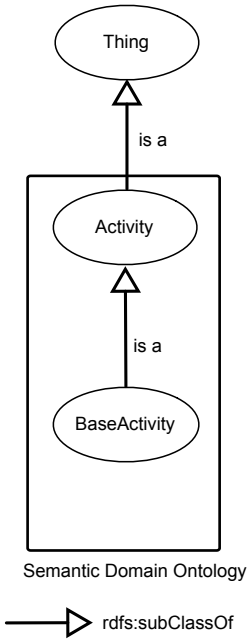

FIGURE 4.7 – An extract of the semantic domain ontology

4.3.4 Semantic trajectory domain ontology

Figure 4.8 presents the `Semantic Trajectory Domain (STD)` ontology, called `owlSemantic-Trajectory`. This ontology contains the three models : mobile object domain ontology, trajectory domain ontology and semantic domain ontology, where :

1. Each trajectory is a trajectory of a mobile object, formally in Axiom 4.13 ;
2. Each trajectory has one or more activity, fomally in Axioms 4.14 - 4.15 ;
3. Each sequence has one base activity, formally in Axioms 4.16 - 4.17.

$$hasTrajectory \subseteq MOD : MobileObject \times TD : Trajectory \tag{4.13}$$
$$TD : Trajectory \subseteq \exists TD : hasActivity \tag{4.14}$$
$$TD : hasActivity \subseteq TD : Trajectory \times SD : Activity \tag{4.15}$$
$$TD : Sequence \subseteq \exists SD : hasBaseActivity \tag{4.16}$$
$$hasBaseActivity \subseteq TD : Sequence \times SD : BaseActivity \tag{4.17}$$

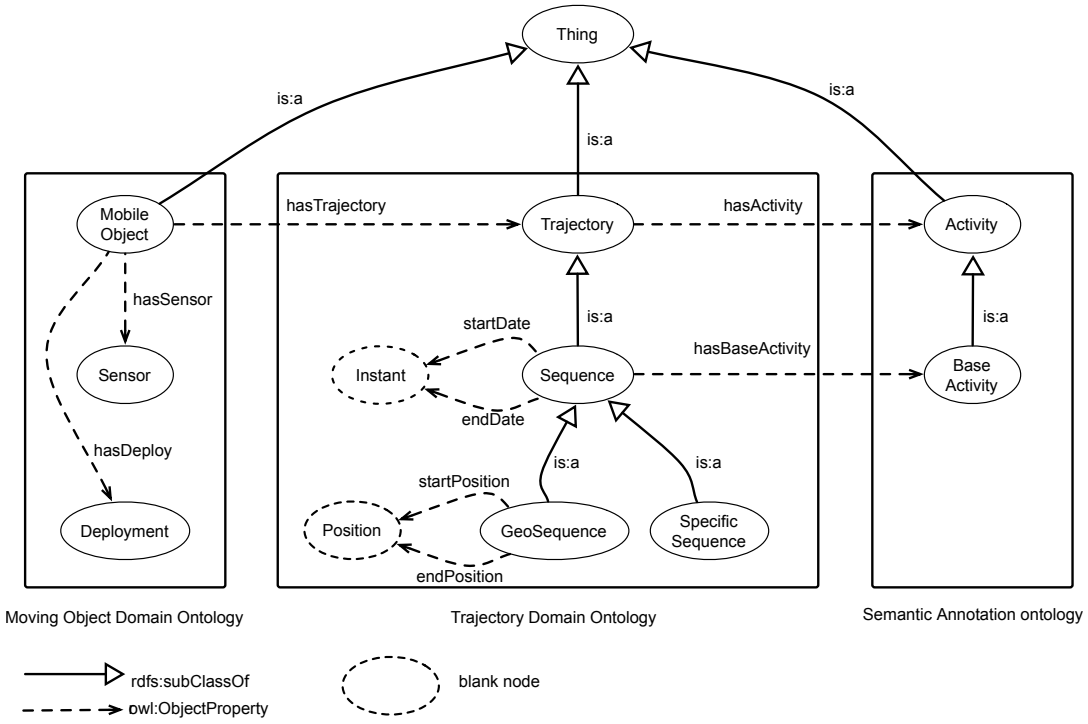

FIGURE 4.8 – An extract of the semantic trajectory domain ontology, `owlSemanticTrajectory`

4.4 Reusing other ontologies

By definition, a trajectory is a set of spatio-temporal concepts. Therefore, our ontology `owlSemanticTrajectory`, presented in Figure 4.8, is formed by these types of concepts. To achieve a complete model, we must consider for each type of concept the ability to connect to an existing ontology model approach. Space and time models can be used to enrich the description of the concepts in the trajectory ontology to represent their spatial and temporal localization. So, we need a spatial and temporal models in order to query trajectories by their spatio-temporal positions. We could answer such as the following queries :

1. Activity of a moving object in a given point ;
2. Activity of a moving object at a specific speed during a given time ;
3. Activity performed during a given time interval in a specific place.

4.4.1 Time ontology

Representing temporal concepts and temporal relationships is also a well researched topic. The requirements of an ontology of time highlight the temporal concepts : `instant` and `interval`. The identification of temporal relationships leads to consider Allen temporal algebra [6].

In [108], Peralta et al. analyzed and tested different temporal ontologies for reusing objective. In particular, `Simple Time Ontology`[9] is a mechanism to support portable ontologies `Ontolingua` [130] and other large ontologies such as `WordNet` [50], `upperCyc` [97], `SUMO` [100] and `LKIF-Core ontology` [66]. The common point among these ontologies is the theory of Allen algebra. However, there is a lack of the structure of the temporal concepts in these ontologies which makes them difficult to be reused. The `Simple Time Ontology` has a structured concepts of time, but lack of documentation.

The `OWL-Time` ontology[10] [65] is developed by the World Wide Web Consortium (W3C). This ontology has a precise specification on temporal concepts and relationships as defined in the theory of Allen, formalized in OWL. Therefore, it is a better candidate for our model as a reused temporal ontology. An extract of the declarative part of this ontology is shown in Fig. 4.9 described in detail in [74].

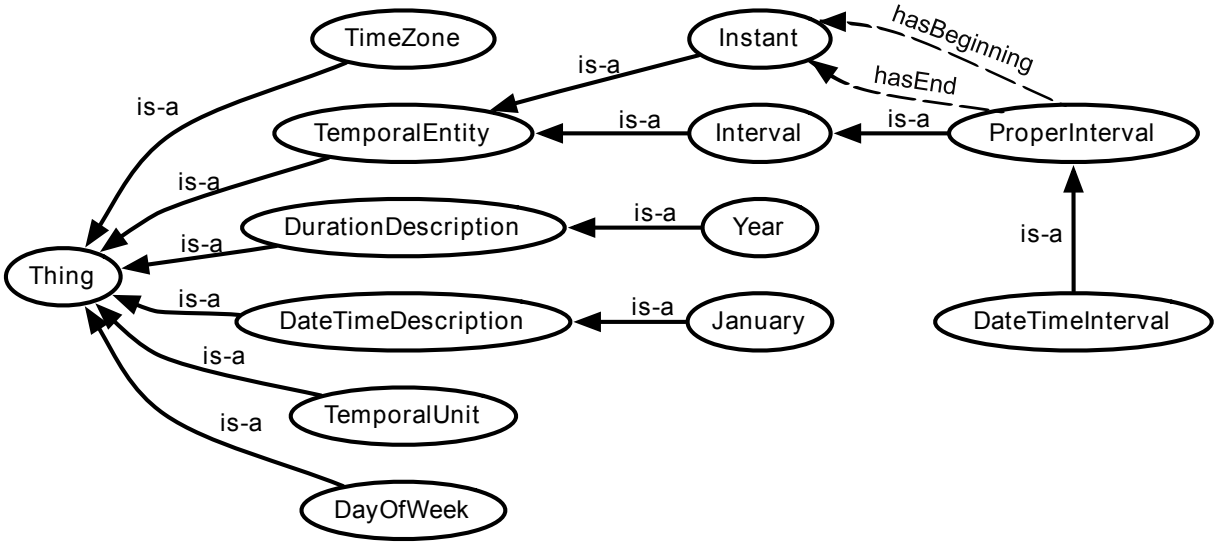

FIGURE 4.9 – A view of `OWL-Time` ontology

4.4.2 Spatial ontology

The requirements of an ontology of space highlight spatial concepts, such as `point`, `line` and `polygon` concepts and others. The identification of spatial relationships leads to consider spatial relationships : `Equals, Within, Touches, Disjoint, Intersects, Crosses, Contains` and `Overlaps`.

The standard `OGC OpenGIS` [75] presents spatial objects and functions over these objects. This standard contains a precise definition of spatial classes and reference systems. In our approach,

9. `Simple Time Ontology` in `http://ontolingua.stanford.edu`
10. `http://www.w3.org/2006/time`

we rewrite the OGC geometry object model (Figure 2.5) in a UML class diagram, then we transform the latter into a formal ontology. We call the resulting spatial ontology `owlOGCSpatial`. Figure 4.10 presents an extract of this ontology.

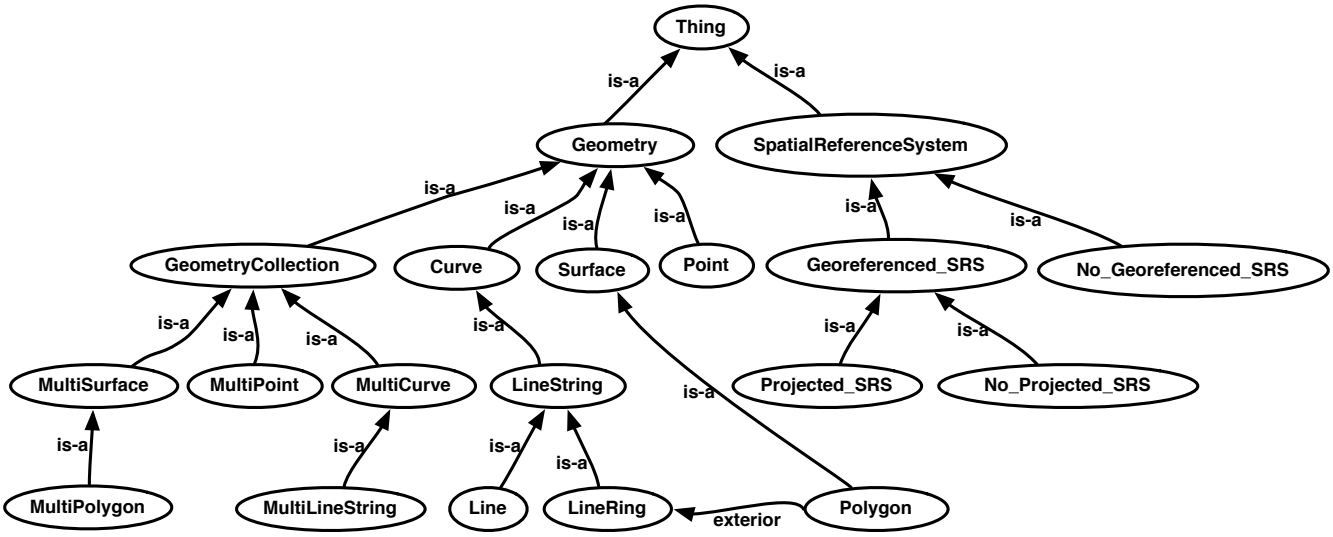

FIGURE 4.10 – A view of `owlOGCSpatial` ontology

4.4.3 Matching trajectory ontology and time/spatial ontologies

4.4.3.1 Matching trajectory and time ontologies

In the `OWL-time` ontology, we are mainly interested in the `ProperInterval` and `Instant` concepts and the two properties `hasBeginning` and `hasEnd` between them. We defined a trajectory ontology (Figure 4.8) where a `Sequence` is a temporal interval having beginning and ending time. To meet these requirements, we need to connect trajectory and time ontologies by finding the alignment between them. Table 4.4 presents the alignment concepts and relationships between trajectory domain ontology and OWL-Time ontology. The concept `Sequence` from the trajectory domain ontology is connected to `ProperInterval` from OWL-Time ontology. Then, the properties of the latter are visible and available for the first.

TABLE 4.4 – Matching trajectory and time ontologies

Trajectory ontology	**OWL-Time ontology**
Sequence	ProperInterval
startDate	hasBeginning
endDate	hasEnd

4.4.3.2 Matching trajectory and spatial ontologies

We defined a trajectory ontology (Figure 4.8) which has to take into account that a `Sequence` has spatial coordinates. To meet these requirements, we need to connect trajectory and spatial

ontologies. Table 4.5 presents the connected concepts between trajectory domain ontology and `owlOGCSpatial` ontology. We connect the `Position` to `Point`. As the `Sequence` has a starting and ending position, we connect it to `Line`.

TABLE 4.5 – Matching trajectory and spatial ontologies

Trajectory ontology	**owlOGCSpatial ontology**
Sequence	Line
Position	Point

4.5 Conclusion

In this chapter, we explained the design of our trajectory model. The concepts and the relationships between the concepts were defined. To meet the semantic trajectory notion, we presented a modeling approach based on an upper trajectory ontology. This ontology contains of three separated ones : 1) a mobile object model, 2) a trajectory model presented as a set of spatio-temporal sequences and 3) a semantic annotation model which is dedicated to the description the activities carried out by a moving object while traveling. The three basic models are then collected in a semantic trajectory model. This modeling approach takes into account the temporal and spatial dimensions. So, we reused the W3C `OWL-Time` ontology and `owlOGCSpatial` ontology for time and space models, respectively. We match the trajectory ontology with the time and spatial ontologies. The objective from these spatio-temporal trajectories is to extend knowledge about moving object movements' characteristics, in particular the behavioral patterns.

Chapter 5

Trajectory ontology inference

Contents

5.1 Introduction

We introduced our framework in Chap. 4, called `owlSemanticTrajectory` in Sect. 4.3, containing and connecting three separated models : mobile object domain ontology, trajectory domain ontology and semantic annotation ontology. We detailed the concepts and the relationships for each ontology. In this chapter, our objective is to extract all the possible knowledge from our framework. Therefore, we need to integrate these concepts and relationships of the three ontologies in an inference mechanism. This mechanism is a software application that extends knowledge based on an existing model. The inference mechanism is based on data and on some additional information defined in an ontology, called `rules`. Data should be populated with the same structure of our model `owlSemanticTrajectory`. We detail the ontology inference mechanism in Sect. 5.2. We introduce rules on which the inference mechanism is based. These rules are related to `owlTime` and `owlOGCSpatial` ontologies in Sect. 5.3.

5.2 Ontology inference

Inferences over ontologies and individuals can be made to extract semantics in the Web Ontology. Inferencing is the ability to make logical deductions based on data and on some additional information defined in an ontology. This additional information is in the form of a vocabulary, e.g., a set of rules or rulebase. Thus, `inference engines`, also called reasoners, are software applications that derive new pieces of knowledge based on previous ones.

5.2.1 Ontology inference layer

Ontology Inference Layer (OIL) is a proposal for a web-based representation and inference layer for ontologies. OIL [71] unifies three important aspects provided by different communities, as shown in Figure 5.1 :

- Description logics [10] are the formal semantics and efficient reasoning support. OIL inherits from Description Logic its *formal semantics* and the *efficient reasoning support* developed for these languages ;
- Frame-based systems are the central modeling primitives of predicate logic. Their central modeling primitives are classes (i.e., frames) with certain properties called attributes. A frame provides a certain context for modeling one aspect of a domain. Therefore, OIL incorporates the essential modeling primitives of frame-based systems into its language. OIL is based on the notion of a concept and the definition of its superclasses and attributes. Relations can also be defined not as attributes of a class, but as independent entities having a certain domain and range ;
- Web standards are a standard proposal for syntactical exchange notations as provided by the Web community. RDF and RDFS are the candidates for a web-based syntax for OIL.

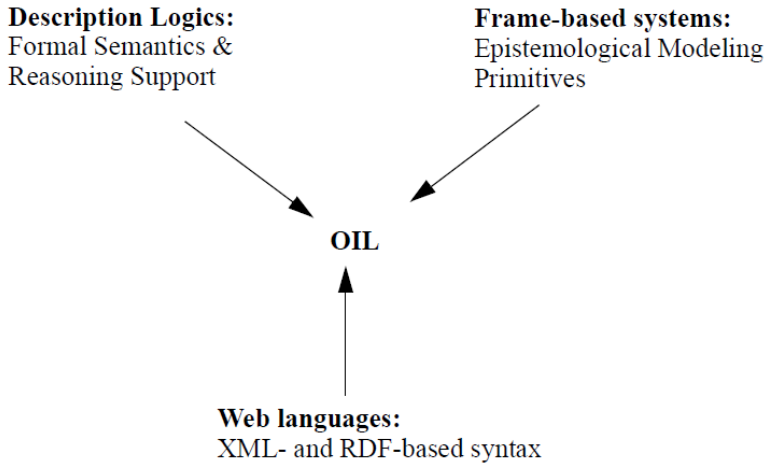

FIGURE 5.1 – The three aspects for ontology inference layer

A simple example may help. The data set to be considered may include the relationship (Flipper is-a Dolphin). An ontology may declare that “every Dolphin is also a Mammal”. This means that the Semantic Web program understanding the notion of “X is also Y” can add the statement (Flipper is-a Mammal) to the set of relationships, although that was not part of the original

data. Another simple example, if we know that John is the father of Mary, we would expect a 'yes' if we query whether John is the parent of Mary. The parent relationship is not asserted, but we know from the ontology example that `fatherOf` is a sub-property of `parentOf`. If '`John fatherOf Mary`' is true, then '`John parentOf Mary`' is also true.

5.2.2 Ontology rules

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. Rules help users to create a repository of knowledge base which are executable. Rules produce new type assertions and new property assertions. In logic, a `rule of inference` (also called a transformation rule) is a function from sets of formulae to formulae based on IF-THEN pattern. The argument is called the premise set (or simply premises) and the value of the conclusion. They can also be viewed as relations holding between premises and conclusions, whereby the conclusion is said to be inferable (or derivable, or deducible) from the premises. If the premise set is empty, then the conclusion is said to be a theorem or axiom of the logic. Rules of inference can be in a form "If p then q" or in a form "p", and returns the conclusion "q". A rule is valid with respect to the semantics of classical logic (as well as the semantics of many other non-classical logics), in the sense that if the premises are true (under an interpretation), then the conclusion is true. Rules must be distinguished from axioms of a theory. In terms of semantics, axioms are valid assertions and usually regarded as starting points for applying rules and generating a set of conclusions. In another words, rules are statements about the system, axioms are statements in the system. Finally, a rule is identified by three parts :

- An IF side pattern for antecedents ;
- An optional filter condition that further restricts the subgraphs matched by the IF side pattern ;
- A THEN side pattern for consequent.

5.2.2.1 Logical rule formalization

Logically, we represent a rule as L. Based on the dictionary Table 6.1, a rule is represented by antecedents, filters and a consequent, as in Axiom 6.1. In Axiom 6.2, each antecedent represents a property ($\mathcal{A}$) for a concept, or a relationship ($\mathcal{R}$) or OWL axiom ($\mathcal{X}$) between two concepts, or between a concept and a defined variable. A rule has a set of filters as in Axiom 6.3. Each filter is applied over a defined variable. A rule has one consequent. A consequent, Axiom 6.4, asserts new relationship to a concept or between two concepts.

$$L^{\mathcal{I}} \subseteq (Antecedent \times \{Filter\} \times Consequent) \tag{5.1}$$

$$Antecedent \subseteq ((\Omega \wedge Type \wedge (\Omega \cup Type \cup Variable)) \times \{Antecedent\}) \tag{5.2}$$

$$Filter \subseteq ((Variable \wedge Operator \wedge \mathcal{B}) \times \{Filter\}) \tag{5.3}$$

$$Consequent \subseteq (\Omega \wedge Type \wedge (\Omega \cup Type)) \tag{5.4}$$

$$Type \subseteq (\mathcal{X}^{\mathcal{I}} \cup \mathcal{A}^{\mathcal{I}} \cup \mathcal{R}^{\mathcal{I}}) \tag{5.5}$$

$$Operator \subseteq (> \cup < \cup = \cup ! = \cup <> \cup >= \cup <=) \quad (5.6)$$

$$Variable : V_1, V_2, ..., V_z \; for \; each \; variable \quad (5.7)$$

5.2.2.2 Integration rules with ontologies

According to the Semantic Web in Figure 2.6, rules are on the top of an ontology. A Web rule language should be useful to express different kinds of rules : "standard-rules", for chaining ontologies properties, such as the transfer of properties from parts to wholes, "bridging-rules" for reasoning across domain, "mapping rules" between Web ontologies for data integration, "querying-rules" for expressing complex queries upon the Web, "meta-rules" for facilitating ontology engineering (acquisition, validation, maintenance) [44].

One important use case for combining rules and ontologies is an ontology alignment, or in general data integration from different data sources. The current status of the Semantic Web standardization effort is specifying standards for ontology and rules layers and how to combine those two layers. The Ontology Layer has reached a certain level of maturity with W3C recommendations such as RDF and OWL, current interest focuses on the Rules Layer. Ontologies are treated as external sources of information, which are accessed by rules that also may provide input to the ontologies. The realization of reasoning with rules and ontologies affects basically four components of the so-called "Semantic Web layer" in Figure 2.6 : RDF, RDFS, Ontology Layer, and Rules Layer. Whereas RDF, RDFS, and OWL are not yet completely clear where and how to fit in rules, possibly involving non-monotonicity, preferences, or other expressive features. This is a limitation of OWL related to the absence of syntactic structures for rule creation. However, these structures enable reasoning and deduction of new facts from information contained in a knowledge base.

Rules and ontologies represent two main components in the Semantic Web vision which are expected to tightly interplay for making this vision a reality. The integration of rules and ontologies is currently under investigation, and many proposals in this direction have been made [47]. Eiter et al. [38] sited two strategies for combining rules as in logic programming and ontologies formalized in classical logic. In the first strategy, rules are introduced by adapting existing semantics for rule languages directly in the Ontology Layer. W3C established Rule Interchange Format (RIF) and Semantic Web Rule Language (SWRL) [69]. Consequently, Horrocks [68] proposed a creation of SWRL that combines OWL DL and RuleML. Unlike OWL, SWRL only allows the addition of relationships and existing properties if they meet the rule. In addition to the OWL predicates, SWRL has supplementary "built-in" functions. These functions extend the initial OWL capabilities ; in particular they enable string comparisons and calculations.

In the second strategy, integration of ontologies and rules is called hybrid approach in which the predicates of rules and ontology are kept strictly separate and only communicate via a "safe interface". A natural choice of rule languages relevant for this integration are those originating from logic programming and non-monotonic reasoning, in particular languages which are based on answer-set programming paradigm [13]. The latter paradigm is a way to realize the Semantic Web Rules Layer. Answer-set programming has its roots in the seminal work by Gelfond et al. [48], who presented a semantics for logic programs with negation as failure and strong negation. This inherent nondeterminism can be exploited to represent different solutions to a problem in the answer sets of a logic program, as in [87]. Answer-set programming [13] plays a central role in the Rules Layer, while OWL/RDF flavors would keep their purpose of description languages, not aimed at intensive reasoning jobs, in the underlying Ontology Layer.

5.2.2.3 Rule languages : examples

`SWRL (Semantic Web Rules Language)` is a proposed language for the Semantic Web that can be used to express rules as well as logic, combining OWL with a RuleML sublanguages of the Rule Markup Language. A SWRL ontology is composed of ordinary OWL axioms and SWRL rules. The rule consists of antecedents and consequents, which both consist of lists of atoms. Since SWRL is an extension of the OWL ontology language, it is restricted to unary and binary DL-predicates. Usually, SWRL rules are part of an OWL ontology encoded in XML or in abstract syntax. The SWRL rules work only on OWL ontologies and are not designed to operate on RDF triples. The proposed rules are of the form of an implication between an antecedent (body) and consequent (head).

The following SWRL rule to assert that the combination of the hasParent and hasBrother properties implies the hasUncle property. SWRL syntax is in Code 5.1.

$$\text{Man}(?x) \wedge \text{hasBrother}(?x, ?y) \wedge \text{hasChild}(?y, ?z) \text{ -> } \text{Uncle}(?x, ?z)$$

Some SWRL rules can be encoded in DL expression. Description logic (DL) is a family of formal knowledge representation languages. DL is used in artificial intelligence for formal reasoning on the concepts of an application domain. It is of particular importance in providing a logical formalism for ontologies and the Semantic Web. Then, we do not need to express a rule in SWRL if we are able to express it in DL. For example, the DL expression for the previous rule is :

$$\text{Man} \sqcap \exists \text{hasBrother}.\exists \text{hasChild}.\top \sqsubseteq \text{Uncle}$$

`Datalog` [37] is a nonprocedural query language based on the logic-programming language Prolog. A user describes the information desired without giving a specific procedure for obtaining that information. The major difficulties in the representation is that in the Datalog facts the attributes are not named. A rule has the following form <head> :-<body> ; Head is a single predicate and the body is a list of predicates. For example, the Datalog for the previous rule is :

Uncle(x,z) :- Man(x), hasBrother(x,y), hasChild(y,z)

```
1  rule ::= '( antecedent -> consequent )'
2  antecedent ::= 'Antecedent( atom )'
3  consequent ::= 'Consequent( atom )'
4
5  atom ::= description '(' i-object ')'
6          | dataRange '(' d-object ')'
7          | individualvaluedPropertyID '(' i-object i-object ')'
8          | datavaluedPropertyID '(' i-object d-object ')'
9          | sameAs '(' i-object i-object ')'
10         | differentFrom '(' i-object i-object ')'
11         | builtIn '(' builtinID { d-object } ')'
12
13 builtinID ::= URIreference
```

Code 5.1 – SWRL syntax

`Notation3 (N3)` is a language for RDF and as a rules language for the Semantic Web. N3 allows declarations of variables. The N3 rule engine is a forward chaining reasoner operating with such rules. Rules may have full N3, even with nested graphs, on both sides of the implication. This gives a form of completeness as rules can generate rules. When used as a rule language on RDF alone, N3 can of course be constrained so that there is no nesting of graphs. Then N" syntax for the rule example is as the following :

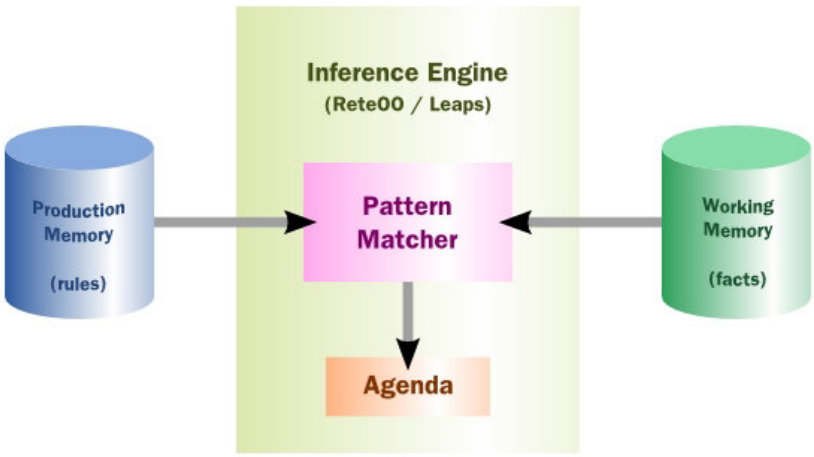

FIGURE 5.2 – High-level view of an inference engine

?x hasType ?Man . ?x hasParent ?y . ?y hasBrother ?z . => ?x hasUncle ?z .

5.2.3 Inference engine

An inference engine is a tool to apply logical rules to the knowledge base and deduce new knowledge. This process would iterate as each new fact in the knowledge base could trigger additional rules in the inference engine. Figure 5.2 shows a high level view of an inference engine. Rules are stored in a Production Memory and the facts that the inference engine matches against are kept in the Working Memory. Facts are asserted into the Working Memory where they may be modified or retracted. A system with a large number of rules and facts may result in many rules being true for the same fact assertion ; these rules are said to be in conflict. The Agenda manages the execution order of these conflicting rules using a Conflict Resolution strategy.

The inference engine can be described as a form of finite state machine with a cycle consisting of three action states : match rules, select rules, and execute rules [122]. In the first state, `match rules`, the inference engine finds all of the rules that are satisfied by the current contents of the data store. When rules are in the typical condition-action form, this means testing the conditions against the Working Memory. The found rule matchings are all candidates for execution : they are collectively referred to as the conflict set. Note that the same rule may appear several times in the conflict set if it matches different subsets of data items. The pair of a rule and a subset of matching data items are called an instantiation of the rule. The inference engine then passes along the conflict set to the second state, `select rules`. In this state, the inference engine applies some selection strategy to determine which rules will actually be executed. The selection strategy can be hard-coded into the engine or may be specified as part of the model. In the larger context of Artificial Intelligence, these selection strategies as often referred to as heuristics. Finally the selected instantiations are passed over to the third state, `execute rules`. The inference engine executes or fires the selected rules, with the instantiation's data items as parameters. Usually the actions in the right-hand side of a rule change the data store, but they may also trigger further processing outside of the inference engine. Since the data store is usually updated by firing rules, a different set of rules will match during the next cycle after these actions are performed. The inference engine then cycles back to the first state and is ready to start over again. This control

mechanism is referred to as the recognize-act cycle. The inference engine stops either on a given number of cycles, controlled by the operator, or on a quiescent state of the data store when no more rules match new data [55].

5.2.3.1 Types of inference engine

There are two types of inference engine : forward chaining and backward chaining :

- `Forward chaining engine` starts with the available data and uses inference rules to extract more data until a goal is reached. An inference engine using forward chaining searches the inference rules until it finds one where the antecedent (If clause) is known to be true. Then, it can conclude, or infer, the consequent (Then clause), resulting as new information to its data. This cascade of rule firings continues until no more rules can fire, as shown in Figure 5.3. As the data determines which rules are selected and used, this method is also called data-driven. Once the inference phase is completed the inference graph will act as if it was the union of all the statements in the original model together with all the statements in the internal deductions graph generated by the rule firings.
  The basic algorithm for many popular rule engines named “RETE algorithm” [46]. This algorithm is orientated to scenarios where forward chaining and ”inferencing” is used to calculate new facts from existing facts, or to filter and discard facts in order to get to some conclusion. The Rete Match Algorithm is an efficient pattern method algorithm for implementing production rule systems and for comparing a large collection of patterns to a large collection of objects. It finds all the objects that match each pattern. The Rete algorithm was designed by Dr Charles L. Forgy’s Ph.D. Thesis.
- `Backward chaining engine` starts with a list of goals (or a hypothesis) and works backwards from the consequent to the antecedent to see if there is data available that will support any of these consequents. An inference engine using backward chaining would search the inference rules until it finds one which has a consequent (Then clause) that matches a desired goal. If the antecedent (If clause) of that rule is not known to be true, then it is added to the list of goals. As the list of goals determines which rules are selected and used, this method is also called goal-driven inference, or hypothesis-driven, because inferences are not performed until the system is made to find a particular goal or question.

5.2.3.2 Inference engine systems : examples

There are many inference engines such as Jess [11], Pellet [12], RacerPro, Jena [13], FaCT++ [14], Drools and Oracle engine [103], etc. We study some of them.

`Jena` [79] is a Java framework for Semantic Web Applications. In addition to providing an API for RDF, RDFS and OWL, it also includes a rule-based inference engine. Jena’s inference engine uses forward and backward chaining and it supports the most common OWL constructs. In addition, it allows users to define their own custom rules.

`OpenRDF Sesame` [24] is a de-facto standard framework for processing RDF data. This includes parsers, storage solutions (RDF databases a.ka. triplestores), reasoning and querying, using the

11. http ://www.jessrules.com
12. http ://clarkparsia.com/pellet
13. http ://jena.sourceforge.net
14. http ://owl.man.ac.uk/factplusplus/

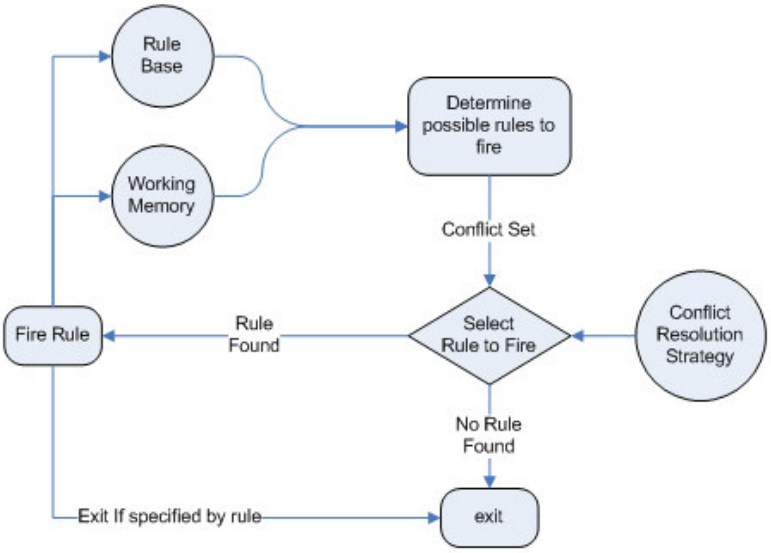

FIGURE 5.3 – Forward chaining inference engine

SPARQL query language. It provides an inference engine for RDFS that uses forward chaining and materialization of the data. It does not support the OWL constructs.

`BigOWLIM` [80] is a scalable application that is fully compatible with the Sesame RDF framework. OWLIM supports RDFS, some OWL constructs [64], and extensions with user-defined rules. The OWLIM reasoner uses two strategies called forward and backward chaining. The rule language has three sections named Prefices for defining namespaces, Axioms for asserting free variable triples and Rules for defining conditions and consequences.

`Drools` is a forward chaining rule engine that uses the rule-based approach, more correctly known as a production rule system. Drools is a rules engine based on Charles Forgy's Rete algorithm tailored for the Java language. Drools 5.2 provides a hybrid chaining with both forward and backwards.

`Pellet` is an open source reasoner for OWL 2 DL in Java. It provides standard and cutting-edge reasoning services for OWL ontologies. For semantically-enabled applications that need to represent and reason about information using OWL, Pellet is the leading choice for systems where sound-and-complete OWL DL reasoning is essential. Pellet is a core component of ontology-based data management applications. It also incorporates various optimization techniques, including novel optimizations for nominals, conjunctive query answering, and incremental reasoning. PelletSpatial [128] extends the Pellet OWL reasoner with qualitative spatial reasoning capabilities. It supports checking the consistency of spatial relations expressed using RCC-8 calculi and computes new spatial inferences from asserted relations. The spatial relations are expressed in RDF/OWL and can be combined with arbitrary domain ontologies.

`Oracle` 11g supports RDFS/OWL and comes with a reasoning engine, more specifically, OWLPrime [103]. Its inference engine is implemented as a database application. It supports forward-chaining rules and extends its SQL dialect with new constructs for querying RDF inside of Oracle's relational DBMS. Inferencing involves the use of rules, either supplied by the reasoner or defined by the user. The inference engine composes code that when executed by the processor, executes inference rules for an RDF model built on the data saved on a triple table

(semantic data table). At the data level, inference is a process of deriving implicit relationships (new triples). Every created triple from the execution of the inference rules is also stored in the semantic data table. Inferencing, or computing entailment, is a major contribution of semantic technologies that differentiates them from other technologies. This inference engine computes production rule based entailment and Oracle supports two kind of regimes [125] :

1. Standard entailment : there are several standard entailment regimes : RDF, RDFS and OWL encoded semantic data models. Support for RDF and RDFS is simplified by the availability of axioms and rules that represent their semantics. Support for major subsets of OWL-Lite and OWL-DL vocabularies have been provided. In this work, we use the subset `OWLPRIME` [139] vocabulary which has approximately 50 inference rules that capture the semantics of the language constructs ;
2. Custom entailment : since the standard vocabularies cannot handle all varieties of semantic application data, it becomes important to provide support for entailment based on arbitrary user-defined rules.

5.3 Trajectory ontology rules

Our spatio-temporal trajectory modeling approach is based on three ontologies : trajectory ontology, time ontology and spatial ontology. Reasoning over trajectory data is preformed by introducing a set of rules operating on temporal and spatial relationships. Reasoners that support rules can be used for inference and consistency checking over spatio-temporal relationships. In addition to reasoning applying on temporal and spatial relations, reasoner applies to the ontology schema to infer additional facts using OWL semantics.

Then, we integrate rules into our ontology model. Considering the three models, we have domain, temporal and spatial rules.

5.3.1 Domain ontology rules

In the domain dimension, we need to understand the behavior of moving objects. Therefore, we focus on defining activities of the moving objects. These activities are called domain knowledge. These activities are carried out through moving objects' trajectories during a given time interval. To define them, we need to contact domain knowledge experts. Then, we consider this knowledge to formulate requirements or rules.

This knowledge can be defined in accordance to a domain application. For example in the bird application, activities could be flying, feeding or resting. In the human application, activities could be sleeping, traveling, eating, walking or working. In our case, the domain ontology rules will be explained in Chap. 8.

5.3.2 Time ontology rules

Reasoning is preformed by introducing a set of rules operating on temporal intervals. Reasoners that support rules can be used for inference and consistency checking over temporal relations. In addition to reasoning applied on temporal relations, a reasoner is applied to the ontology schema to infer additional facts using OWL semantics.

The temporal reasoning rules are based on the 13 Allen relationships [6], as shown in Figure 2.3 in Sect.2.2.2. These are : `intervalEquals`, `intervalBefore`, `intervalDuring`, `intervalOverlaps`, `intervalOverlappedBy`, `intervalStartedBy`, `intervalFinishedBy`, `intervalFinishes`, `intervalContains`, `intervalMetBy`, `intervalMeets`, `intervalStarts`, and `intervalAfter`.

5.3.3 Spatial ontology rules

Spatial relation is one of the most important conceptual problems in the fields of spatial reasoning, Geographical Information System (GIS) and computer vision, as important as the spatial object itself. Spatial relation plays an important role in the process of spatial reasoning, spatial query, spatial analysis, spatial data modeling and map interpretation. Spatial relation is the relation between the objects with spatial characters. It usually consists of topological relations, metric relations and order relations. These relations are the bases of spatial data organizing, querying, analyzing and reasoning. All spatial entities are inherently related to some other spatial entity. Whether two entities intersect somehow or are thousands of miles apart, the relationship that they share can be described and evaluated.

Topological relationships 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 [113]. RCC-5, RCC-7, RCC-8, RCC-10 and RCC-13 are deduced by the RCC theory. The most commonly used form of this calculus is referred to as RCC-8 calculus and specifies the eight mutually exhaustive pairwise disjoint relations between region pairs, as shown in Figure 5.4 :

- DC(x, y) (x is disconnected from y) ;
- x = y (x is identical with y) ;
- PO(x,y) (x partially overlaps y) ;
- EC(x,y) (x is externally connected with y) ;
- TPP(x,y) (x is a tangential proper part of y) ;
- NTPP(x,y) (x is a non-tangential proper part of y) ;
- TPPi(x,y) (y is a tangential proper part of x) ;
- NTPPi(x,y) (y is a non-tangential proper part of x).

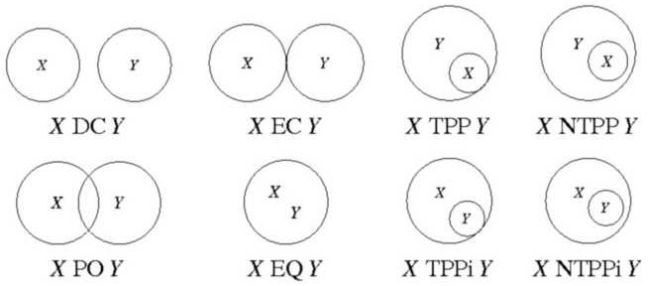

FIGURE 5.4 – Region connection calculus 8 relationships

The same set of eight geospatial topological relations is described with different names based on the Dimensionally Extended Nine-Intersection Model (DE-9IM). The DE-9IM representation was developed by Clementini and others [26, 27] based on the seminal works of Egenhofer and

others [36, 35]. These spatial relationships are used to test the existence of a specified topological spatial relationship between two geometries. OGC considers two kinds of spatial relationships :
– Topological relationships are shown in Figure 5.5 : `Equals, Within, Touches, Disjoint, Intersects, Crosses, Contains, Overlaps` and `Relate`;
– Functions for distance relationships : `Distance`.

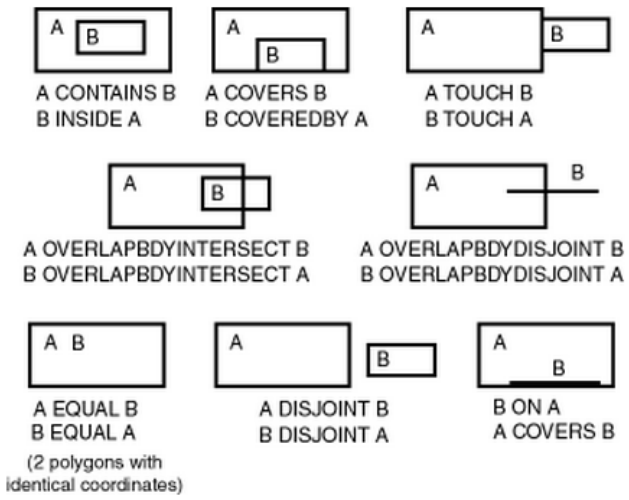

FIGURE 5.5 – OGC topological relationships

5.4 Conclusion

This chapter dedicated to the inferences in the ontology of semantic trajectories. Our objective was to interpret efficiently spatio-temporal data on the trajectory ontology. So, we detailed an inference mechanism to facilitate the extraction of knowledge. Inference rule and inference engine are parts of this mechanism. The language Semantic Web Rules Language (SWRL) is a language for the inference rules and is an indispensable ally of Web Ontology Language (OWL). We also illustrated the different operating models of an inference engine based on rules. Regarding the ontology of trajectory, three types of rules are associated to infer knowledge or, more precisely, explicit knowledge : 1) application domain rules (for example, deduct activities), 2) temporal rules based on Allen's relationships algebra, and 3) spatial rules based on the Open GIS Consortium (OGC) topological relationships.

We are about to implement the spatio-temporal trajectory ontology. To compute the ontology inference mechanism, it is necessary to implement the spatio-temporal rules. We will present more details in the next chapter.

Part III

Research design and implementation

Table of Contents

In this part, we specify our choice for implementing our framework. Then, we present the steps of the implementation of our trajectory modeling approach and trajectory ontology inference. Finally, we apply our model in a specific application domain.

In our implementation, we are based on RDF triple store. In Chap. 6, we detail the trajectory ontology RDF triple store. For this, we are based on Oracle Database Semantic Technologies for modeling, storing and querying. Based on that, we present the steps of our trajectory ontology implementation.

Then in Chap. 7, we introduce the trajectory ontology inference. We implement the trajectory ontology rules, domain, temporal and spatial rules. Then, we detail the mechanism of Oracle rule-based inference engine in the computation process and in the execution of rules. We show the complexity of this process, especially when using user-defined rules for temporal and spatial. Here, we introduce our contribution to reduce this complexity. Therefore, we design and implement some refinements over the ontology inference. For this enhancement, we address temporal and spatial neighbor refinements. Moreover, we present a refinement to reduce the inference repetition.

Overall, to evaluate the explained model, we need to apply it over an application domain. Then, our generic model is applied to a case of marine mammal trajectories, particularly seal trajectories in Chap. 8. We detail the application scenario to represent the seal trajectory model. To integrate it with our model, we transform it to a seal trajectory ontology modeling approach. We populate the final ontology in RDF triple store. To consider the knowledge of seal application, we implement the seal trajectory ontology rules and apply the inference over captured seal trajectory data. Related to the inference complexity problem, we design and implement a seal trajectory ontology inference refinement. We name it two-tier inference filters.

Chapter 6

Trajectory ontology RDF triple store

Contents

6.1 Introduction

In the past few years, there were many attempts to provide a database model for large graph data [90]. Research in graph databases fields was popular in the early 1990s with databases models like LDM [83], GOOD [51], O2 [84], and GraphDB [59]. The international conference GraphConnect [52], which first edition was held in 2012, is a major meeting for discussing graph databases models and their applications. The main-stream graph databases provide an object model for nodes and relationships. These graph databases focus on either RDF triplets, linked data, or relationships for storage. These databases often use direct memory links to adjacent nodes rather than requiring joins or keys lookups. Currently, data models focus more on providing an object-oriented, or relationship oriented and structure.

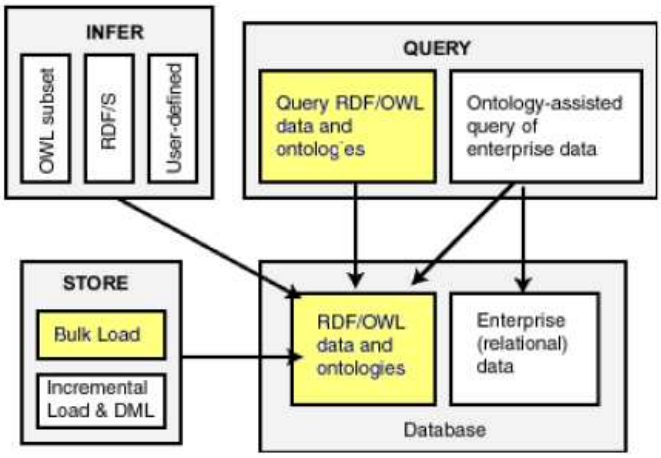

FIGURE 6.1 – RDF triple store capabilities

Here are some examples of graph databases. AllegroGraph [4] is a high-performance, RDF databases model. AllegroGraph supports SPARQL, RDFS++, and Prolog reasoning. DEX [90] is a very efficient, bitmaps-based graph database model written in C++. The focus of DEX is performance in the management of very large graphs, and even allows the integration of various data sources. FILAMENT [45] is a graph persistence library built on the top of PostgreSQL. This library allows querying graph data with SQL through JDBC. G-STORE [29] is a prototype query language and storage manager for large graphs. It is also built on the top of PostgreSQL.

In our work, we use Oracle RDF triple store. This technology has evolved in Oracle DBMS version 10g, 11g and takes the name of "Oracle Spatial and Graph - RDF Semantic Grap" [102] in Oracle DBMS version 12c. This system provides support for persistence, inference and querying ontologies through the implementation of RDF, RDFS and a large part of OWL standards [103]. The DBMS defines a core in its metabase to support technologies related to RDF data. In the following section, we detail the Oracle RDF triple store and then we use this technology for the implementation of our ontology .

6.2 Oracle RDF triple store

RDF Semantic Graph (Formerly Oracle Database Semantic Technologies) is a standards-based, scalable, secure, reliable and efficient RDF management platform. Based on a graph data model, RDF triples are persisted, indexed and queried, like other object-relational data. Figure 6.1 shows that RDF infrastructure contains semantic data and ontologies (RDF/OWL models), as well as traditional relational data.

In the following sections, we detail RDF data modeling. The way RDF data are stored, loaded and queried in a database.

6.2.1 RDF data model

RDF data model is a collection of conceptual tools for describing entities and the relationships among these entities and for modeling them in a database [121, 28]. Since Oracle database 10g

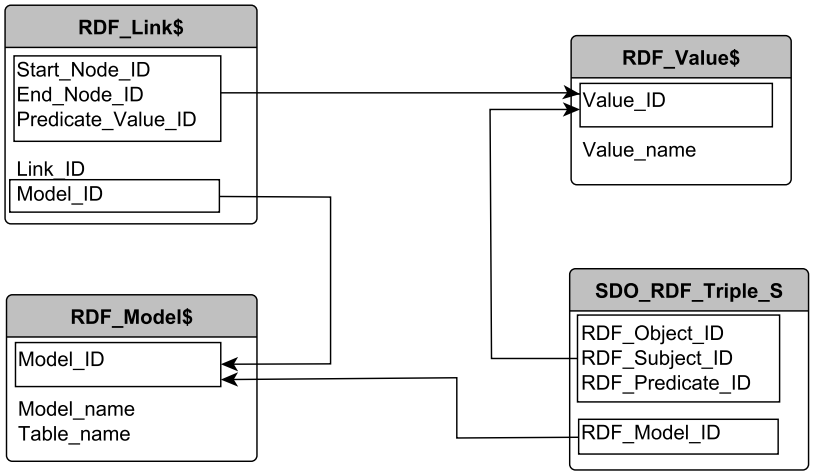

FIGURE 6.2 – Attributes of `SDO_RDF_TRIPLE_S` object type

Release 2, this model has been used for storing RDF and OWL data. This functionality builds on the top of Oracle Spatial Network Data Model (NDM). NDM is the Oracle solution for managing graphs within the Relational Database Management System (RDBMS) [129]. Then, RDF data has a simple data structure as a directed graph managed by NDM. This graph is implicitly defined by sets of triples [114, 63]. Subjects and objects of RDF triples are stored as start-nodes and end-nodes of a graph, respectively. Properties of triples are stored as directed links that describe relationships between the nodes. Each RDF triple is treated as one unique database object. A specified table is created to hold references to triples related to one model. In Oracle, two data types are defined for RDF data :

– The `SDO_RDF_TRIPLE` object type represents RDF data sets of triples (subject, predicate, object) ;
– The `SDO_RDF_TRIPLE_S` object type stores triples in database tables, (the `_S` for storage). Figure 6.2 shows the attributes of this type referring to tables under the MDSYS schema (the owner of spatial which is a part of interMedia). This type contains IDs of the subject, predicate and object of a triple referring to their text value saved in the table `RDF_Value$`. It also contains ID of the model of these triples referring to the model's name in the table `RDF_Model$`.

6.2.2 RDF data storage

An RDF triple is stored as one database object in a semantic data network. In this network, the subjects and objects of triples are mapped to network nodes, and the predicates are mapped to network links that have subject start-nodes and object end-nodes. The nodes are stored only once, regardless of the number of times they participate in triples. A new link is created whenever a new triple is inserted. RDF graph' data and metadata are stored in the system as entries in tables under the MDSYS schema. To store RDF data in the database :

1. A tablespace is recommended for all RDF data tables, since RDF data storage tends to be very large ;
2. A semantic network is created by the procedure `SEM_APIS.CREATE_SEM_NETWORK` to hold the RDF model ;

3. A table is created to store RDF triples. It is recommended that this table includes a column named `ID` of type `NUMBER` and a column named `Triple` of type `SDO_RDF_TRIPLE_S` ;
4. An RDF model is created by specifying a model name, the name of the table created before. This model is created by the `SEM_APIS.CREATE_SEM_MODEL` procedure.

6.2.3 RDF data loading

RDF data are loaded in an RDF triple store. To load RDF triples into a table, there are three options as following :
– Bulk loading is a highly optimized method for loading medium to large number (e.g., billions) of triples. RDF data are loaded into a staging table as a preparation for loading them into the created table ;
– Batch loading is an optimized method to handle loading a medium number (e.g., a few millions) of triples. Its advantage is that, unlike bulk loading, it does not require object values to stay within 4000 bytes ;
– Incremental loading via transactional SQL statement `INSERT INTO` is a recommended method for a small number (e.g., up to a few thousands) of triples.
For bulk loading and batch loading, only N-Triple [30] file-based input format is supported. Incremental loading requires use of an object type constructor, `SDO_RDF_TRIPLE_S`, with target RDF model name and lexical values for subject, predicate and object components of the triple used as arguments. Figure 6.3 is an activity diagram modeling the insertion of an RDF triple into a graph. The parts of a triple are inserted into the RDF_values$ table and then an entry is inserted into the RDF_link$ table.

6.2.4 Rules and rulebases

RDF triple store is a rulebased system. Each rulebase consists of a set of rules. Oracle supports RDF, RDFS, RDFS++, OWLSIF, OWLPrime, OWL2RL and OWL2EL rulebases [116]. The RDF and RDFS rulebases are created to add RDF support to the database. However, RDFS++ is a minimal extension to RDFS. OWLSIF [64] contains OWL with IF Semantic. Finally, OWLPrime supports full OWL capabilities, such as property characteristics, comparisons and restrictions, as well as class comparisons and expressions. Code 6.1 presents the RDFS/OWL vocabulary constructs included in OWLPrime rulebase. In our work, we use OWLPrime rulebase.

```
rdfs:subClassOf, rdfs:subPropertyOf, rdfs:domain, rdfs:range,
owl:SymmetricProperty, owl:TransitiveProperty, owl:inverseOf,
owl:equivalentClass, owl:equivalentProperty, owl:hasValue,
owl:someValuesFrom, owl:allValuesFrom, owl:differentFrom,
owl:FunctionalProperty, owl:InverseFunctionalProperty,
owl:disjointWith, owl:complementOf, owl:sameAs
```

Code 6.1 – RDFS/OWL Constructs Included in `OWLPrime` rulebase

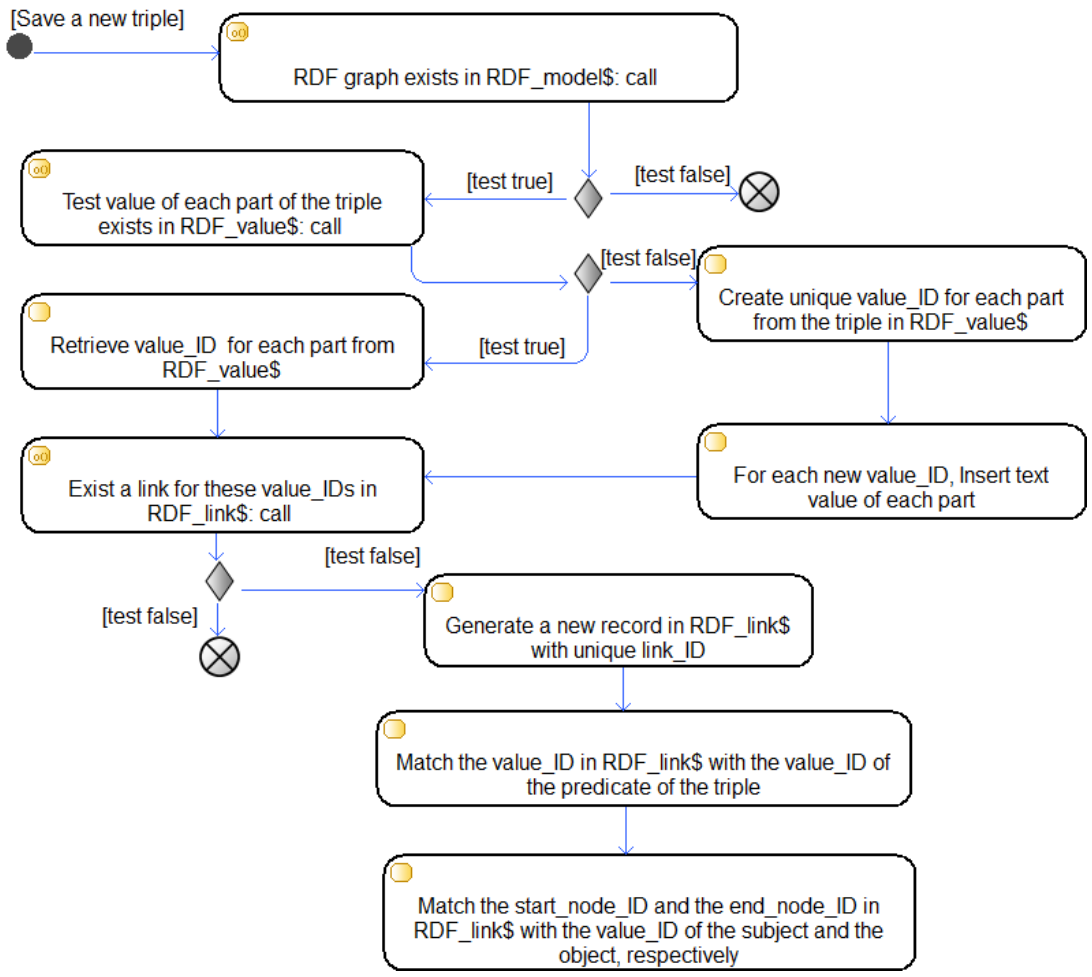

FIGURE 6.3 – Insert a new triple in an RDF triple store

6.2.5 User defined rules

Each rule in a rulebase is based on an `IF-THEN` pattern. A rule may yield additional triples when applied to an RDF model. Thus, the rule-based system is said to be a deduction system. In deduction systems, a rule has the three following components :
- `IF` pattern is the antecedent pattern as a subgraph to be matched in the data model ;
- `Filter` condition is an optional pattern as a boolean condition to be satisfied. This pattern typically involves variables from the antecedent ;
- `THEN` pattern is the consequence pattern describing the triples to be generated when the rule is fired.

A rule fires when its antecedent pattern and filter condition are both satisfied. The consequent pattern are the new triples generated from the firing rule. For each rulebase, a view, named `MDSYS.SEMR_rulebase-name`, is generated. Each rule is a row in this view. Users use this view to insert, delete, or update a rule. Table 6.1 gives the dictionary of this view for deduction-oriented antecedent-consequent rules.

TABLE 6.1 – `MDSYS.SEMR_rulebase-name` view dictionary

Column name	**Data type**	**Description**
Rule_Name	Varchar2 (30)	name of the rule
Antecedents	Varchar2 (4000)	IF side pattern for the antecedents
Filter	Varchar2 (4000)	filter condition that further restricts the subgraphs matched by the IF side pattern
Consequent	Varchar2 (4000)	THEN side pattern for the consequent
Aliases	SEM_ALIASES	one or more ontologies to be used

6.2.6 Rules index

A rules index (entailment) is an object containing pre-computed triples from applying a specified set of rulebases to a specified set of models. If a graph query refers to any rulebases, a rule index must exist for each rulebase-model combination in the query. To create an entailment, the `SEM_APIS.create_entailment` procedure must be used. Table 6.2 gives the parameters of this procedure. When a rule index is created, a view, called `MDSYS.SEMI_entailment-name`, is also created to include a row for each triple.

6.2.7 RDF data query

In order to query RDF triples, Oracle defines `SEM_MATCH` table function. This function searches for a given pattern into the RDF triples based on some rules. Table 6.3 gives the parameters of this function. The `query` attribute is a string literal with one or more triple patterns, usually containing variables. The `models` attribute identifies the RDF models to use. The `rulebases` attribute identifies rulebases for the considered models. The `aliases` attribute identifies ontologies. The `filter` attribute identifies any additional selection criteria. Finally, the `SEM_MATCH` table function returns an object of type `anydataset` [15].

15. Anydataset can be seen as an array of elements that shares the same data type. The data type can be an Oracle built in data type or a user defined one

TABLE 6.2 – Parameters of the `SEM_APIS.create_entailment` procedure

Column name	Data type	Description
entailment_Name	Varchar2	name of the entailment
Models	`SEM_Models`	one or more model names
Rulebases	`SEM_Rulebases`	one or more rulebase names
Passes	Number	the number of rounds that the inference engine should run, default `SEM_APIS.REACH_CLOSURE`
Inf_Components	Varchar2	represent inference components, default NULL
Options	Varchar2	options to control the inference process

TABLE 6.3 – Attributes of the `SEM_MATCH` function

Column name	Data type	Description
Query	Varchar2	one or more triple patterns
Models	`SEM_Models`	one or more model names
Rulebases	`SEM_Rulebases`	one or more rulebase names
Aliases	SEM_ALIASES	one or more ontologies to be used
Filter	Varchar2	any additional selection criteria

6.3 Ontology implementation

Implementing our approach includes the creation the declarative and the imperative parts of the trajectory, spatial and time ontologies. In Oracle RDF data Store, we have to :

1. Prepare the creation of the ontologies ;
2. Create the declarative parts of the ontologies ;

6.3.1 Preparation of the ontologies

Code 6.2 refers to the preparation for storing triples related to the trajectory, spatial and time ontologies, explained as follows :

1. Lines 1 to 5 create a tablespace, named `tbl_space` ;
2. Line 7 creates a semantic network for the tablespace created ;
3. Lines 9 to 11 create a semantic data table for each ontologies. `owlSemanticTrajectory_data` is a table for the trajectory ontology ;
4. Lines 13 to 15 create a semantic model for each ontology referring to the created table and to its column `triple`.

```
CREATE BIGFILE TABLESPACE tbl_space
   DATAFILE '/data/oradata/tbl_space.dbf'
   SIZE 1024M REUSE AUTOEXTEND ON
   MAXSIZE UNLIMITED
   EXTENT MANAGEMENT LOCAL UNIFORM SIZE 64M;

EXECUTE SEM_APIS.CREATE_SEM_NETWORK ('tbl_space');

CREATE TABLE owlSemanticTrajectory_data    (id NUMBER, triple SDO_RDF_TRIPLE_S);
CREATE TABLE owlTime_data                  (id NUMBER, triple SDO_RDF_TRIPLE_S);
CREATE TABLE owlOGCSpatial_data            (id NUMBER, triple SDO_RDF_TRIPLE_S);

EXECUTE SEM_APIS.CREATE_SEM_MODEL('owlTrajectory','owlSemanticTrajectory_data', 'triple');
EXECUTE SEM_APIS.CREATE_SEM_MODEL('owlTime',      'owlTime_data',               'triple');
EXECUTE SEM_APIS.CREATE_SEM_MODEL('owlOGCSpatial','owlOGCSpatial_data',         'triple');
```

Code 6.2 – Preparation to create `owlSemanticTrajectory`, `OWL-Time` and `owlOGCSpatial` ontologies

6.3.2 Creation of the declarative parts of the ontologies

To create the declarative parts of the trajectory, time and spatial ontologies, we implement the concepts and their relationships provided by Protégé tool. Protégé [16] is a system for creating ontologies. It is very popular in the field of Semantic Web and the level of research in computer science. Protégé is developed in Java. It can read and save ontologies in most formats : RDF, RDFS, OWL, etc.. We visualize the ontology schema by ProtégéVOWL [17]. ProtégéVOWL implements the Visual Notation for OWL Ontologies (VOWL) by providing graphical depictions for elements of the Web Ontology Language (OWL).

We implement the concepts and the relationships for the `owlSemanticTrajectory` ontology presented in Sect. 4.3.4, the `OWL-Time` ontology introduced in Sect. 4.4.1 and the `owlOGCSpatial` ontology explained in Sect. 4.4.2. We save the resource of the ontologies in N-Triple format. In general, we use bulk loading form to load the ontology into the suitable database table. We load the declarative parts of the `owlSemanticTrajectory` ontology in the `owlSemanticTrajectory_data` table, the `OWL-Time` ontology in the `owlTime_data` table and the `owlOGCSpatial` ontology in the `owlOGCSpatial_data` table.

6.4 Matching the ontologies

Matching ontologies is fundamental while considering different and independent sources of knowledge, like in our case the trajectory, time and spatial ontologies. OWL language defines the ontology-import `owl:imports` construction to import additional ontologies. OWL language can connect concepts of two ontologies using the built-in property `owl:equivalentClass`, and connect properties using the built-in property `owl:equivalentProperty`. The construction `owl:equivalentClass` states that the two classes involved have the same class extension (i.e., both class extensions contain exactly the same set of individuals). The `owl:equivalentProperty` states that the two properties have the same property extension.

16. http ://protege.stanford.edu/
17. http ://vowl.visualdataweb.org/

6.4.1 Matching trajectory and time ontologies

The ontology `owlSemanticTrajectory` imports the ontology `OWL-Time`. The concept `sequence` is an equivalent class to the concept `properInterval` from `OWL-Time`. The `startDate` and `endDate` properties of a sequence are equivalent properties to the `hasBeginning` and `hasEnd` properties from `OWL-Time`. Figure 6.4 shows the mapping defined between the trajectory and time ontologies according to our requirements.

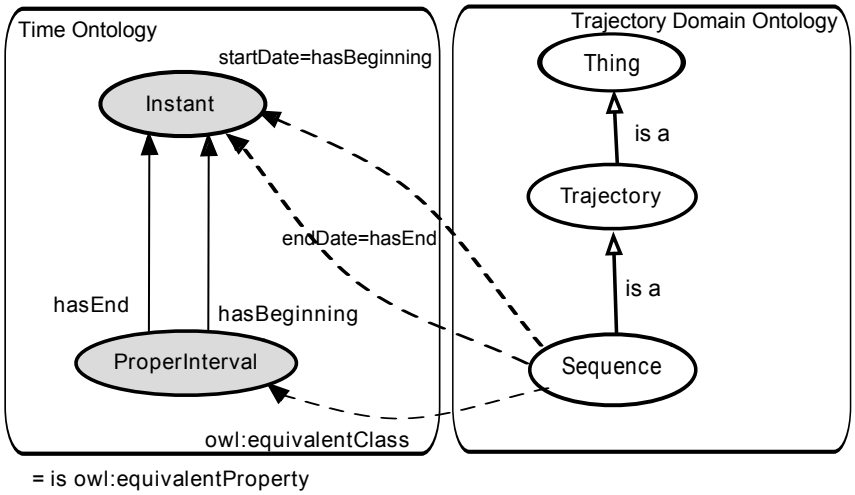

FIGURE 6.4 – Trajectory and time ontologies mapping

6.4.2 Matching trajectory and spatial ontologies

We defined the trajectory domain ontology (Figure 4.8) with the following considerations :

– `Sequence` : can be considered as a spatial line `owlOGCSpatial:Line` ;
– `Position` : can be considered as a spatial point `owlOGCSpatial:Point`.

To meet these requirements, we need to connect the trajectory and the spatial ontologies. As these ontologies are formalized in OWL, we use this language to define the needed matching. The ontology `owlSemanticTrajectory` imports the ontology `owlOGCSpatial`, so all statements of the latter are visible and available for the first ontology. Figure 6.5 shows the mapping defined between the trajectory domain and spatial ontologies according to our requirements. In this mapping, the concept `Sequence` is an equivalent class to the concept `Line` from `owlOGCSpatial`. The concept `Position` is an equivalent class to the concept `Point` from `owlOGCSpatial`. The `startPosition` and `endPosition` properties of a sequence are the beginning and ending points, respectively.

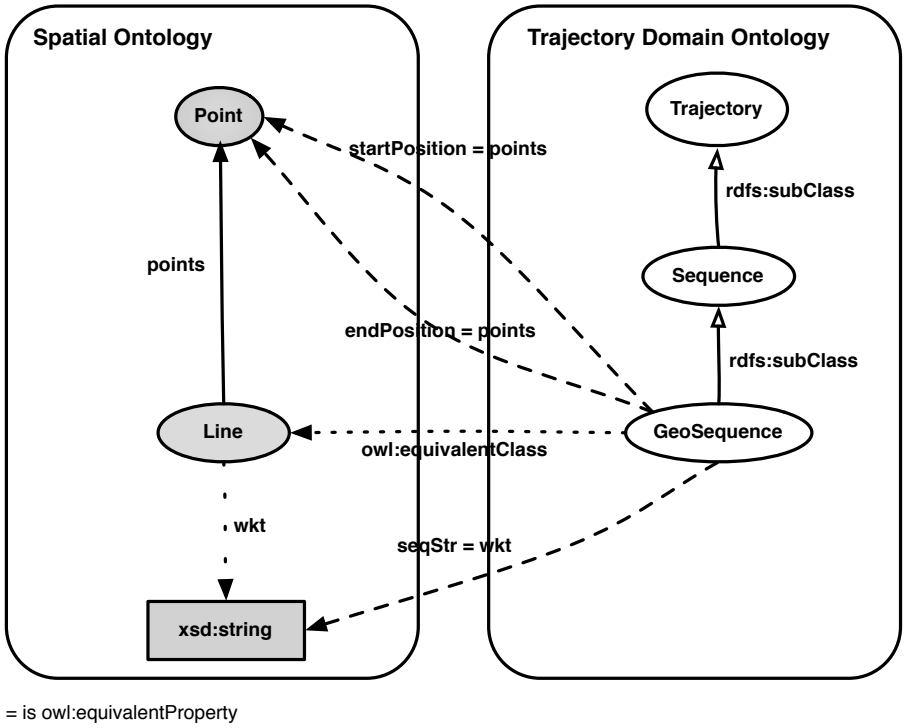

FIGURE 6.5 – Trajectory and spatial ontologies mapping

6.5 Conclusion

In this chapter, we implemented our trajectory ontology modeling approach. This implementation is based on RDF triple store and detailed the RDF triple store in Oracle Database Semantic Technologies. This system, which provides support for storage, inference and querying ontologies transformed into RDF triples, is available from the 10g version of Oracle software. It also has the advantage of being a system based inference rules. We prepared and made the implementation of the declarative parts of the three ontologies in RDF triple store Oracle : trajectory, time and spatial. The concepts and relationships defined were then transformed into RDF triples and finally loaded into the corresponding tables in the Oracle RDF triple store. Considering different and independent ontologies, matching ontologies was fundamental. The matching between the trajectory ontology with time and spatial reused ontologies was performed using OWL axioms of equivalence.

Chapter 7

Trajectory ontology inference

Contents

7.1 Introduction

In this chapter, we start prepare the heart of the inference mechanism over our framework. We implement the trajectory ontology rules : Allen's temporal rules ; and OCG topological spatial rules. Our objective is understand how the inference engine rule-based Oracle managed by the system. Then, the Oracle inference engine computed over the ontology rules. We detail how the rules are executed by the inference. After many experiments, we address the complexity of the inference mechanism over user rules. We address some enhancements over the inference computation : neighbor, repetition refinements.

7.2 Trajectory ontology rules

Each rule has declarative and imperative parts in the ontology. These sets of rules will directly influence the ontological inference process used by the query engine.

7.2.1 Time ontology rules

We implement the rule base `owlTime_rb` to hold the temporal interval relationships. These relationships are : `intervalEquals`, `intervalBefore`, `intervalDuring`, `intervalOverlaps`, `intervalOverlappedBy`, `intervalStartedBy`, `intervalFinishedBy`, `intervalFinishes`, `intervalContains`, `intervalMetBy`, `intervalMeets`, `intervalStarts`, and `intervalAfter`. Each relationship has a declarative part as an RDF `ObjectProperty` and an imperative part, formally, an associated rule as an `IF-THEN` pattern. Appendix 1 presents the implementation of all temporal rules. Figure 7.1 shows the declarative part of the `intervalAfter` rule. This rule is an object property between two proper intervals. Code 7.1 is the imperative part of the rule. It is based on operations defined in the table `TM_RelativePosition` of the ISO/TC 211 specification about temporal schema [72], detailed in Sect. 2.2.2. In Code 7.1, the line 7 expresses the condition :

$$self.begin.position > other.end.position$$

where

$$\begin{cases} self & = & tObj2 \\ other & = & tObj1 \\ self.begin.position & = & beginTime2 \\ other.end.position & = & endTime1 \end{cases}$$

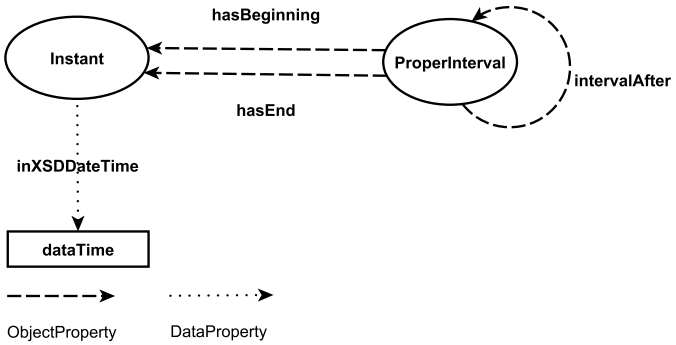

FIGURE 7.1 – Declarative part of `intervalAfter` rule

```
1 EXECUTE SEM_APIS.CREATE_RULEBASE('owlTime_rb')
2 INSERT INTO mdsys.semr_owltime_rb
3 VALUES('intervalAfter_rule',
4 '(?tObj1 rdf:type ot:ProperInterval)      (?tObj2 rdf:type ot:ProperInterval        )
5  (?tObj1 ot:hasEnd ?end1          )      (?end1 ot:inXSDDateTime ?endTime1         )
6  (?tObj2 ot:hasBeginning ?begin2  )      (?begin2 ot:inXSDDateTime ?beginTime2     )',
7 '(beginTime2 > endTime1                                                          )',
8 '(?tObj2 ot:intervalAfter ?tObj1                                                 )',
9 SEM_ALIASES(SEM_ALIAS('ot','http://www.w3.org/2006/time#'))                      );
```

Code 7.1 – Implementation of `intervalAfter rule`

7.2.2 Spatial ontology rules

In this thesis, we consider spatial topological relationships. These relationships are : `Equals, Within, Touches, Disjoint, Intersects, Crosses, Contains, Overlaps` and `Relate`. Each relationship has a declarative part as an RDF `ObjectProperty` and an imperative part, formally, an associated rule as `IF-THEN` pattern. We create a rulebase named `owlOGCSpatial_rb` to hold spatial relationships rules. Appendix 3 presents the implementation of all spatial rules. Figure 7.2 shows the declarative part of the `Contains` rule. This rule is an object property between two geometries. Code 7.2 presents the imperative part of the spatial `Contains` rule. Lines 4 to 6 represent the `IF` part of the rule. We construct a subgraph and necessary variables, namely, the two spatial objects `sObj1` and `sObj2` their strings coordinates, respectively, `wktSObj1` and `wktSObj2`, and the `srid` which is the Spatial Reference System Identifier. Line 9 is the `Filter` part of the rule which evaluates the spatial relationship between the two spatial objects using a function called `evalSpatialRelationship`. This function builds a bridge between the ontology spatial rules and the spatial operators in Oracle Spatial DBMS. Line 10 in Code 7.2 is the consequent or the `THEN` part of the rule.

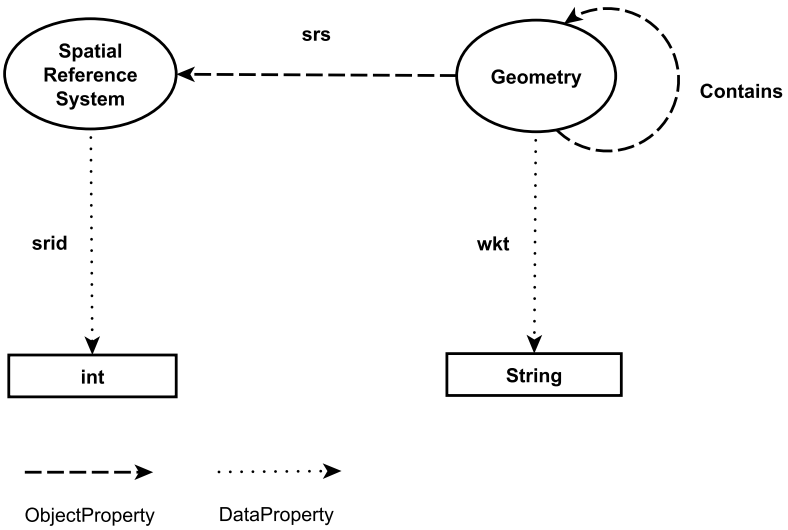

FIGURE 7.2 – Declarative part of `Contains` rule

```
1  EXECUTE SEM_APIS.CREATE_RULEBASE('owlOGCSpatial_rb');
2  INSERT INTO mdsys.semr_owlOGCSpatial_rb
3  VALUES('contain_rule',
4  '(?sObj1      rdf:type      os:Geometry            )  (?sObj2 rdf:type     os:Geometry )
5   (?spaRefSys1 rdf:type      os:SpatialReferenceSystem)  (?sObj1 ?     os:srs ?spaRefSys1   )
6   (?spaRefSys2 rdf:type      os:SpatialReferenceSystem)  (?sObj2 ?     os:srs ?spaRefSys2   )
7   (?spaRefSys1 os:srid       ?srid                  )  (?spaRefSys2  os:srid ?srid        )
8   (?sObj1      os:wkt        ?wktSObj1              )  (?sObj2 os:wkt      ?wktSObj2     )',
9  '(evalSpatialRelationship(sObj1,wktSObj1,sObj2,wktSObj2,srid,''CONTAINS'' ) = 1          )',
10 '(?sObj1      os:contains   ?sObj2                                                       )',
11 SEM_ALIASES(SEM_ALIAS('os','http://l3i.univ-larochelle.fr/owlOGCSpatial#'))              );
```

Code 7.2 – Implementation of `Contains rule`

Figure 7.3 illustrates the functionality of the `evalSpatialRelationship` during the execution of the eight spatial rules over spatial objects. For every two spatial objects, the inference procedure executes spatial rules. The function `evalSpatialRelationship` calls the corresponding

Oracle spatial operator for the current running spatial rule, as in Table 7.1. From the inference computation, a new relationship is computed and saved as a new inferred triple.

TABLE 7.1 – `evalSpatialRelationship` connects spatial ontology rules to Oracle spatial operators

`evalSpatialRelationship` **bridge**	
Ontology spatial rule	**Oracle spatial operator**
Equals_rule	SDO_Equal
Contains_rule	SDO_Contains
Overlaps_rule	SDO_Overlaps
Touch_rule	SDO_Touch
Covers_rule	SDO_Covers
CoveredBy_rule	SDO_CoveredBy
AnyInteract_rule	SDO_AnyInteract
Inside_rule	SDO_Inside

7.3 Oracle rule-based inference engine

Reasoners, that support DL-safe rules such as Oracle engine, can be used for inference and consistency checking over spatio-temporal relationships. The Oracle reasoner is applied to an ontology schema to infer additional facts using OWL semantics. The Oracle inference system adopts a forward-chaining strategy. From inference results, Oracle supports a set of graph queries over an original and inferred data.

7.3.1 Inference engine entailment

The inference process starts with creating an entailment through the `CREATE_ENTAILMENT` procedure [103] to perform inference against a set of semantic models and a set of rulebases. This entailment can be used with user-defined rulebases and predefined rulebases. Intuitively, an entailment means that inferred data are a logical consequence of an input semantic model and a specified set of rulebases. Code 7.3 creates an entailment using the trajectory, temporal and spatial models. The `USER_RULES=T` option is required while applying user-defined rules.

```
1 SEM_APIS.CREATE_ENTAILMENT('owlTrajectory_idx',
2 SEM_MODELS('owlTrajectory','owlTime','owlOGCSpatial'),
3 SEM_RULEBASES('OWLPrime',owlTime_rb,'owlOGCSpatial_rb'),
4 SEM_APIS.REACH_CLOSURE,
5 NULL,
6 'USER_RULES=T');
```

Code 7.3 – Entailment over the trajectory domain, temporal and spatial models

7.3.2 Inference process computation

The inference process consumes a huge amount of time to be computed over ontologies and the users' rules. The objective is to understand the way the inference computes and the time taken.

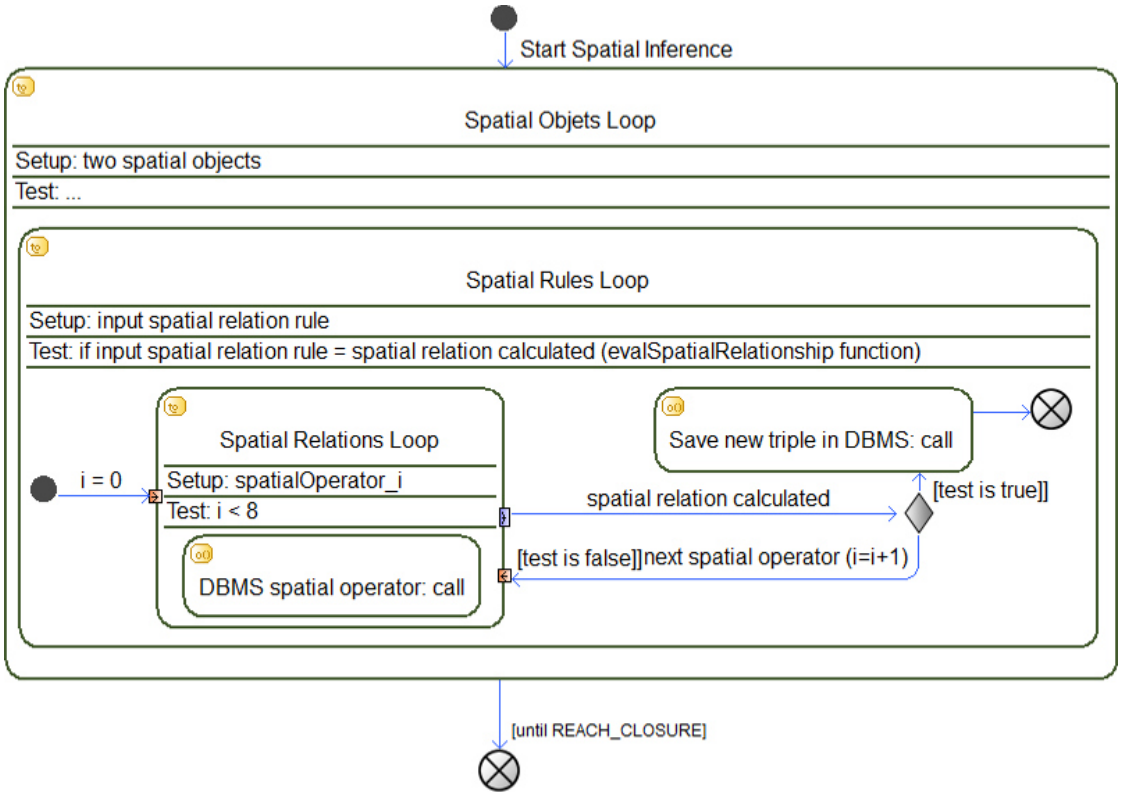

FIGURE 7.3 – Activity diagram for spatial inference process

The first solution is to generate a proof, which is a list of triples used to derive a new inferred triple. However the proof can not be generated while using users' rules. The second solution is the `AUDIT_SYS_Operations` which enables the auditing of operations issued by the user.

From the auditing, we found the following steps which detail The Oracle inference engine work [139]. Figure 7.4 is the activity diagram of the ontology inference process in Oracle.

- The inference process starts with the creation of a new partition in the semantic data table and the creation of a temporary table to hold new inferred triples. An exchange table is created with the same structure and the same index as the semantic data table ;
- The inference process selects the following information : the semantic data model (ID, name), the values in RDF_values$ table, the RDF_rulebase$ (ID, name) and the rules in the selected rulebases from RDF_rules$;
- The core inference logic is driven by a set of inference rules. Each rule will be executed during the inference process and only new triples are added into the temporary table. The inference process has many passes. In one pass, all rules will be executed ;
- When the rules no longer generate any new triple, the inference process terminates by copying out all triples from the temporary table into the exchange table ;
- Exchanging data from the exchange table into the newly created table partition in the semantic data table.

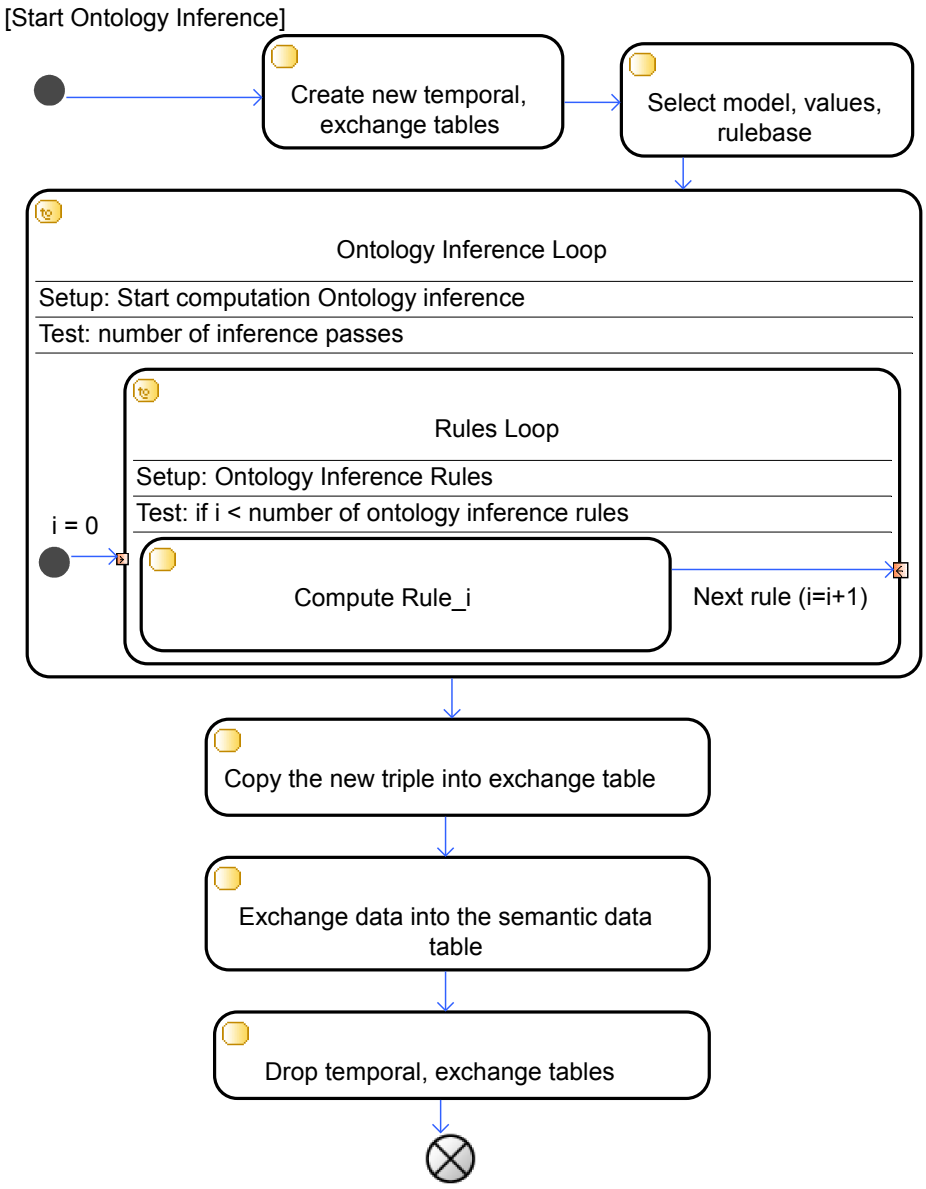

FIGURE 7.4 – The ontology inference process in Oracle

7.3.3 Inference rule execution

Rules in Oracle can be divided into three categories :

1. Axioms are rules that have no antecedent ;
2. 1-shot rules have an antecedent, but no variables in the consequent ;
3. General rules have an antecedent and have variables in the consequent.

In the computation time, each user-defined rule is automatically translated into an SQL query statement. This SQL query is executed on the semantic data table. For each rule antecedent pattern, the engine searches for matched triples in the semantic data table. The matched triples are joined based on common variables which are in the rule's filter pattern. Finally, the rule engine adds new triples related to the consequent pattern. Figure 7.5 is the activity diagram of execution of an ontology inference rule in Oracle.

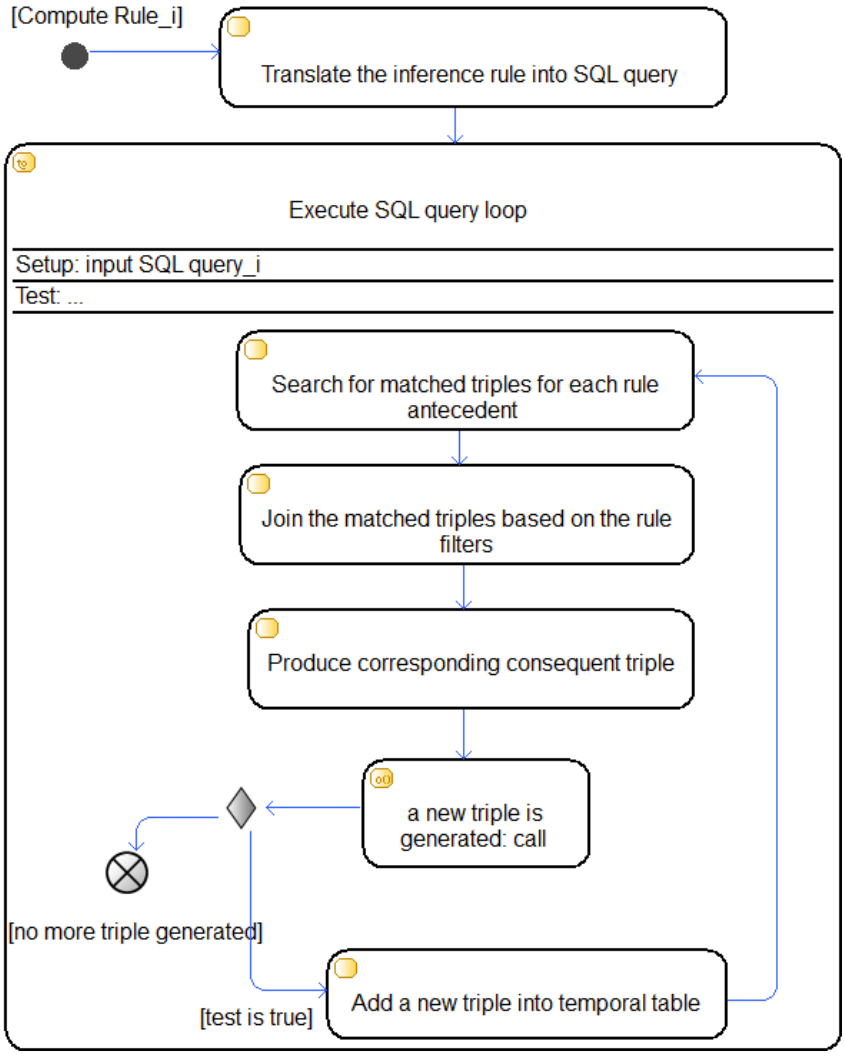

FIGURE 7.5 – Execution of an ontology inference rule in Oracle

The set of results from the inference is fully materialized in order to support graph queries over an original and inferred data.

7.4 RDF triple store inference and complexity

The inference mechanism is computed over the model, rules and loaded triples. Users query these triples and the inferred one with a `SEM_MATCH` query language. When new triples are loaded, users are no longer able to run any query neither over the new data, nor over the previous inferred results. In this case, the inference mechanism should be recomputed again over the new data as well over the previous data. This is because the Oracle inference mechanism is not an incremental process, especially with user-defined rules. However, some OWL constructs are not very suitable for inferencing and the user may need reasoning at query time. However, Oracle, version 11G, does not support any query at the computation time.

In the Oracle Semantic data store, any trial to add a new triple manually in the database's tables will break the validity of the database. Then, the user should delete the database and start again. However for this case, OpenLink's Virtuoso and BigOWLim inference engines take in charge additional inferences and facts while answering a user query.

7.4.1 User-defined rule inference

Supporting user-defined rules, Oracle Semantic data store has limitations on supporting efficient options in this case :

1. The `INC` option enables incremental inference taking into account the inference of the previous step. This option saves index execution and time. However it is not available with user-defined rules ;
2. The `REACH_CLOSURE` option is the default value for the number of rounds for the engine. It must be respected in the case of user-defined rules ;
3. The `DISTANCE` option generates additional distance information that is useful for semantic operators. It is also not available with user-defined rules.

Finally, we have to mention that the Oracle inference engine does not integrate the OWL and user-defined rule inference components similarly. Behind the scene, Oracle first runs the OWL rules till a closure occurs. Then, similarly Oracle runs user-defined rules till a closure occurs, and repeats the whole process till no new triple is generated from either component. Moreover, the inference time will increase as vocabulary becomes increasingly expressive, especially when using OWLPrime with user-defined rules. This is because the inference time depends on the complexity of the rules in a vocabular [139].

7.4.2 Rule execution complexity

Logically, we represent a user-defined rule as L and a set of rules as $\{L_1, L_2..., L_u\}$. From IF-THEN pattern, we can say that a rule is represented as the following :

$$L^{\mathcal{I}} \subseteq (Antecedent \times \{Filter\} \times Consequent) \tag{7.1}$$

Considering the way the inference engine works, we try to specify the complexity of execution an inference rule. While the parts of a rule are patterns, we represent a pattern as P. The representation of the number of patterns in one antecedent in a rule L_j is $\#(P_{L_j})$. For an

ontology G in a domain D_i, we represent the number of ontology patterns as $\#(P_G)$. Considering the number of cycles, the inference engine runs to execute the rules, we represent the number of cycles as N. We represent Computing Running Consumption of rules as CRC. Then, Computing Running Consumption of a rule L_j as CRC_{L_j} is in Equation 7.2 :

$$CRC_{L_j} = N * \left[\#(P_{L_j}) * \#(P_G)\right] \tag{7.2}$$

The worst case is when the number of rule's patterns are equal to the ontology patterns $\#(P_{L_j}) = \#(P_G)$ in Equation 7.3,

$$CRC_{L_j} = N * \left[\#(P_G)^2\right] \tag{7.3}$$

Computing running consumption of U ontology rules in an ontology G is in Equation 7.4 :

$$CRC = N * \sum_{j=1}^{U} \left[\#(P_{L_j}) * \#(P_G)\right] \tag{7.4}$$

7.5 Ontology inference refinements

The inference mechanism computes relationships between all pairs of data and annotates them with activities of moving objects. This mechanism is needed for queries on the spatio-temporal trajectory ontologies. Our objective is to enhance the inference mechanism as mush as possible, particularly, when using user-defined rules.

7.5.1 Neighbor refinement

The spatial and temporal ontology rules are computed redundantly to compute the spatial and temporal relationships between objects during the inference mechanism. Then, we tackle the complexity problem of the computation of the inference. To reduce the computation of rules in the inference mechanism, we limit the size of the data considered in the computation into the needed ones.

To enhance the inference process, the user can define, for example, domain restrictions and constraints to limit the computation in a useful way for their work's objective. These limitations can be directional considering objects in the same direction or can be constrained by distance considering a specific distance between objects. The user also can apply restrictions related to the type of the considered objects.

To enhance the computation of rules in the inference mechanism, we define a refinement called **neighbor inference**. Over this refinement, the inference mechanism is computed just between neighbor objects.

7.5.1.1 Temporal neighbor refinement

Calculating the inference between all sequences of trajectories considering all time rules takes a huge amount of time and space storage capacity. To enhance the inference mechanism, we define a refinement called **temporal neighbor inference** : "*A temporal neighbor is when a sequence happened within a conceptual distance to another sequence*". The goal of this refinement, algorithm 1, is to consider the distance between two sequences in order to calculate the corresponding temporal relationships. This refinement takes all sequences (L_Seq) and the considered temporal neighbor number (TN). The function *distance* measures the distance between the two considered sequences and return a number. We apply this refinement over the implementation of the 13 temporal rules. Appendix 2 presents the implementation of all the refinement temporal rules. Then, when computing the inference, the temporal rules carry out this refinement. It is still difficult to determine the best candidate for the temporal neighbor distance.

```
   input  : List of sequences : L_Seq
   input  : A tempotral neighbour : TN
1  for S_r, S_a ∈ L_Seq do
2  |  if distance (S_a, S_r) ≤ TN then
3  |  |  calculate the temporal rule between S_r and S_a;
4  |  end
5  end
```

Algorithm 1: Temporal neighbor refinement algorithm

7.5.1.2 Spatial neighbor refinement

The spatial neighbor refinement limits the computation of the inference to the objects located in a specified area. So, we call this refinement *Area of interest*. The area of interest refinement is given by Algorithm 2, considering all spatial objects (L_SP) in our model and an area of interest ($area$). It considers the eight topological spatial relationships. This algorithm starts with two spatial objects (S_r, S_a). The algorithm checks if these two objects belong to the interested area by the Oracle function `SDO_WITHIN_DISTANCE`. Then, the result of this function allows the computation of the spatial relationship between the two considered objects. If the two objects do not belong to this area, the algorithm goes for another spatial candidates.

```
   input  : List of spatial objects : L_SP
   input  : An interested area : area
1  initialization;
2  for S_r, S_a ∈ L_SP do
3  |  if SDO_WITHIN_DISTANCE (S_r, S_a, area) then
4  |  |  calculate spatial relationship between S_r and S_a;
5  |  end
6  end
```

Algorithm 2: Area of interest refinement algorithm

We modify our function `evalSpatialRelationship` 7.3 between the spatial rules and Oracle spatial operators to consider this refinement. Figure 7.6 shows the spatial inference process considering neighbor refinement. The first step of this figure is checking if the two considered objects belong to the area of interest. In this refinement, the optimization coefficient for the interested area depends on an application domain and can be estimated after considering the statistical data dispersion.

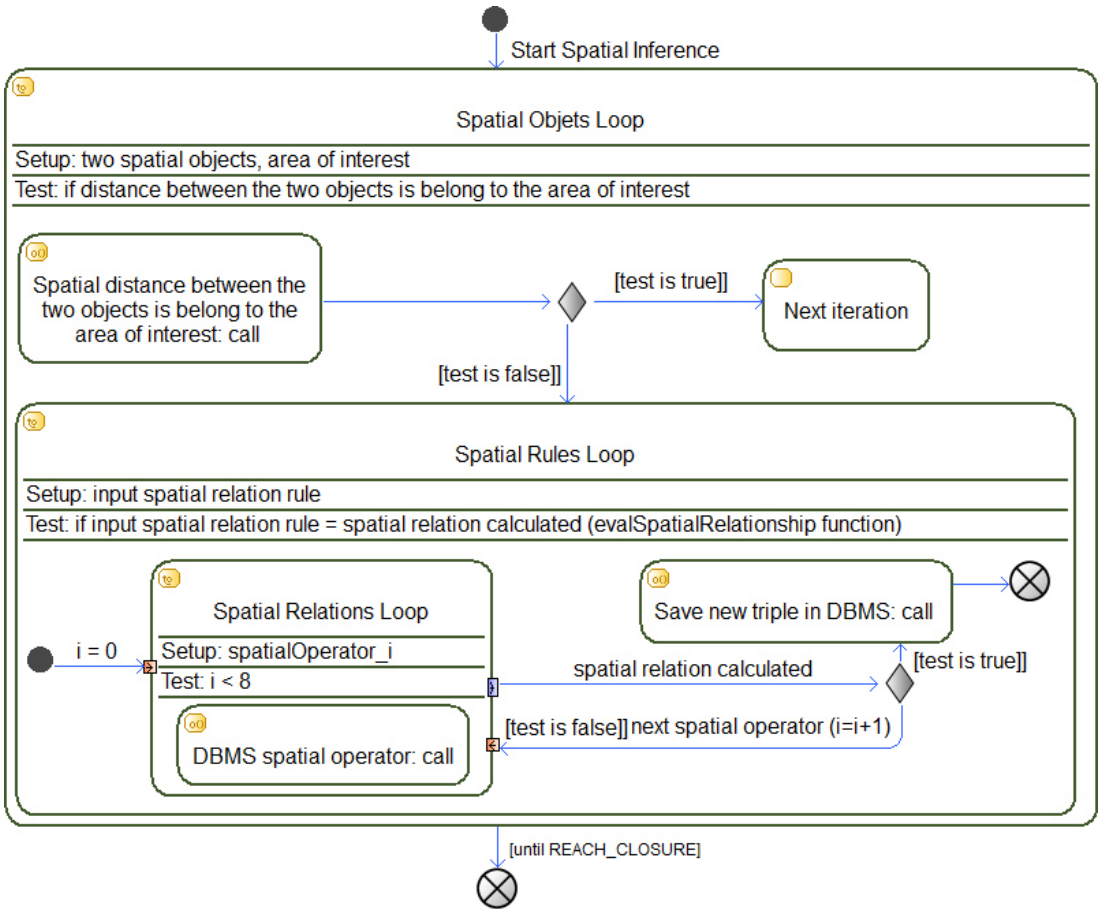

FIGURE 7.6 – Spatial inference process considering neighbor refinement

7.5.2 Inference repetition refinement

We mention that the inference engine has many computation cycles. In Oracle, we mention the `REACH_CLOSURE` problem in the case of applying user-defined rules. This number of cycles to reach the closure is a part of the computation complexity. To control this number of cycles, we define an inference refinement called `Passes refinement`. In this refinement, we limit this repetition into one pass, however, we keep the quality of the results. The idea is to save the inference results in a database from the first pass and use these results for the other passes, so

that they are not computed again. Algorithm 3 illustrates the passes refinement. This algorithm considers all spatial objects (L_SP) and the eight spatial relationships. The algorithm checks an existing spatial relationship between the two considered objects in the database. When there is no relationship, this means that the inference passes for the first time. The inference process will be computed normally and its results will be saved in the database. In the other case where the inference was already performed before, the algorithm passes to the next two objects. We modify our function `evalSpatialRelationship` 7.3 to consider this refinement and to optimize the passes of the engine during the computation of the inference. Figure 7.7 shows the spatial inference process considering passes refinement.

```
   input  : List of spatial objects : L_SP
1  initialization;
2  for S_r, S_a ∈ L_SP do
3  |   if Spatial relationship between (S_r, S_a) ∉ database then
4  |   |   Res := calculate Spatial relationship between S_r and S_a;
5  |   |   Save Res in the database ;
6  |   end
7  end
```

Algorithm 3: Passes refinement algorithm

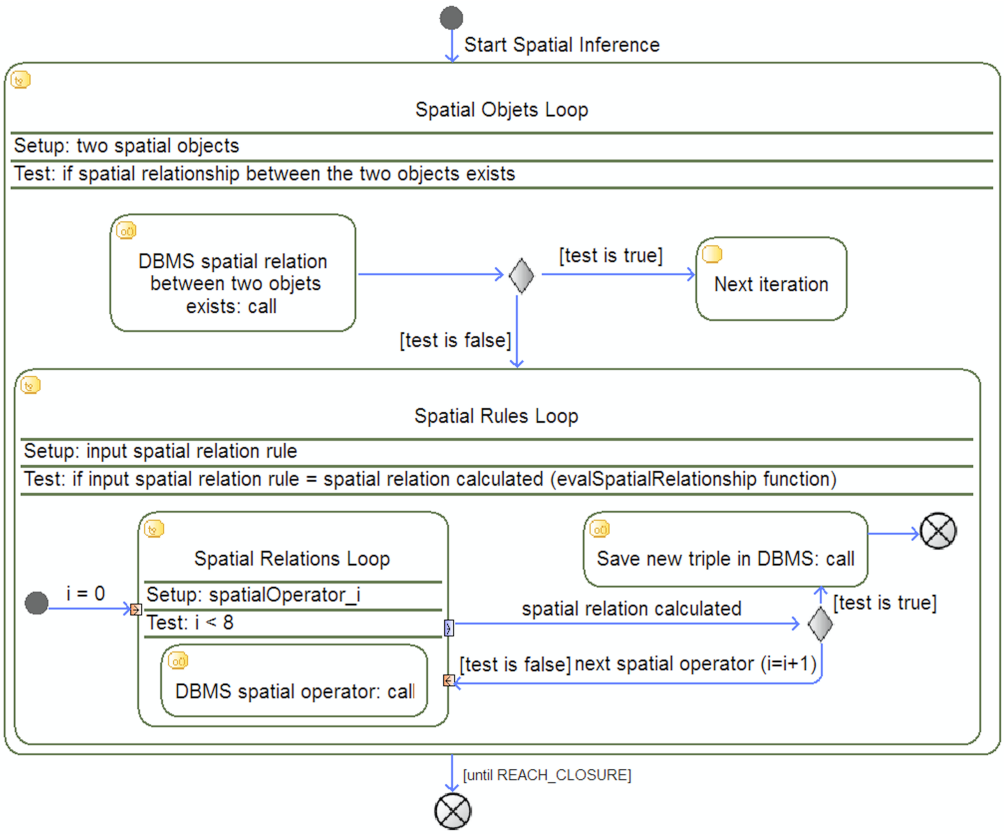

FIGURE 7.7 – Spatial inference process considering passes refinement

7.6 Conclusion

In this chapter, we prepared the application of the inference mechanism over our framework. We implemented the trajectory ontology rules. Allen's temporal rules and OCG topological spatial rules are considered. Then, the Oracle inference engine computed over the ontology rules.

We focused on how the inference engine rule-based Oracle which over the ontology rules managed by the system. We addressed rule execution complexity. Then, we tried to understand technicality the work of the inference engine. We applied the inference mechanism using different engines. Moreover, we performed experiments to understand the way the inferred data is separated and saved in the database. The difficulty was when we manually tried to add data into the database as inferred data, the whole database becomes invalid, then we had to delete it and rebuilt it again. Besides the fact that Oracle supports the inferences of OWL, we highlighted the system limitations on inference rules defined by the user. The Oracle inference mechanism is not an incremental process when using user-defined rules. For this problem, in the step N, we passed to the inference engine new data with the previous inferred data from the step N-1. However, the engine recomputed the inferred data (step N-1) in all the cases. Given these obstacles related to both Oracle and inherent rules of inference, tackle the inference computation complexity. We designed and implemented two types of refinements to enhance the inference computation. Temporal and spatial neighbor refinements, which are to enable the inference rules only temporal or spatial for trajectories close in distance, were defined. Moreover, we presented a refinement to reduce the inference repetition which is made to avoid repeating ontology rules unnecessarily.

Chapter 8

Application : Seal trajectories

Contents

8.1 Introduction

We presented our generic trajectory model in Chap. 4. We showed how this can serve as a high-level model to take into account a domain application. In this chapter, we discuss the domain application "marine mammal". This generic model is applied to the case of marine mammal trajectories. The context is the following. Seals are observed equipped with GPS sensors, temperature and depth. Biologists want to know how seals use the marine environment for food. The sensors in the presence identify three types of state : out of the water, cruise, dive.

8.2 Seal trajectory model

Today's advances in sensors technological have enabled capturing events with a better accuracy. In our study, the information collected onboard by the tag include a GPS location, the environment (air or water), the sea temperature and the dive depth when the tag is below the sea surface. The collected data increase exponentially, analyzing data in an optimized way therefore becomes more complex.

8.2.1 Application scenario

We consider trajectories of seals, Figure 8.1. The data comes from the LIENSs [18] (CNRS/University of La Rochelle) in collaboration with SMRU [19]. These laboratories work on marine mammals. Trajectories of seals between their haulout sites along the coasts of the English Channel or in the Celtic and Irish seas are captured using GNSS [20] systems provided by SMRU, as shown in Figure 8.2. LIENSS deployed 63 tags since 2006 on marine mammals (30 on harbour seals and 33 on grey seals). The tags recorded 8952 raw data per day on average. The captured data, seal trajectories, can be classified into three main states : haulout, cruise and dive. Figure 8.3 shows the three states, the transitions and their guard conditions :

- GPS locations are captured every 20 minutes when the seal is at the surface ;
- diving data contains maximum depth, total duration, surface duration (time spent at the surface after a dive and before the next one) and the TAD index. The TAD (Time Allocation at Depth) index defines the shape of a seal's dive, as mentioned in [43]. A dive starts when the tag goes below a chosen "depth threshold". For most GPS/GSM deployments, the chosen depth threshold was 1.5 meters ;
- a haulout is a period of time spent on land by the seal. It starts when the tag is dry for at least 10 minutes and ends when the tag is wet for at least 40 seconds ;
- temperature of the sea water is given for 12 points per dive.

The main goal of the biologists when analyzing the captured data is to understand how seals use the marine habitat to feed. They are particularly interested in understanding how, when and where seals look for their food, and therefore characterize their preferred habitat (do they prefer one type of sediment, bathymetry, etc) and finally, where and when they could interact with human activities (such as fisheries). With our biologist, we define the main activities of seals when they are in the water : foraging, traveling and resting. These are the activities of parts of a trajectory. Biologist are interested in :

1. Foraging activities ;
2. Foraging activities during a given time interval ;
3. Foraging activities performed during a given time interval in a specific zone.

For all these queries, we have to define a seal trajectory domain rule called `Foraging`. However, for the last two queries, time rules must be defined between trajectory's parts. For example, the query 3 needs `Foraging` domain rule, `During` time rule and `Include` topological relationship as illustrated by Table 8.1.

18. http ://lienss.univ-larochelle.fr
19. SMRU : Sea Mammal Research Unit- http ://www.smru.st-and.ac.uk
20. GNSS : Global Navigation Satellite System

FIGURE 8.1 – Seal with a sensor

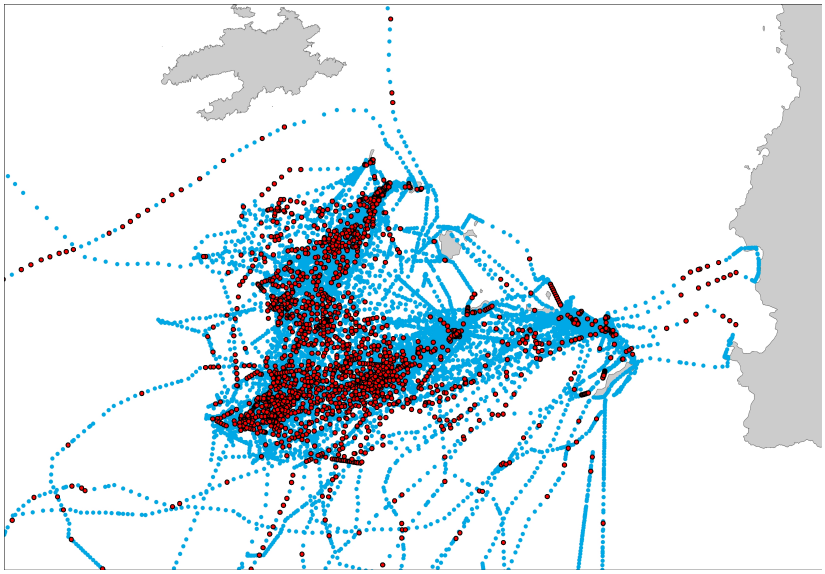

FIGURE 8.2 – Seal data captured by SMRU tags

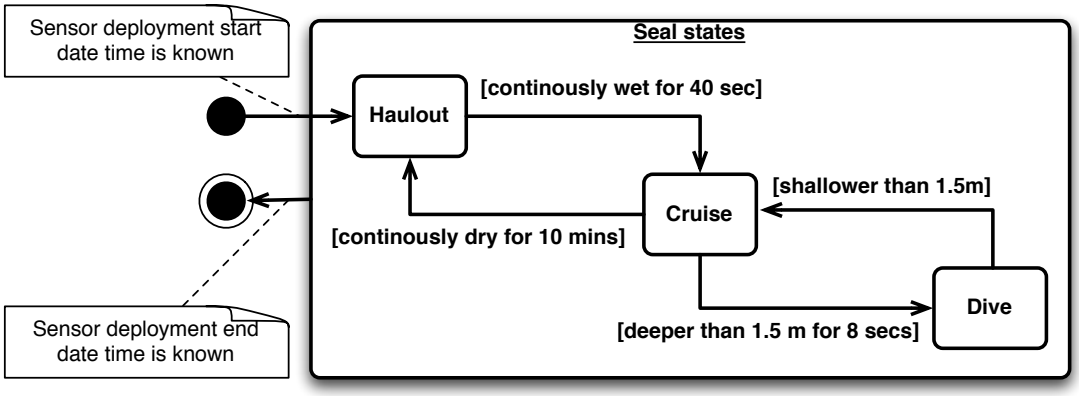

FIGURE 8.3 – The three states of seal trajectory

TABLE 8.1 – Domain, time, space concepts and rules needed for answering the query 3

Concepts and rules			Description
Concepts	Domain	Dive	specific part of the seal trajectory
	Time	Temporal interval	the given temporal interval
	Space	topological zone	the specific zone
Rules	Domain	Foraging	seal activity
	Time	During	temporal relationship between the activity and time interval
	Space	Include	topological relationship between the activity and specific space

8.2.2 Seal domain model

Two seal species occur in French waters : the grey seal (Halichoerus grypus) and the harbour seal (Phoca vitulina). The seal domain 8.4 is considered as a mobile object in our application. The `Seal` concept is connected to the `Mobile Object` concept as shown in the UML class diagram 8.4.

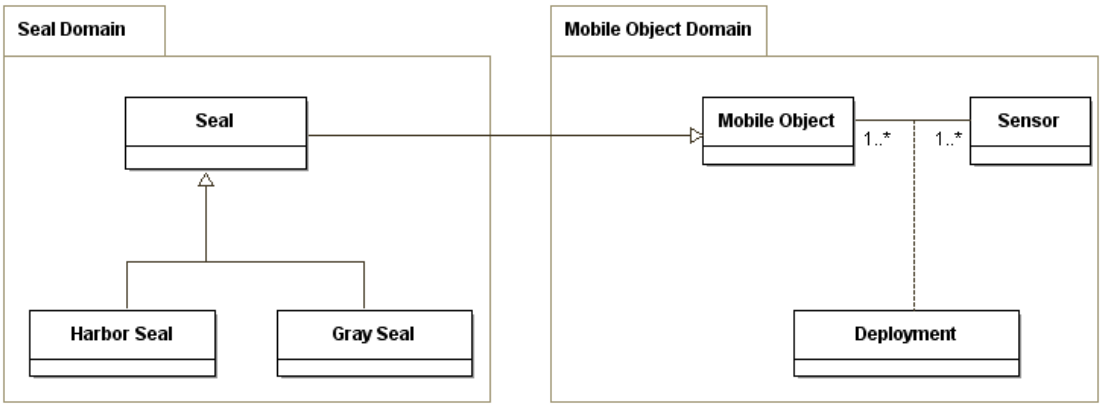

FIGURE 8.4 – Seal domain model

8.2.3 Seal trajectory model

A trajectory of a seal is composed of sequences. Every sequence characterizes a defined state of the seal : `Dive, Haulout` and `Cruise`. They are classes characterized spatially and temporally. A `CTD` (Conductivity-Temperature-Depth) sensor records the marine environment parameters such as temperature, pressure and water conductivity. `Summary` is metadata about the three states. From the analysis of the captured data, table 8.3 presents the concepts of the seal trajectory which are considered as the concepts of the trajectory domain. We define a seal trajectory model connected to the trajectory domain model, as shown in Figure 8.5.

TABLE 8.3 – Mapping the seal trajectory to the trajectory domain model

Seal trajectory	Trajectory domain
Dive, Haulout, Cruise	GeoSequence
CTD, Summary	Specific Sequence

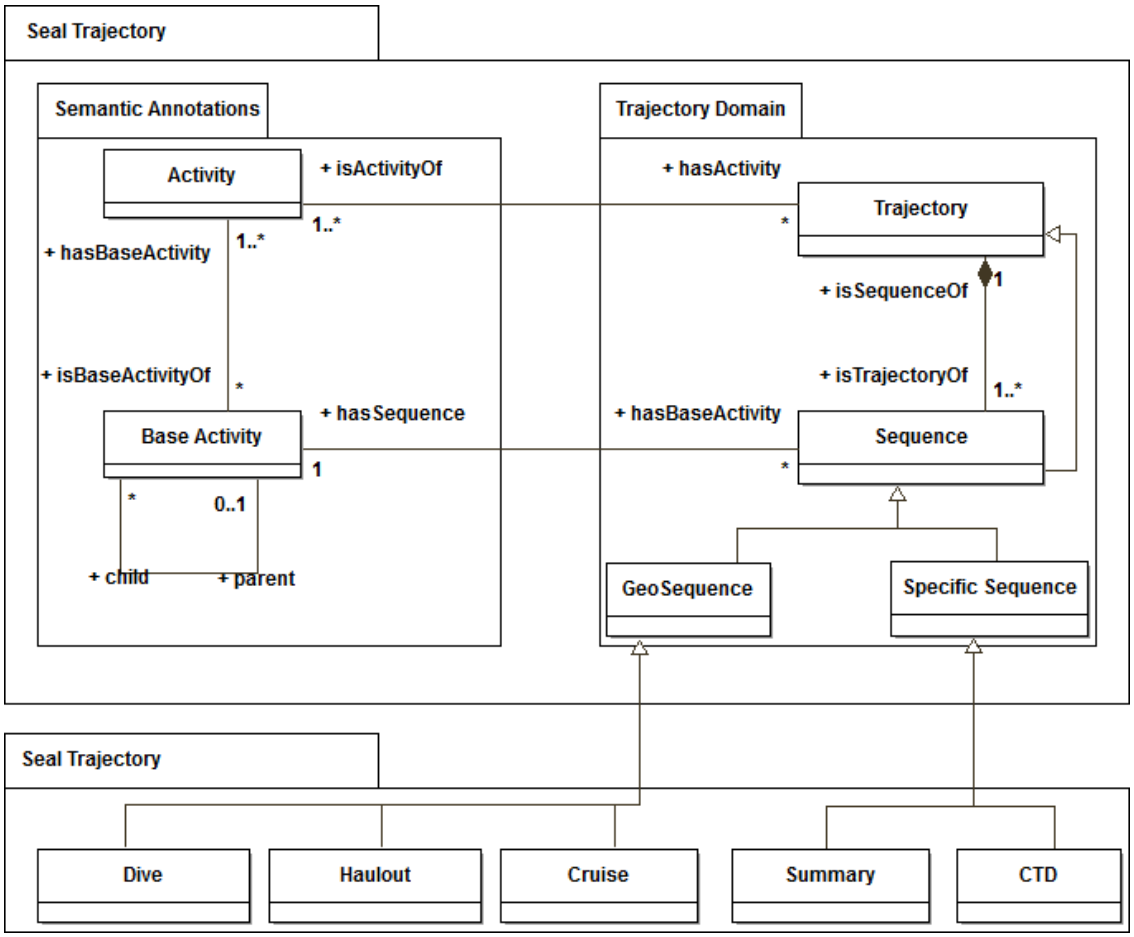

FIGURE 8.5 – Seal trajectory domain model

8.3 Seal trajectory ontology

8.3.1 Seal trajectory ontology concepts

We transform the seal domain model in (Figure 8.4) to mobile object seal ontology (Figure 8.6). A seal is considered as a mobile object in the `Mobile Object Domain` (MOD) ontology. An encoded formalization for this ontology is shown in Axiom 8.1.

$$Seal \subseteq MOD : MobileObject \tag{8.1}$$

We transform the seal trajectory, Figure 4.2, into a seal trajectory ontology, Figure 8.6. The formalization of these concepts are detailed in Axioms 8.2 - 8.10. In the `Seal Trajectory` ontology, geoSequence can be one of the three seal states, as shown in Axiom 8.2. These states are non-intersecting enforced by Axioms 8.3 -8.4. Each dive has a feature (`isFeatureOf`), Axiom 8.5. The class `Feature` contains four features : max_depth, dive_dur, sur_dur and TAD, as in Axioms 8.6 - 8.9. The Specific Sequence metadata is `Summary` and `CTD` defined by Axiom 8.10.

$$TD : GeoSequence \subseteq Dive \cup Haulout \cup Cruise \tag{8.2}$$
$$Dive \cap Haulout \cap Cruise = \emptyset \tag{8.3}$$
$$\forall x.Dive(x) \rightarrow TD : GeoSequence(x) \wedge \neg Haulout(x) \wedge \neg Cruise(x) \tag{8.4}$$
$$isFeatureOf \subseteq Dive \times Feature \tag{8.5}$$
$$Feature \subseteq \{f | \#(max\_depth \cap (\{f\} \times Double)) \geq 1\} \tag{8.6}$$
$$Feature \subseteq \{f | \#(tad \cap (\{f\} \times Double)) \geq 1\} \tag{8.7}$$
$$Feature \subseteq \{f | \#(dive\_dur \cap (\{f\} \times Double)) \geq 1\} \tag{8.8}$$
$$Feature \subseteq \{f | \#(sur\_dur \cap (\{f\} \times Double)) \geq 1\} \tag{8.9}$$
$$TD : SpecificSequence \subseteq Summary \cup CTD \tag{8.10}$$

8.3.2 Seal trajectory ontology with activities

We focus on the activities carried out during a seal trajectory (i.e. when the seal is in the water between two consecutive haoulouts). According to the domain expert, we distinguish four main activities :

- `Resting` is an activity where the seal is either sleeping or at least resting (limited activity) under water. Usually this is close to the haulout sites but in some circumstances it could be further away ;
- `Traveling` is a seal activity corresponding to the displacement of the animal from one place of interest to another one. It is usually far from the haulout site ;
- `Foraging` is an activity where the seal is looking for a prey or catching/eating it. It is usually far from the haulout site ;
- `Traveling-Foraging` is a seal activity which combines both previous activities, either because the seal is opportunistically feeding along its travel, or because we are unable to distinguish these two activities from the data collected.

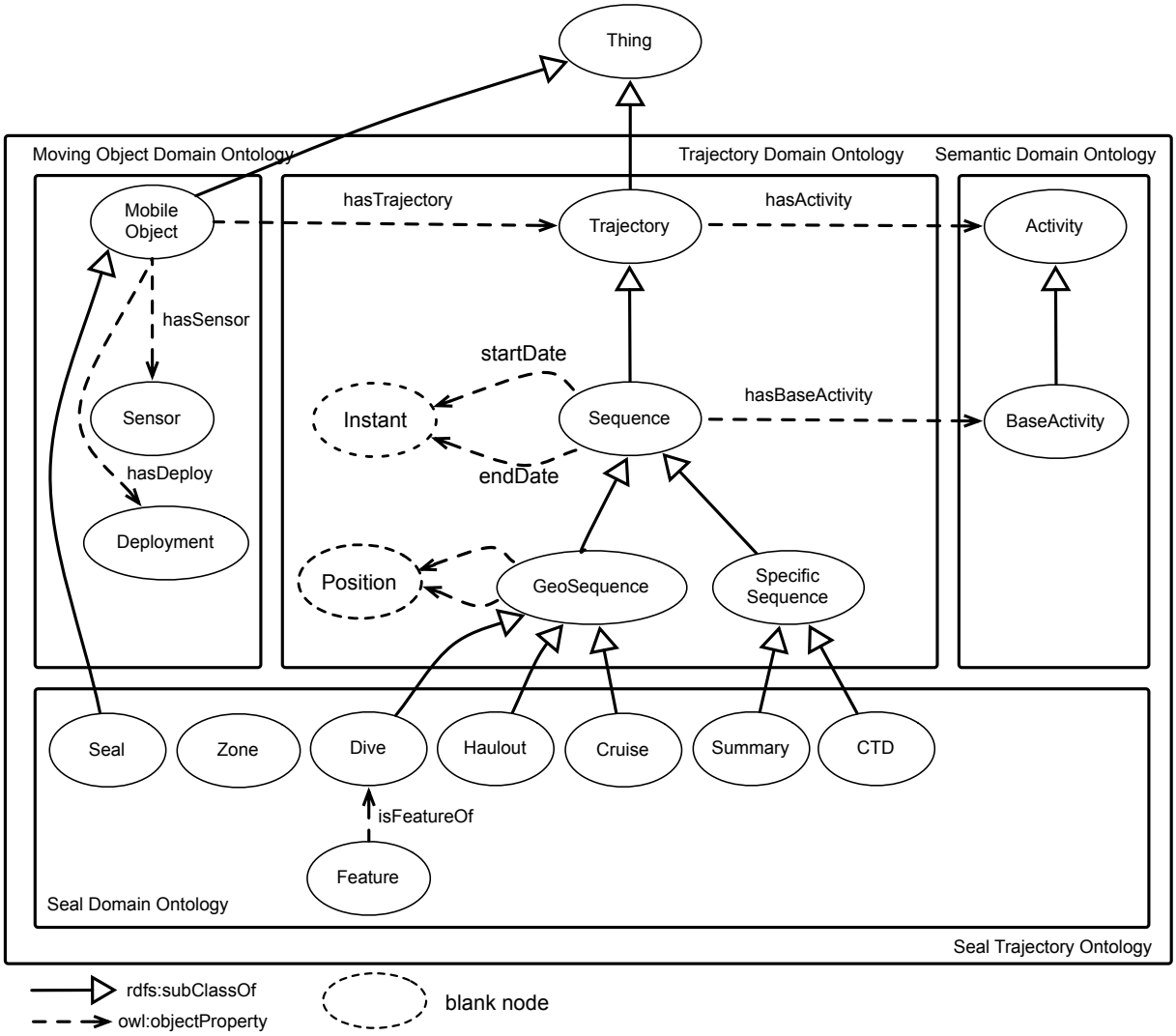

FIGURE 8.6 – Overview of the seal trajectory ontology

In the design part, Figure 8.7 shows an overview of the seal trajectory ontology with their activities. These activities are connected to the `BaseActivity` in the `Semantic Domain` (SD) ontology, Axiom 8.11. They are non-intersecting forced by Axiom 8.12.

$$SD : BaseActivity \subseteq Foraging \cup Traveling \cup Resting \tag{8.11}$$

$$Foraging \cap Traveling \cap Resting = \emptyset \tag{8.12}$$

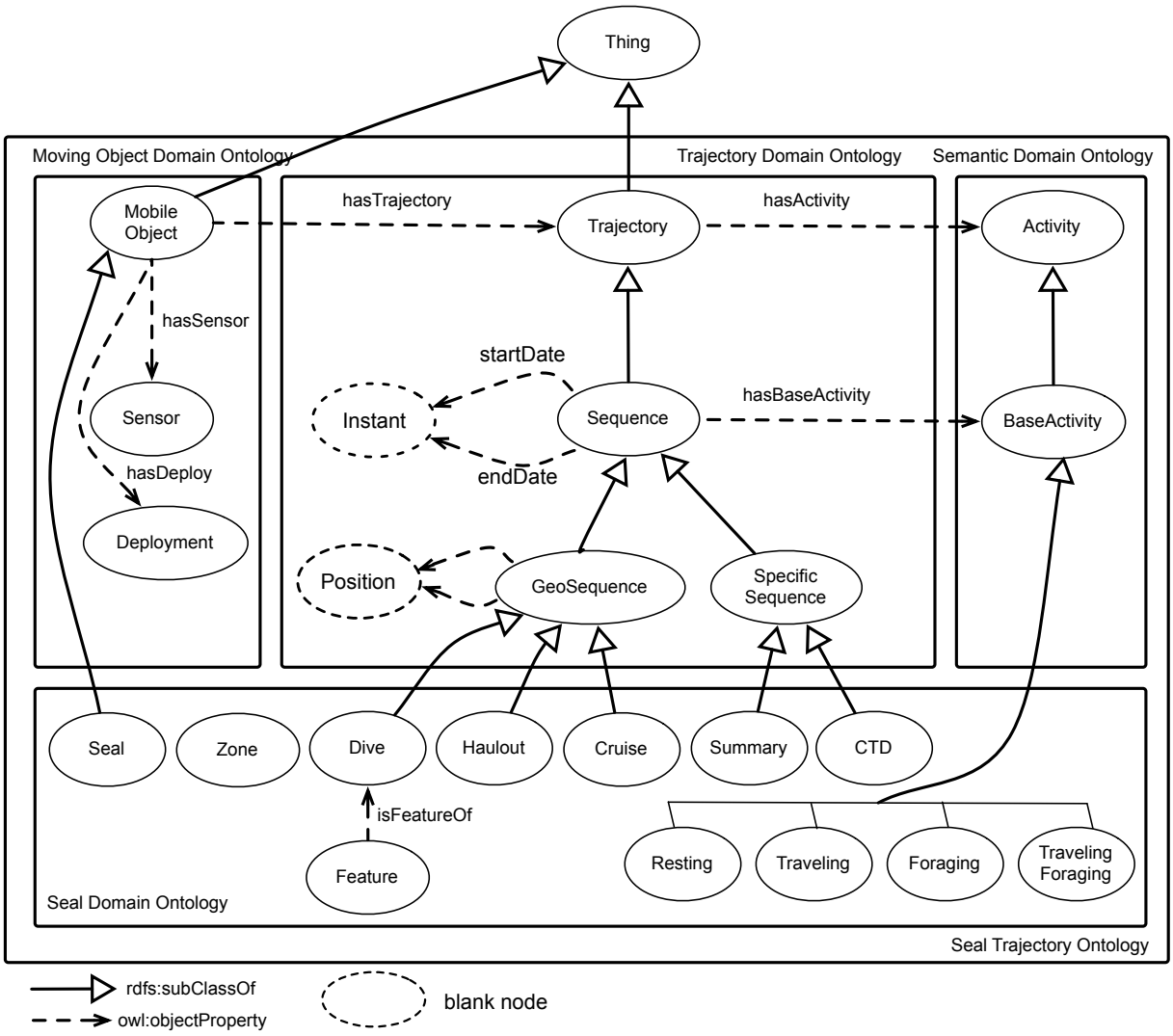

FIGURE 8.7 – Overview of the seal trajectory ontology with their activities

8.4 Application implementation

Our approach for creating declarative and imperative parts of seal trajectory ontology in the Oracle Semantic Data Store iw as follows :

1. Creating the declarative parts of the seal trajectory ontology ;

2. Matching the ontologies (seal trajectory and spatial ontologies) ;
3. Mapping relational data to RDF ;
4. Populating the ontologies ;
5. Creating seal trajectory ontology rules ;
6. Defining seal trajectory ontology inference.

8.4.1 Seal trajectory ontology declarative parts

Code 8.1 refers to the preparation to store seal trajectory triples, explained as follows :

1. A semantic data table for the seal ontology is created to refer its semantic data. This table, called `owlSealTrajectory_data`, has a column of type `SDO_RDF_TRIPLE_S (triple)`, which refers to semantic data, Line 1 ;
2. A semantic model for the seal ontology is created by the `SEM_APIS.CREATE_SEM_MODEL` procedure, called `owlSealTrajectory` and refers to the `triple` column, Line 2.

```
1 CREATE TABLE owlSealTrajectory_data (id NUMBER, triple SDO_RDF_TRIPLE_S);
2 EXECUTE SEM_APIS.CREATE_SEM_MODEL('owlSealTrajectory','owlSealTrajectory_data', 'triple');
```

Code 8.1 – Preparation to create `owlSealTrajectory` ontology

8.4.2 Matching the ontologies

We defined the seal trajectory ontology (Figure 8.7) which has to take into account the following considerations :

– `Zone` : can be considered as a spatial polygon `owlOGCSpatial:Polygon` ;
– `Dive` : can be considered as a spatial line `owlOGCSpatial:Line`.

To meet these requirements, we need to connect seal trajectory and spatial ontologies. Figure 8.8 shows the mapping defined between the seal trajectory and spatial ontologies according to our requirements. So, the concept `Zone` is an equivalent class with the concept `Polygon`. The second requirement where the concept `Dive` is an equivalent class with the concept `Line` is realized while `Dive` is inherited from the `Sequence` in the Trajectory Domain Ontology and the `Sequence` is an equivalent class with `Line`.

8.4.3 Mapping relational data to RDF

The majority of data underpinning the Web and in domains such as life sciences and spatial data management are stored in Relational DataBases (RDB) with their proven track record of scalability, efficient storage, optimized query execution and reliability. As compared to the relational data model, RDF is a more expressive data model and data expressed in RDF can be interpreted, processed and reasoned by software agents [117]. This is why the strategies for mapping relational data to RDF abound in [76]. The direct mapping defines an RDF graph representing data in a relational database. It can be seen as a transformation witch takes a relational database (data and scheme) as input and gives an RDF graph as output. In our work, we use the D2RQ Mapping Language [21]. D2RQ is a declarative language for mapping relational database schemes to RDF vocabularies and OWL ontologies. The language is implemented in the D2RQ Platform. Figure 8.9 illustrates the mapping process from RDB to RDF in this thesis.

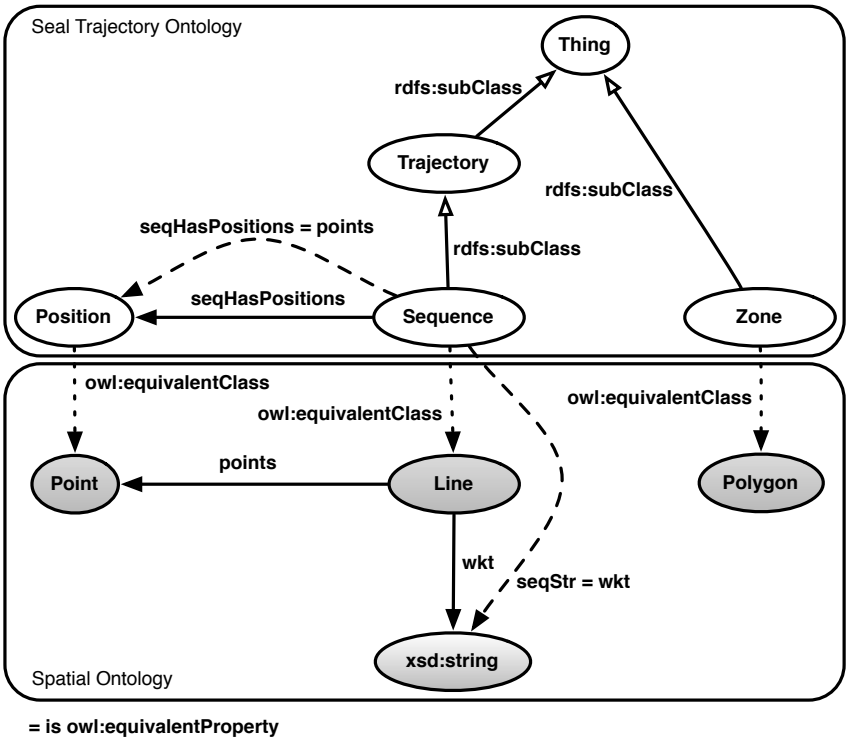

FIGURE 8.8 – Seal trajectory and spatial ontologies mapping

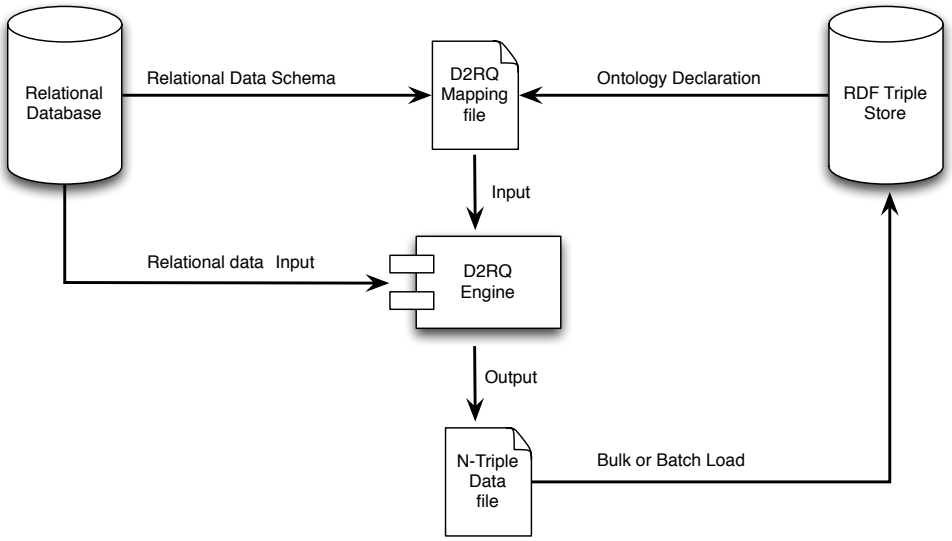

FIGURE 8.9 – RDB to RDF mapping process using D2RQ

8.4.4 Populating the ontologies

Loading RDF data in Oracle RDF triple store supports three forms (bulk load, batch load and incremental loading), discussed in 6.2.3. We load seal trajectory data in the table `owlSealTrajectory_data` using the bulk load form.

8.4.5 Seal trajectory ontology rules

Seal trajectory ontology (Figure 8.7) considers the seal's activities. Each seal activity has both a declarative part as an RDF `Class`, and an imperative part, formally, an associated rule as `IF-THEN` pattern. Figure 8.10 shows the declarative part of an activity `Foraging`. The imperative parts of activities are defined as rules in the ontology in a rulebase called `sealActivities_rb`. Therefore, every activity, or rule, will match a row in the view `MDSYS.SEMR_sealActivities_rb`.

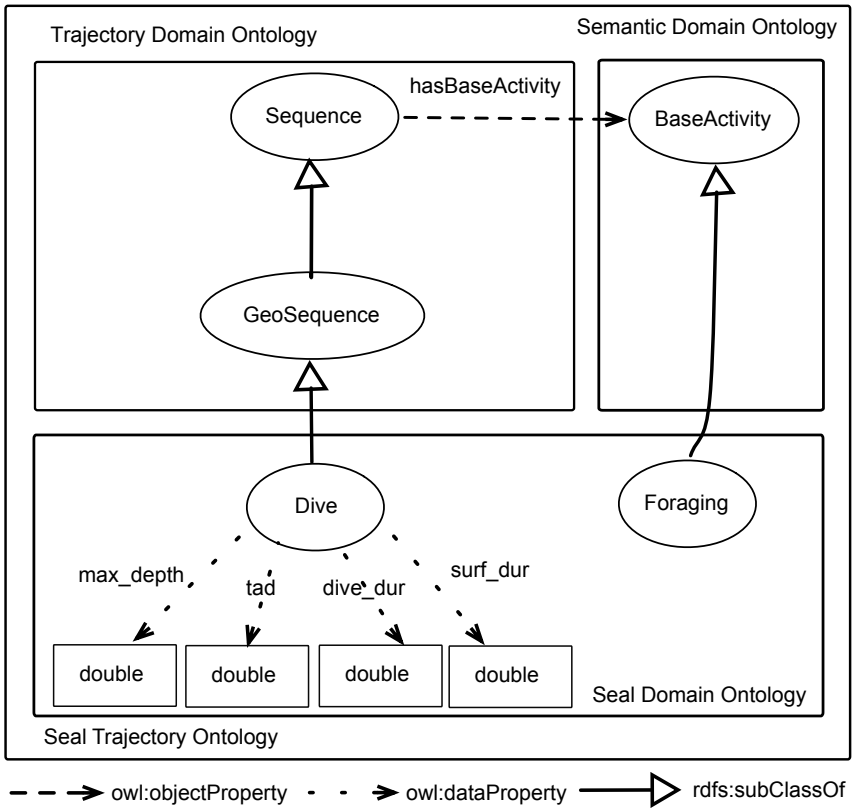

FIGURE 8.10 – Declarative part of `Foraging` rule

According to the domain expert, we take into consideration different parameters to define the seal activities. The parameters are the geometrical shape of dives (`TAD`), the maximum dive depth and surface ratio which is the ratio between surface duration and dive duration. The activities are :

– `Resting` is when a seal is sleeping at the sea bottom with the TAD higher than 0.9. The surface duration after the dive state should be quite high so that seals have enough time to breath before another sleep under water ;

TABLE 8.4 – Decision table associated with seal activities

Rules	**Max dive depth** (meter)	**Dive shape** or TAD	**Surface ratio** = surface dur/dive dur
Resting	< 10	>0.9	> 0.5
Travelling	> 3	< 0.7	$<$ all
Foraging	> 3	> 0.9	< 0.5
TravellingForaging	> 3	> 0.7 & < 0.9	< 0.5

– `Traveling` could be in any dive depth deeper than 3 meters, but the TAD should be lower than 0.7 because the seal does not need to spend a lot of time at the maximum depth. The surface duration does not make any difference in this case ;
– `Foraging` is when the dive depth is deeper than 3 meters. The TAD however should be high (>0.9) because the grey seal is a benthic forager, which means it is feeding on fish located on or close to the sea bottom (i.e at the maximum depth available). Also the surface duration is short because the seal wants to go back quickly to look for more fish ;
– `TravelingForaging` is when the dive depth is deeper than 3 meters. The TAD however should be higher than (>0.7) and smaller than (>0.9). Also the surface duration is short because the seal wants to go back quickly to look for more fish.

The decision Table 8.4 summarizes the above conditions to be considered in the IF part of a rule associated with each activity. Based on this table, Code 8.2 is the imperative part of the `Foraging` activity. Line 4 to 9 construct a subgraph and necessary variables needed by the `IF` part of `foraging rule`. Line 10 is the `Filter` part of this rule. Line 11 gives the `THEN` part associating a foraging activity to an object. Line 12 defines the aliases of seal trajectory ontology.

```
1  EXECUTE SEM_APIS.CREATE_RULEBASE('sealActivities_rb');
2  INSERT INTO mdsys.semr_sealActivities_rb
3  VALUES( 'foraging_rule',
4  '(?diveObject               rdf:type                    s:Dive              )
5   (?diveObject               s:max_depth                 ?maxDepth           )
6   (?diveObject               s:tad                       ?diveTAD            )
7   (?diveObject               s:dive_dur                  ?diveDur            )
8   (?diveObject               s:surf_dur                  ?surfaceDur         )
9   (?diveObject               s:seqHasActivity            ?activityProberty   )',
10 '(maxDepth > 3) and (diveTAD > 0.9)        and (surfaceDur/diveDur < 0.5      )',
11 '(?activityProberty         rdf:type                    s:Foraging          )',
12 SEM_ALIASES(SEM_ALIAS('s','http://l3i.univ-larochelle.fr/owlSealTrajectory#')) );
```

Code 8.2 – Implementation of `Foraging rule`

8.4.6 Seal trajectory ontology inference

We want to apply the domain activity over all considered data. So, we need to take into consideration the domain application, `Seal Trajectory` ontology, when applying the inference. We modify the entailment `owlTrajectory_idx` created in Sec.7.3, to consider the model and the rulebase of seal trajectory ontology sealActivities_rb. Code 8.3 creates the entailment over trajectory `owlTrajectory`, temporal `owlTime`, spatial owlOGCSpatial and seal trajectory `owlSealTrajectory` models and their rulebases : owlTime_rb, owlOGCSpatial_rb, sealActivities_rb, respectively.

```
SEM_APIS.CREATE_ENTAILMENT('owlTrajectory_idx',
SEM_MODELS('owlTrajectory','owlTime','owlOGCSpatial','owlSealTrajectory'),
SEM_RULEBASES('OWLPrime',owlTime_rb,'owlOGCSpatial_rb','sealActivities_rb'),
SEM_APIS.REACH_CLOSURE,
NULL,
'USER_RULES=T');
```

Code 8.3 – Entailment over domain trajectory application, temporal and spatial models

8.5 Seal trajectory ontology inference refinement

A complexity problem occurs when computing the inference over all trajectory data. However, users could not be interested in all of these data. Their interest focuses on places where a moving object stays longer and visits more often. Therefore, the observed movement path of the moving object will consist of two different search strategies, Figure 8.11. Extensive movement corresponds to periods of movement with high speed and low turning rate, and hence, a small area will be crossed more quickly. Conversely, intensive movement corresponds to periods of movement with low speed and high turning rate. The latter term is `Area of Restricted Search (ARS)` [81] which represents an area where a moving object spends more time.

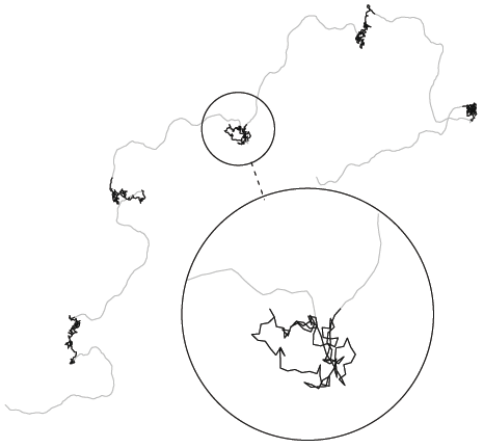

FIGURE 8.11 – Simulated moving object movement path consisting of two unique search strategies. Extensive movement (grey) and intensive movement/ARS (black)

8.5.1 Two-tier inference refinement

We introduce a two-tier inference refinement on the trajectory data. In other words, two distinct operations are performed to enhance the inference : primary and secondary inference operations. Figure 8.12 shows the two-tier inference refinement. The primary filter is applied to the captured data to classify them into a set of interesting places (ARSs). The primary filter allows fast selection of the classified data to pass along to the secondary inference. The latter computes the inference mechanism considering the ARS. Then, instead of annotating each sequence in the model, we annotate the ARSs with the expert knowledge activity model. The inference process

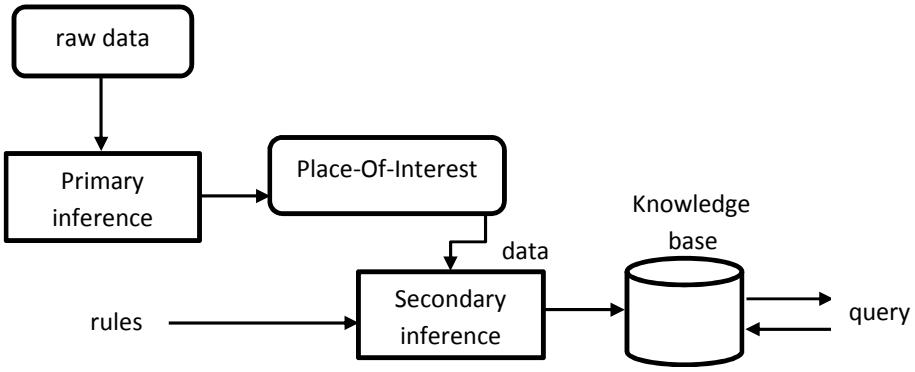

FIGURE 8.12 – Two-tier inference filter refinement

is computed for each ARS. The secondary inference yields the final knowledge data that the user can query.

Our proposal is to analyze the captured data before computing the ontology inference. This analysis is achieved thanks to our primary inference. This step considers trajectories that are segmented by the object positions. These positions change and remain fixed. Spaccapietra [126] named the former `moves` and the latter `stops`. For this reason, a trajectory is seen as a sequence of moves going from one stop to the next one.

Definition 7 (Stop) *A stop is a part of a trajectory having a time interval and represented as a single point.*

Definition 8 (Move) *A move is a part of a trajectory represented as a spatio-temporal line.*

The primary filter defines interesting ARSs for a moving object. Each ARS contains sequences aggregated according to two properties : places that the moving object visited more often or places where it stayed longer. This filter takes the two parts of a trajectory (move and stop) data as input and gives ARSs as output. The secondary inference is the inference mechanism applied over these ARSs and the considered rules.

8.5.2 Two-tier inference refinement algorithm

The two-tier inference refinement is applied over trajectory data. The primary filter classifies data into a set of ARSs. The primary filter is based on the following definitions :

Definition 9 (Neighbors) *Neighbors of a point* (p_i) *are a list of points from the Move data where the distance between* p_i *and any neighbor point is smaller than a fixed radius.* $Neighbor(p_i) = \{(p_j)_{j=1}^{n} : p_i, p_j \in Move,\ distance(p_i, p_j) < radius\}$.

Definition 10 (Points_Neighbors) *Points_Neighbors are a list of points and their neighbors.* $Points\_Neighbors = \{(p_i, Neighbors_i)_{i=1}^{n} : p_i, Neighbors_i \in Move\}$.

input : $Move$
input : $Stop$
input : $radius$
output: $Places$
1 initialization;
2 $Neighbor \leftarrow \phi$;
3 $Points\_Neighbors \leftarrow \phi$;
4 $Places \leftarrow \phi$;
5 **for** *each* $p_i \in Move$ **do**
6 | calculate $Neighbor(p_i)$;
7 | $Points\_Neighbors \leftarrow (p_i, Neighbors(p_i))$;
8 | $Move \leftarrow Move - Neighbor(p_i)$;
9 **end**
10 **for** *each* $p_i \in Points\_Neighbors$ *AND condition*$(distance(p_i, Stop) > radius)$ **do**
11 | **if** $distance(p_i, Places[j]) > radius$ **then**
12 | | $Places[k] \leftarrow (Neighbors_i, 1)$;
13 | **else**
14 | | $Places(Neighbors_j, nVisits_j) = ([Neighbors_j, Neighbors_i], nVisits_j + 1)$;
15 | **end**
16 **end**

Algorithm 4: The area-restricted search (ARS) algorithm

Definition 11 (Places) *$Place_i$ is an interesting place which contains the $Neighbor(p_i)$ and number of its visits ($nVisits$) by the moving object. $Places = \{(Neighbors_i, nVisits_i)_{i=1}^{n} : Neighbors_i \in Move,\ nVisits_i \in number\}$.*

Algorithm 4 is the primary filter. Lines 5-9 gather the move data into groups of neighbors. These groups are defined with respect to a *radius*, an input for this algorithm. This radius is a fixed distance between two points to calculate the neighbors. The candidate of the radius is related to the application view of a trajectory. The group list is in *Points_Neighbors*.

Lines 10-16 define the interesting places. In general, we can consider all the members of *Points_Neighbors* or a part of them respecting to a domain condition. For example, the application view could be interesting when a place has 60 points and over, or could be interesting in any place having at least a point. For defining a place, the coordinates of the neighbors could be an interesting place after applying two conditions. Every point that belongs to a place should be far from the stop data more than the fixed radius. Any place should not have any neighbor place within the radius distance, otherwise we merge the two coordinates and increase the visits number. The ARSs are the output *Places* of this algorithm.

8.6 Conclusion

In this chapter, We described an application of the conceptual trajectories seal proposals.

we applied our modeling approach in a particular domain application : marine mammal trajectories. We presented the seal trajectory model considering the expert knowledge. Then the associated ontology with the generic semantic trajectory model was described. The concepts and the relationships between these concepts were exposed. An implementation of this ontology was detailed. The ontology was populated from the collected data that was stored in a database, before being transformed into RDF triples through D2RQ mapping engine. To consider the knowledge of seal application, semantic rules were used to characterize specific activities (rest, movement, foraging). The inference mechanism is computed using these rules. To solve the inference complexity problem, we designed and implemented a refinement of inference rules, called a two-tier refinement refinement. The first step is to determine regions of interest with high-density data and therefore may contain reveal interesting information activities of seals. Then, the inference applies over these places.

Part IV

Research results

Table of Contents

This part is the evaluation part of our work and contribution. We evaluate the trajectory ontology inference in Chap. 9. In our evaluation, we consider sets of real GPS seal trajectories. To evaluate the time inference over seal trajectories, we consider the 13 temporal interval rules. Then to evaluate the spatial inference, we consider the eight topological spatial rules. Moreover, we evaluate the trajectory ontology over the spatio-temporal inference considering spatio-temporal rules. From the results of these experiments, we address the computation complexity, especially user-defined rules inference complexity.

In Chap. 10, we evaluate our contribution : trajectory ontology inference refinement. Firstly, we evaluate the temporal inference refinement, then the spatial inference refinement. Moreover, we perform experiments to evaluate the spatio-temporal inference refinements. Overall, we discuss the enhancement made by these refinements in term of reducing the inference complexity. Finally, we perform experiments to evaluate the application refinement. The two-tier filter inference refinement reduces the considered data into interesting data organized in interesting places.

Chapter 9

Trajectory ontology inference

Contents

9.1 Introduction

Experiments are performed over huge amount of captured seal data. In 2011, we have 410 690 raw data as seal dives and 1 255 raw data as seal haulout. We also consider data captured in 2006, 2007 and 2008. These experiments are proceeded on a high-performance server running OS Linux 2.6.18 and Java 1.6 on an Intel (R) Xeon X7560 CPU at 2.27GHz with a maximum of 40GB. The reasoner invoked is Oracle 11G. In our experiments, we consider sets of real GPS seal trajectory data. The evaluation curves is according to the number of data (dive) treated.

9.2 Trajectory ontology inference : time inference

Code 9.1 creates an entailment over the domain seal trajectory and time models. This entailment uses a subset of OWL rules called `OWLPrime` [103], the seal trajectory rule 8.4 and the 13 temporal rules 7.2. In our experiment, we measure the time needed to compute the entailment (Code 9.1) for different sets of real trajectory data for one seal. Its movements were captured from 16 to 18 June 2011 and led to the capture of 10 000 data. In this experiment, the seal activity rulebase contains only the foraging rule. The input data for this entailment are only dives. Figure 9.1

shows the experiment results for the time computation in seconds needed by the entailment. For example, for 450 dives, the inference takes around 60 000 seconds ($\simeq$ 16.6 hours). Figure 9.2 shows the experiment results for the number of the triples inferred by the inference mechanism. For example, for 450 dives, the inference takes around 2 200 000 triples.

```
SEM_APIS.CREATE_ENTAILMENT('owlSealTrajectory_idx',
SEM_MODELS('owlSealTrajectory','owlTime'),
SEM_RULEBASES('OWLPrime','sealActivities_rb', 'owlTime_rb'),
SEM_APIS.REACH_CLOSURE, NULL, 'USER_RULES=T');
```

Code 9.1 – Entailment over the `owlSealTrajectory` and `owlTime` ontologies

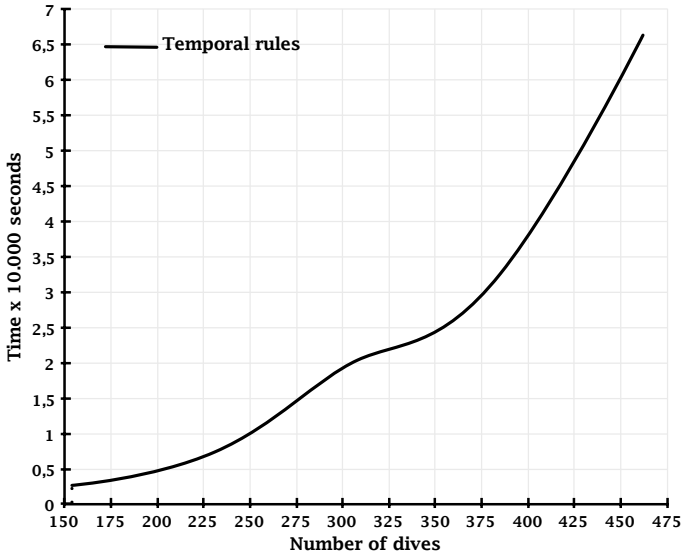

FIGURE 9.1 – Time computation over the temporal rules

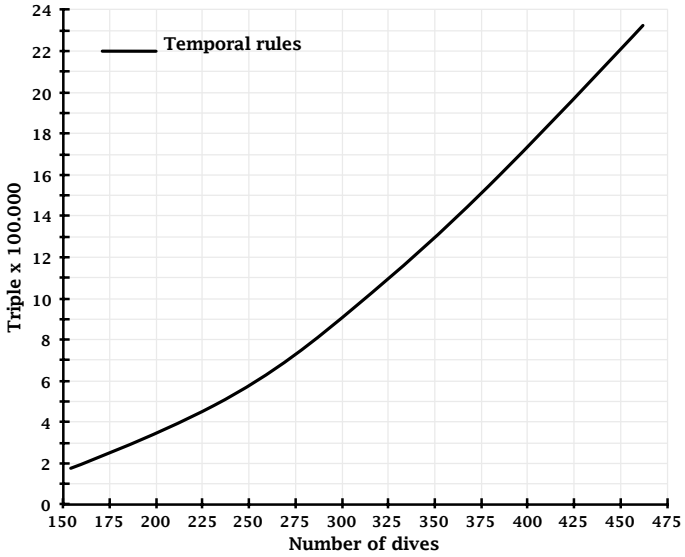

FIGURE 9.2 – Triples inferred over the temporal rules

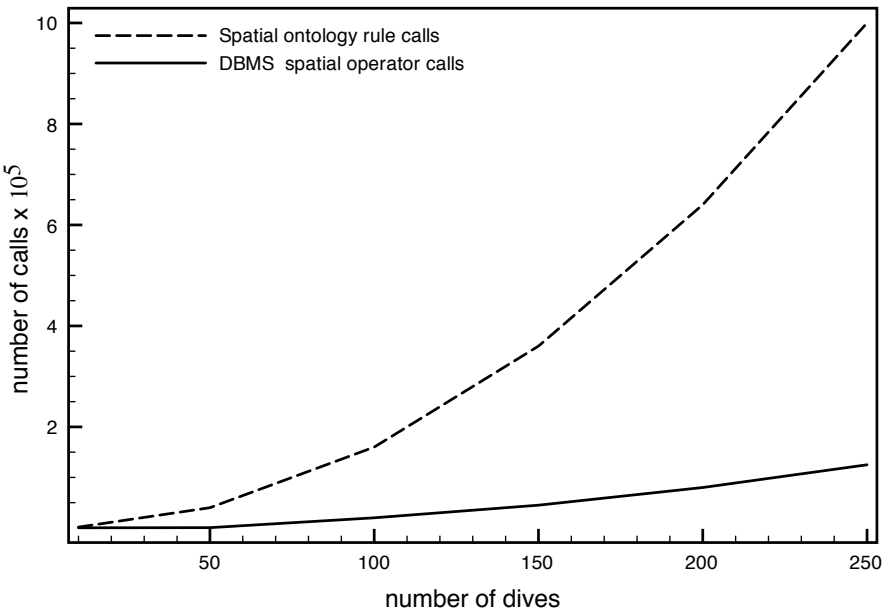

FIGURE 9.3 – Spatial ontology rules and DBMS spatial operators calls

9.3 Trajectory ontology inference : spatial inference

The spatial ontology inference is the process of applying the eight spatial rules 7.2 to compute topological relationships between spatial objects. For this, the system needs an entailment over seal trajectory and spatial rules. The system must compile the spatial relationships between zones and dives, considered as spatial polygons and lines, respectively.

In Oracle RDF triple store, an entailment contains precomputed data inferred from applying a specified set of rulebases to a specified set of semantic models. Code 9.2 creates an entailment using domain trajectory and spatial models. This entailment uses a subset of OWL rules called `OWLPrime` [103], the domain seal trajectory rules 8.4 and spatial rules 7.2. Other options are also required like the number of rounds that the inference engine should run. Where applying user-defined rules `USER_RULES=T`, the number of rounds should be assigned as default to `REACH_CLOSURE`.

```
1 SEM_APIS.CREATE_ENTAILMENT('owlSealTrajectory_idx',
2 SEM_MODELS('owlSealTrajectory','owlOGCSpatial'),
3 SEM_RULEBASES('OWLPrime','sealActivities_rb','owlOGCSpatial_rb'),
4 SEM_APIS.REACH_CLOSURE,
5 NULL,
6 'USER_RULES=T');
```

Code 9.2 – Entailment over `owlSealTrajectory` and `owlOGCSpatial` models

In this part, we discuss the cost of the spatial inference process. This cost study depends on how many Oracle operations are performed during the inference process. Figure 9.3 shows the number of executions of the spatial ontology rules is 1 000 000 and the number of DBMS spatial operator calls is 125 000, for 250 dives. Figure 9.4 shows time executions of the spatial ontology rules. For example, for 300 dives, the inference takes around 120 000 seconds ($\simeq$ 35.5 hours).

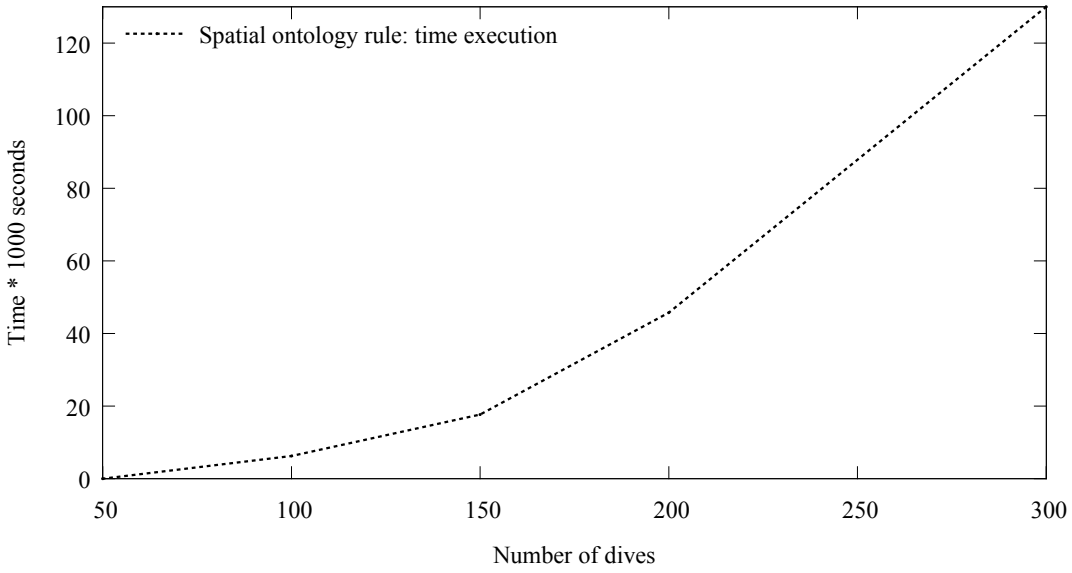

FIGURE 9.4 – Time computation of spatial ontology rules needed by the inference mechanism

9.4 Trajectory ontology inference : spatio-temporal inference

In this section, we perform experiments with the aim of outlining the inference difficulties over all the needed ontologies and rules. The inference uses the eight spatial rules, the 13 temporal rules 7.2 and the trajectory domain foraging rule 8.4. Figure 9.5 shows the experimental results over the three ontologies and their rules. The vertical axis measures the number of executions needed to compute the inference mechanism for each considered number of dives. For example, for 500 dives, the inference needs 6 000 000 calls to be normally computed over 22 spatio-temporal trajectory rules.

9.5 Computation complexity

9.5.1 Spatial inference complexity

Figure 9.3 in the experimental evaluation of spatial ontology inference on semantic trajectory shows two surveys :

1. *Spatial ontology rules calls* : the results show the evolution of the spatial rules calls based on the number of dives. The size of this number is given by Equation 9.1. The `DiveNb` is the number of dives loaded in the ontology model and the `RuleNb` is the number of the spatial rules considered during the inference.

$$Rule\ index\ execution = 2 * DiveNb^2 * RuleNb \tag{9.1}$$

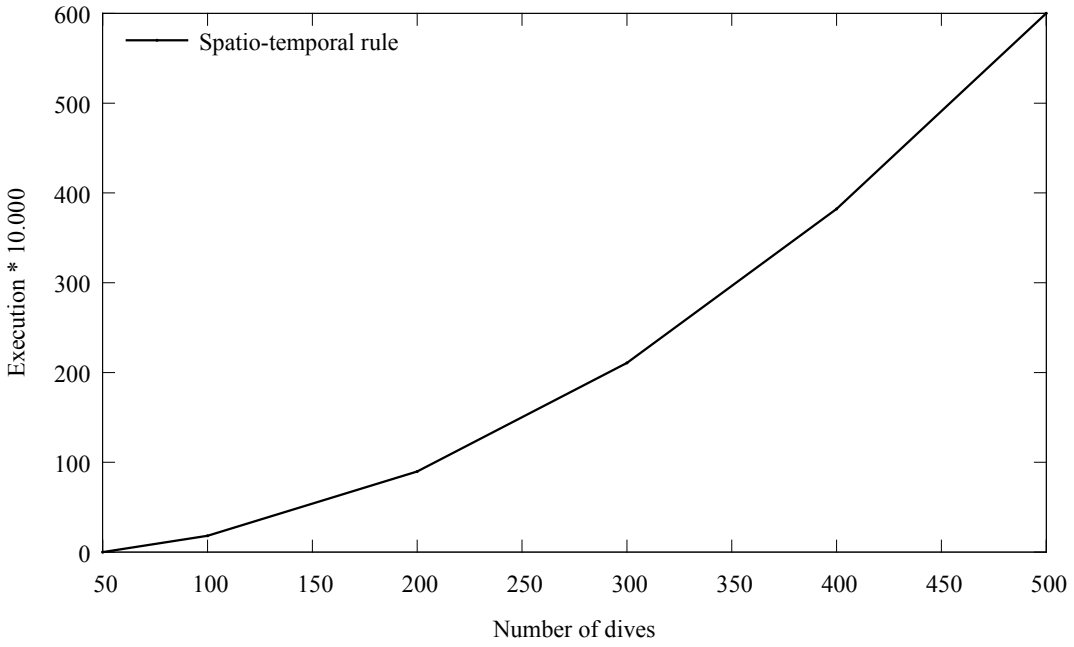

FIGURE 9.5 – Spatio-temporal trajectory ontology inference

2. *DBMS spatial operators calls* : determines the number of calls of each Oracle spatial operator. The size of this number is given by Equation 9.2.

$$Oracle\ spatial\ operator\ call = 2 * DiveNb^2 \tag{9.2}$$

The complexity of the size of computing the inference on spatial rules appears clearly from the results in Fig. 9.3 and from the form of Equations (9.1, 9.2). The number of the spatial ontology rule calls and the number of Oracle spatial operator calls are growing quickly when the loaded data increases.

9.5.2 User-defined rules inference complexity

Oracle Semantic Data Store did not perform special options for the inference over spatial data. Moreover, Oracle Semantic Data Store has limitations on supporting effective options in the case of user-defined rules :

1. The `INC` option enables incremental inference taking into account the inference of the previous step. This option saves index execution and time. However it is not available with user-defined rules ;
2. The `REACH_CLOSURE` option is the default value for the number of rounds for the engine. It must be respected in the case of user-defined rules ;
3. The `DISTANCE` option generates additional distance information that is useful for semantic operators. It is also not available with user-defined rules.

Furthermore in our experiments, we compute the inference independently of any semantic query, which means that we compute the inference mechanism off-line. Moreover in terms of updating data or new data captured, the inference process should be computed once again.

9.6 Discussion

We evaluated the inference performance in terms of inferred graph size, and computation time. We computed the inference over OWLPrime and user-defined rules. We noticed that the inference time increases seemingly out of proportion when using user-defined rules. This is because the Oracle inference engine does not integrate OWL and user-defined rule inference components similarly. Behind the scene, Oracle first runs OWL related inference till a closure occurs. Then Oracle runs user-defined rules till a closure occurs, and repeats the whole process till no new triple is generated from either component. Finally, the inference time increases as the vocabulary becomes increasingly expressive, especially when using OWLPrime with user-defined rules. In term of time computation, the inference over the time rules takes 16.6 hours for 450 dives, while the inference over the spatial rules takes 35.5 hours. In term memory and space storage, the inference over the time rules generates more than 2 millions for 450 dives, the inference over the spatial rules is executed 1 million times and the Oracle spatial operators is executed 125 thousands for 250 dives. AS a conclusion, we have to enhance the inference computation complexity by our refinements.

9.7 Conclusion

We evaluated the inference computation using temporal, spatial and spatio-temporal rules. We addressed the inference complexity over these user-defined rules. The system of Oracle manages the user rules differently than the standard OWLPrime rules. We should mention that we could not be able to compute the inference using these rules over more that 500 raw trajectories.

Chapter 10

Trajectory ontology inference refinement

Contents

10.1 Introduction

This chapter is the mirror of the previous one, however it shows evaluations over the temporal, spatial, and spatio-temporal refinements, as well as the two-tire inference refinement associated with regions of interest. We present results in terms of time computation in hours, memory and space storage in number of executions.

10.2 Temporal inference refinement

We performed experiments with the aim of measuring the impact of the introduction of temporal refinement in the inference process computation. Figures 10.1 and 10.2 show results from experiment, on the time computation in seconds and the storage space in triples needed by the inference calculation. The evolution curves is given by the number of dives. In all the following experiments, shown in Fig. 10.1 and 10.2, we consider the domain rules :

1. The experiment named *Temporal rules* analyses the inference on real data taking into account a classic version of temporal rules ;

2. The experiment named *Temporal rules refined - Real data* analyses the inference on real data considering the refinement of temporal rules given by Algorithm 1 ;
3. The experiment named *Temporal rules refined - Generated data* analyses the inference on generated data as in the previous experiment.

It clearly appears that the experiment 1 gives poor inference results in terms of time computation and space storage. For example, for 500 dives, the inference takes around 67.000 seconds ($\simeq$ 18.5 hours) and generates 2.300.000 triples. In our point of view, this problem occurs because of time integration without applying any domain constraints on temporal rules. In this work, we propose a first solution to this problem by defining a domain constraint on temporal intervals based on the conceptual distance in the ontology hierarchy. This constraint limits the calculation of temporal rules into the neighbourhood of the current interval. From the seal trajectory domain and with our biological feedback, we candidate the conceptual distance between two sequences to five minutes (300 seconds). So, we modify the implementation of the temporal rules considering this candidate. For instance, the implementation of the `intervalAfter_Refined` rule, is given by the code 10.1.

```
1 EXECUTE SEM_APIS.CREATE_RULEBASE('owlTime_rb')
2 INSERT INTO mdsys.semr_owltime_rb VALUES(
3 'intervalAfter_Refined_rule',
4 '(?timeObject1 rdf:type owltime:ProperInterval)(?timeObject1 owltime:hasEnd ?End1)(?End1 :
      inXSDDateTime ?EndTime1)
5 (?timeObject2 rdf:type owltime:ProperInterval)(?timeObject2 owltime:hasBeginning ?Begin2)(?Begin2 :
      inXSDDateTime ?BeginTime2)',
6 '(BeginTime2 > EndTime1)
7 ((timeIntervalLengthInSeconds(dateTime2TimeStamp(EndTime1),dateTime2TimeStamp(BeginTime2))<300)',
8 '(?timeObject2 owltime:intervalAfter_Refined ?timeObject1)',
9 SEM_ALIASES(SEM_ALIAS('owltime','http://www.w3.org/2006/time#')));
```

Code 10.1 – Creating of the temporal `intervalAfter_Refined rule`

In the experiment 2, we consider real GPS/GSM data and the inference uses the refined temporal rules. The time computation and space storage results show the improvement made on the inference calculation comparing to the experiment 1. For example, for 500 dives, the inference takes less than 30.000 seconds ($\simeq$ 8 hours) and generates less than 1.100.000 triples. In the experiment 3, the inference is calculated on generated data and uses the refined temporal rules. Generated data contains temporal intervals with the same initial density for temporal relationships. The results show the reasonable time computation and space storage taken by the inference mechanism. These experiments provide a view of the inference behaviour while considering independent or neutral data.

10.3 Spatial inference refinement

In this section we evaluate the two spatial refinements we introduced in this work. The inference uses the eight spatial rules and the trajectory domain foraging rule.

Firstly, we evaluate the area of interest refinement. Related to the seal trajectory domain and to our domain knowledge, we limit the area of interest to 500 meters. We pass this candidate to Algorithm 2. The experimental results of this proposed refinement are shown in Figure 10.3. The results show its impact by the three following experiments :

1. *Spatial ontology rule calls - constraints refinement* presents the number of executions of the spatial ontology rules using the area of interest refinement ;

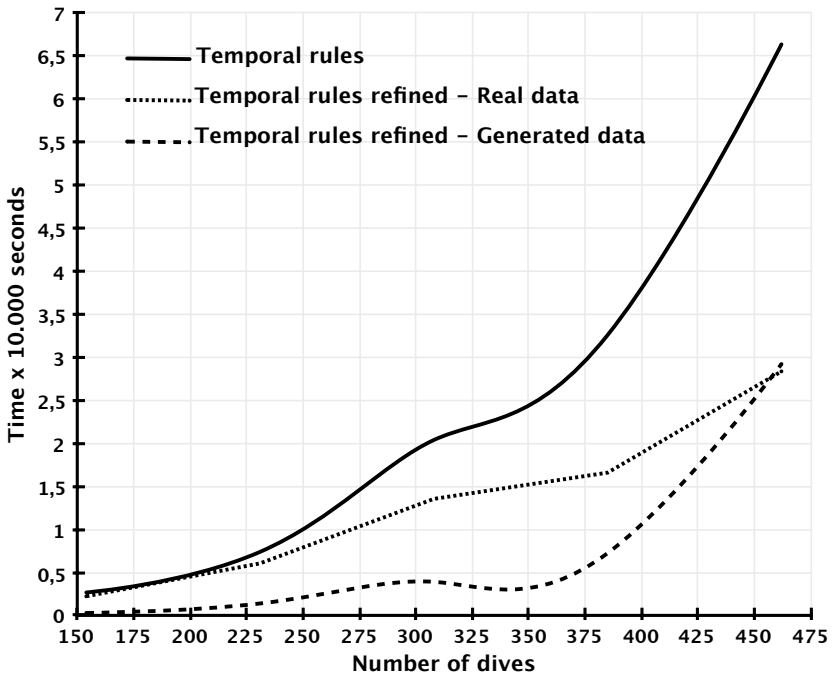

FIGURE 10.1 – Inference time computation with(out) the temporal rules refinement

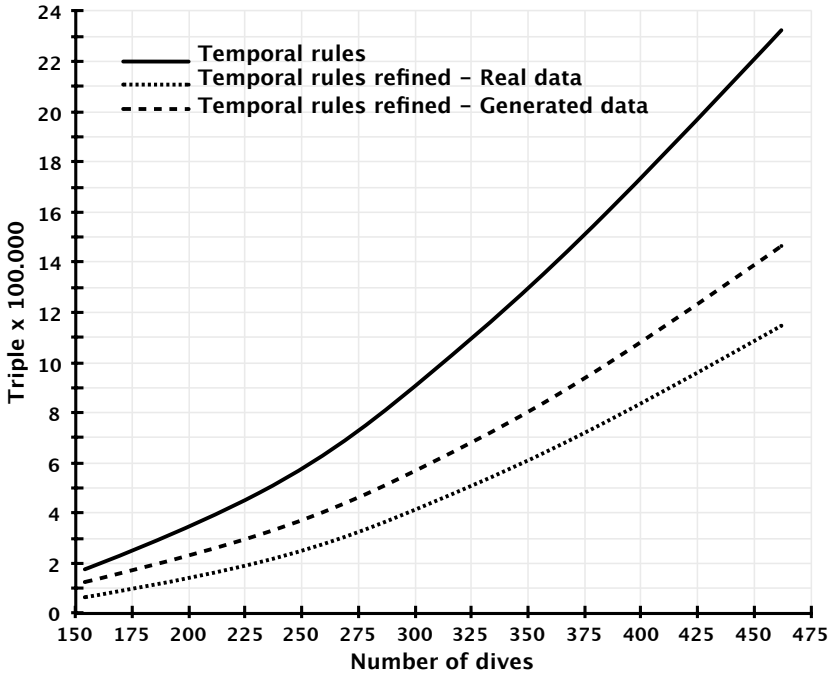

FIGURE 10.2 – Inference storage space taken with(out) the temporal rules refinement

2. *Spatial ontology rule calls not executed* gives the number of reduced executions of the spatial ontology rule after applying this constraints refinement comparing to the results in Figure 9.3 ;
3. *DMBMS spatial operator calls - constraints refinement* provides the number of execution of each Oracle spatial operator during the inference process with area of interest refinement.

We observe a decrease in both of the spatial ontology rules executions and the number of DBMS spatial operator executions. For example, considering 250 dives, in the normal case of inference, Figure 9.3, the number of executions of the spatial ontology rules is 1 000 000 and the number of DBMS spatial operator calls is 125 000. However in the refinement case Figure 10.3, the number of executions of the spatial ontology rules is 130 000 and the number of DBMS spatial operator calls is 16 000. We therefore obtained a positive impact by applying the area of interest refinement.

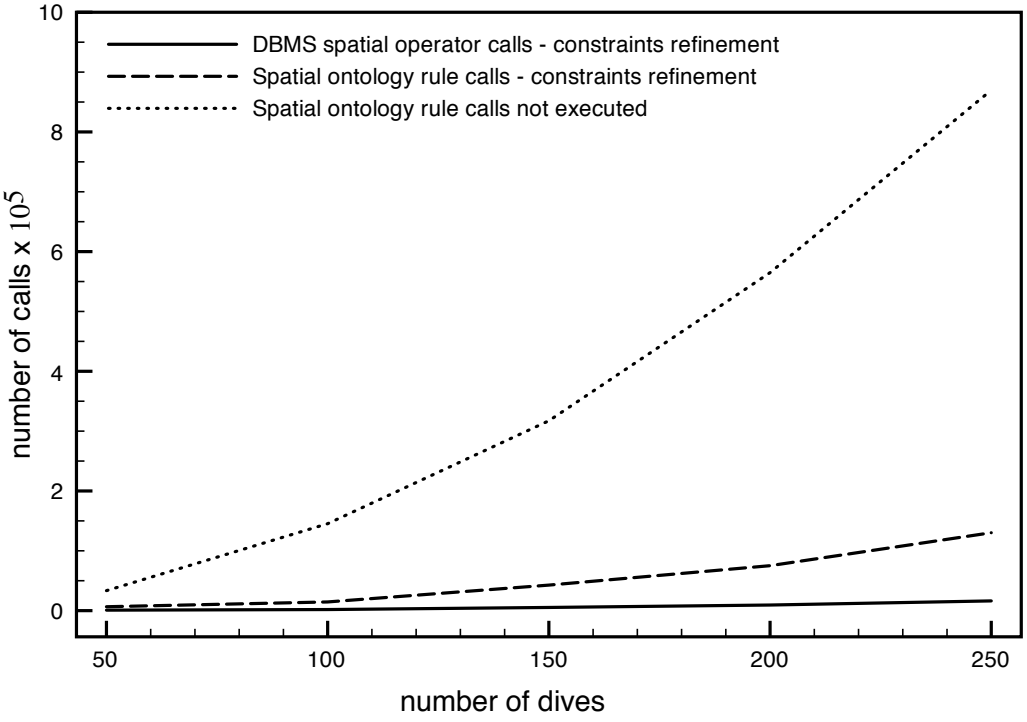

FIGURE 10.3 – Enhancement of the spatial ontology inference with constraints refinement

Secondly, we evaluate the proposed passes refinement. The experimental results are shown in Figure. 10.4. The impact results are shown by the three following experiments :

1. *Spatial ontology rule calls* presents the number of the spatial ontology rule calls during the normal inference process, without any refinement ;
2. *Spatial ontology rule calls - passes refinement* displays the number of the spatial ontology rule calls when applying the passes refinement ;
3. *Spatial ontology rule calls - passes and constraints refinement* provides the number of the spatial ontology rule calls when applying the passes and area of interest refinements together.

We observe a decrease in the number of the spatial ontology rule calls while applying the passes refinement and while applying the constraints and the passes refinements together. For example, considering 300 dives, in the normal case of inference, the number of the spatial ontology rule calls is 2 000 000. However in the passes refinement case, the number of the spatial ontology rule calls is 500 000.

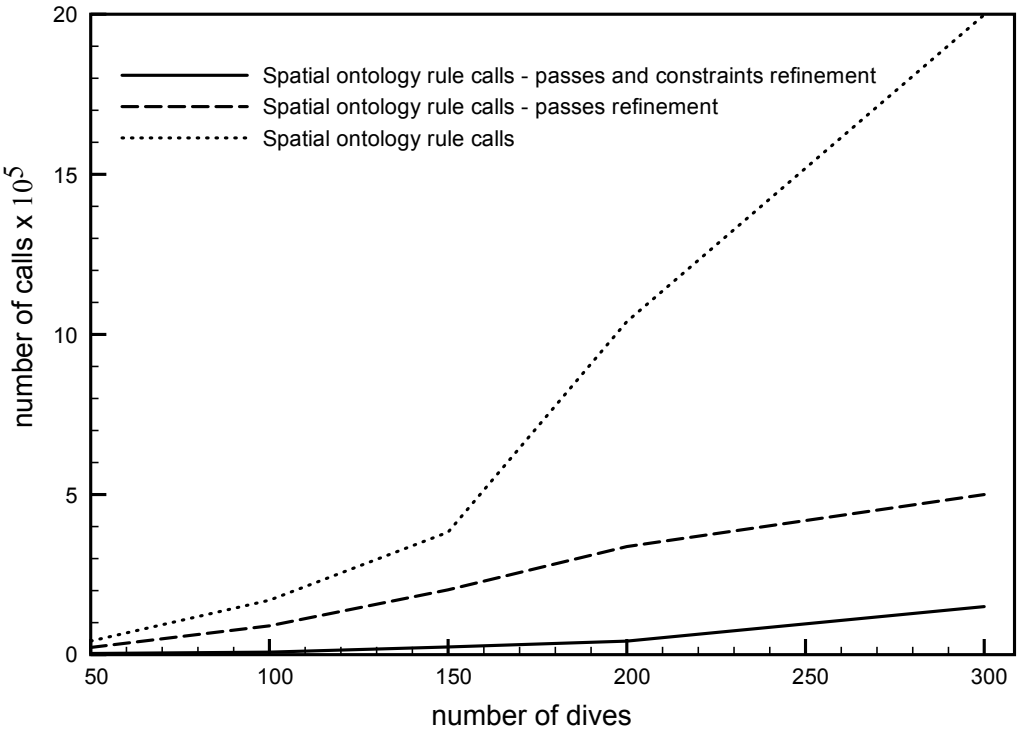

FIGURE 10.4 – Evaluation of the spatial ontology inference over the proposed refinement

Figure. 10.5 shows time execution of the spatial rules and refined spatial rules. The inference mechanism takes 126 000 seconds to execute the spatial rules, while around 34 000 seconds to execute spatial rules over the passes refinement and even less for both passes and constraints refinements. Finally, the refinements over spatial rules effects positively the computation complexity of the inference process.

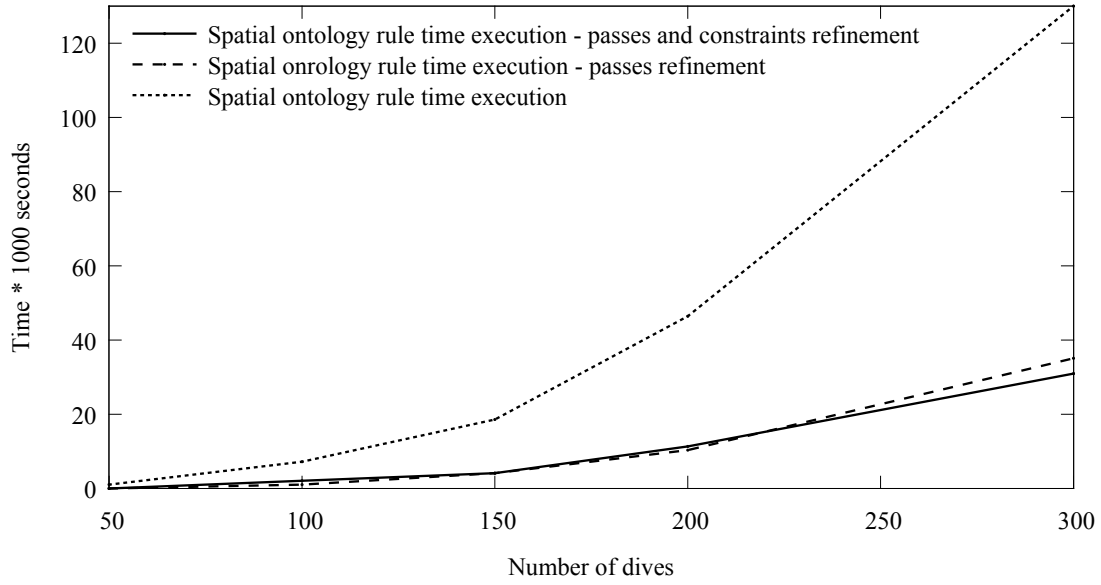

FIGURE 10.5 – Time execution of the spatial ontology inference over spatial refined rules

10.4 Spatio-temporal inference refinement

In this section, we perform experiments with the aim of outlining the inference difficulties over all the needed ontologies and rules. We evaluate the impact of the two spatial refinements and the temporal neighbor refinement over the executions of the inference mechanism. The inference uses the eight spatial rules, the 13 temporal rules and the trajectory domain foraging rule. For evaluating the constraints refinement and temporal neighbor refinement, the candidate depends on the application domain and can be estimated after considering the data statistical dispersion. Related to the seal trajectory domain and to our domain knowledge, we set the temporal neighbor between two sequences at five minutes (300 seconds) and the constraints to 500 meters.

Figure 10.6 shows the number of executions of the inference over the normal and the refined temporal and spatial rules. In our experiments, we consider sets of real GPS seal trajectory data. The evaluation curves is given by the number of dives. The vertical axis measures the number of executions (calls) needed to compute the inference mechanism for each considered number of dives.

For example, for 500 dives, the inference needs 6 000 000 executions to be normally computed over the 22 spatio-temporal rules. Over the 22 refined rules, the inference needs 2 000 000 executions to be computed. In term of the inferred triple, Fig. 10.7 shows 5 300 000 triples for the normal execution and 3 400 000 triples over the 22 spatio-temporal refined rules.

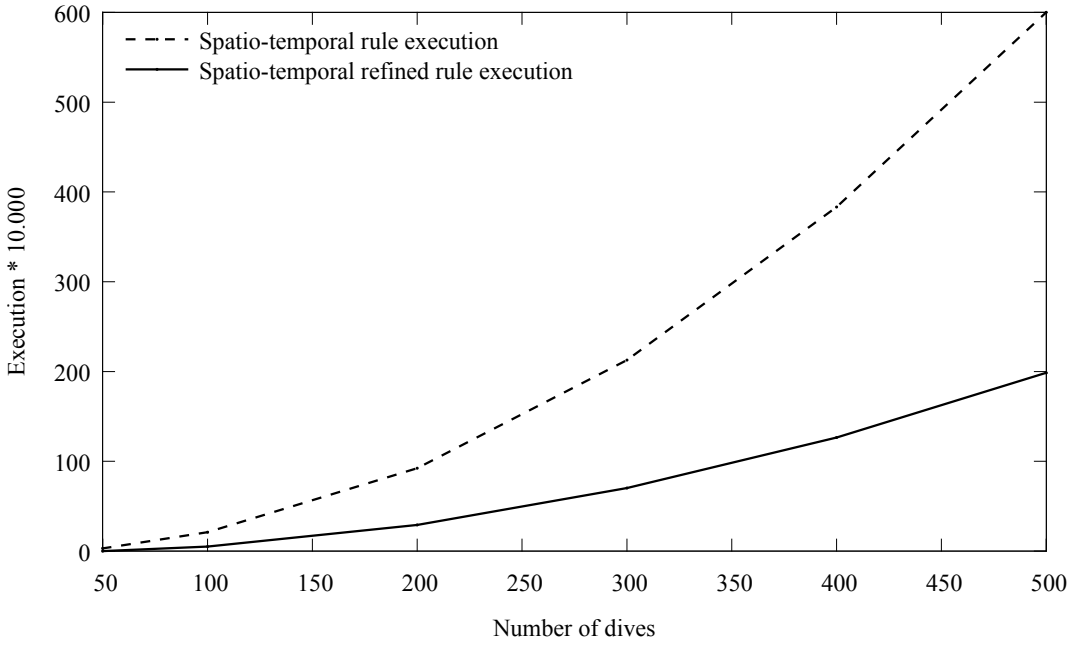

FIGURE 10.6 – Computation the inference over the domain, spatial and temporal rules

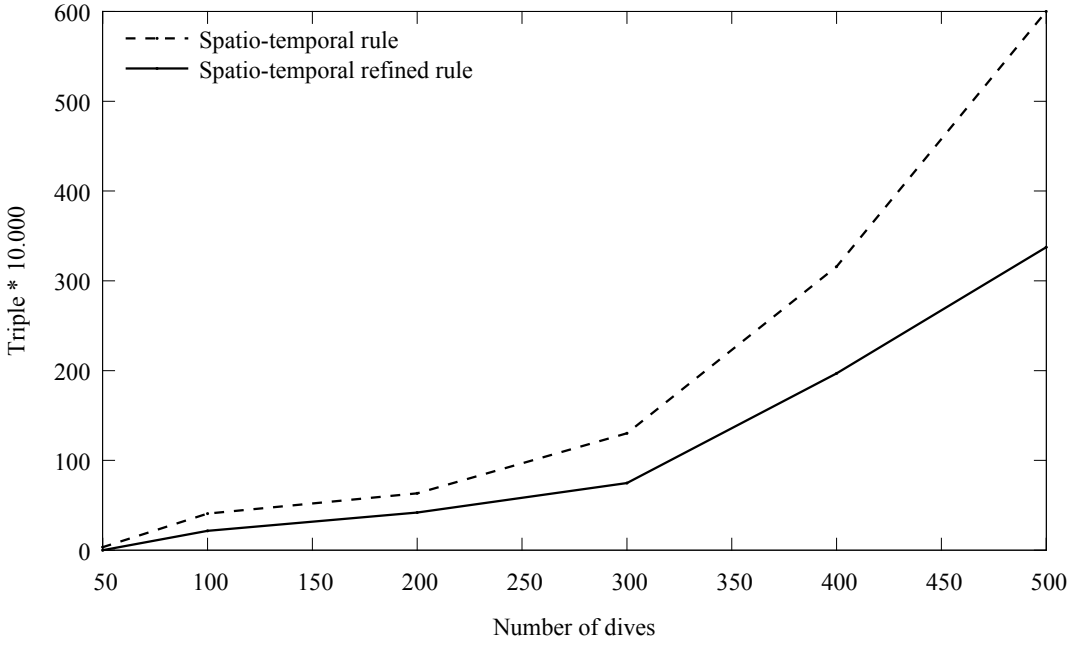

FIGURE 10.7 – Triple inferred over the domain, spatial and temporal rules

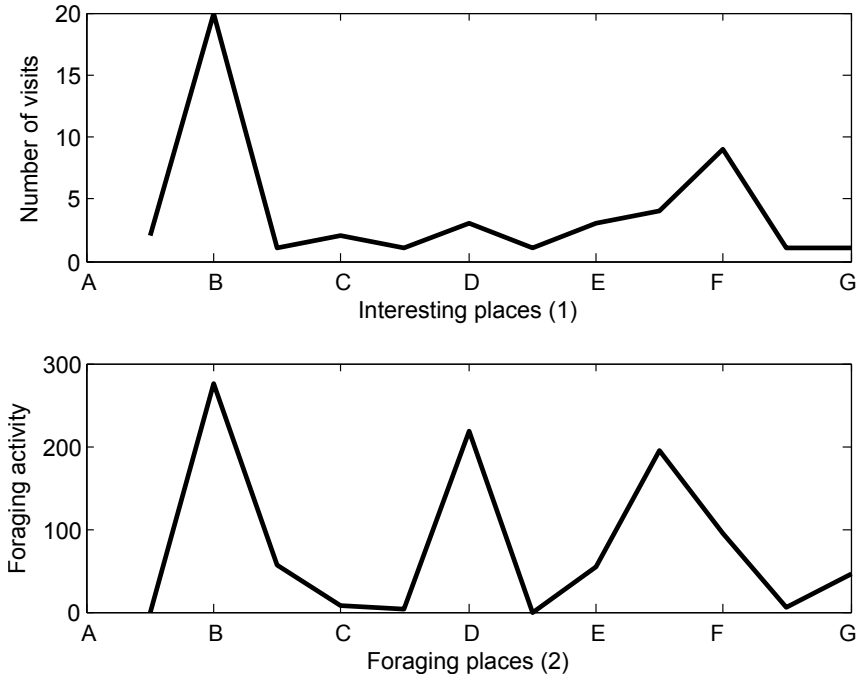

FIGURE 10.8 – Interesting places and foraging places

10.5 Two-tier inference refinement

To analyze our data, we consider the same datasets in Sect. 9.2. We pass these data to the ARS algorithm. This algorithm analyzes the data and gives as output the places and their visits, as shown in Fig. 10.8 interesting places (1). For example, the "place B" has 20 visits from the moving object, therefore may contain an interesting information related to the seal's activities.

Figure 10.9 shows the evaluation over the two-tier inference refinement with temporal neighbor refinement. This figure shows the results in term of number of triples generated by the inference. For example, for 450 dive, the inference mechanism generates 400 000 triples over the two-tier inference and the temporal neighbor refinements, while it generates 1 100 000 triples over the temporal neighbor refinement.

For evaluating the foraging places in Fig. 10.8 foraging places (2), we pass the coordinates of these places to our biologist to evaluated. The biologist finds that these are places where direct (visual) observations of seals (and dolphins) have been made in the past, and they correspond to places that are also known by local fishermen to be good for fishing thanks to the tidal current and the sediment type. Figure 10.10 is a map showing the coastline and the boundaries of the marine park in western Brittany with the evaluated foraging places. This information included in the map will be forwarded to the managers of the marine park, so that they take it into account for the management of interactions between seals and human activities.

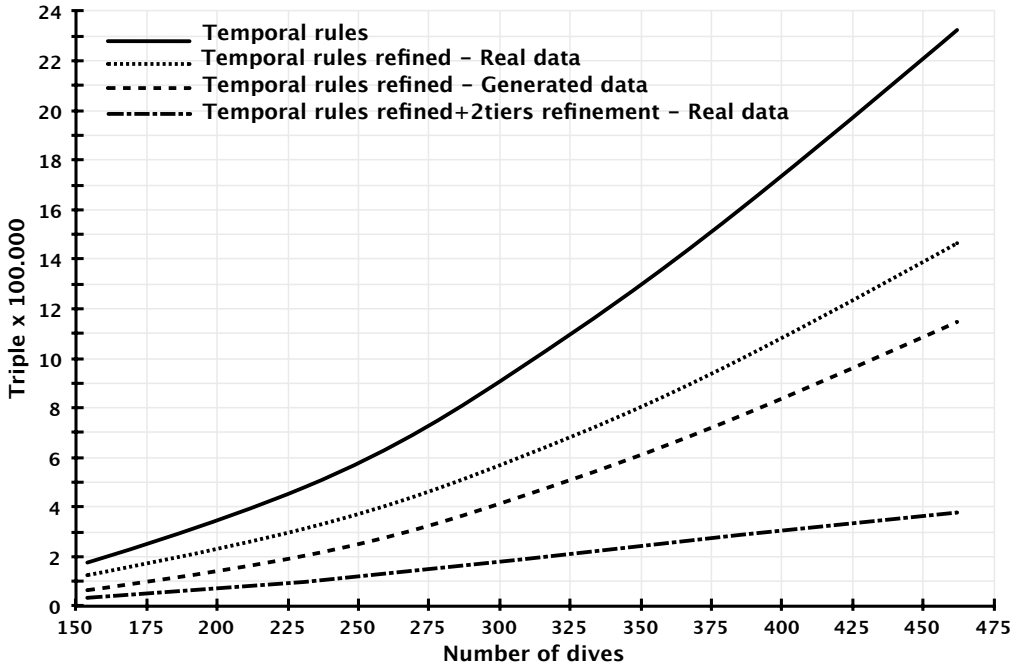

FIGURE 10.9 – Enhancement over the two-tier inference refinement

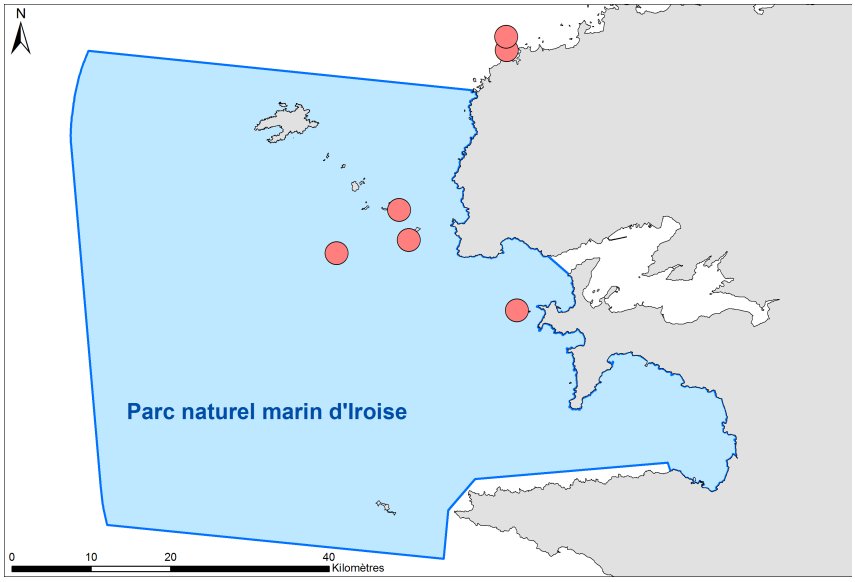

FIGURE 10.10 – marine park in western Brittany with the evaluated foraging places

TABLE 10.1 – Results of time neighbor refinement

Rules	**Data**	**Time** (hour)	**Triple** * 10^3	**Inference enhancement**
Temporal	450	16.6	2 200	2 times
Temporal neighbor	450	8	1 100	

TABLE 10.2 – Results of spatial neighbor and passes refinements

Rules	**Data**	**Spatial rule** execution * 10^3	**Time** (hour)	**Execution enhancement**	**Time enhancement**
Spatial	300	2 000	35	15 times	-
Spatial neighbor	300	130	-		
Spatial passes	300	500	9	4 times	$<$ 4 times
Spatial passes_neighbor	300	150	8.7	13 times	$>$ 4 times

10.6 Discussion

Table 10.1 shows the summary of the inference experiments over the time rules and the time refined rules (time neighbor refinement). This comparison shows 2 time enhancement over the time computation, memory and space storage of the inference.

Table 10.2 shows the summary of the inference experiments over the spatial rules and the spatial refined rules (spatial neighbor refinement and spatial passes refinement). In term of memory and space storage, this comparison shows 7 time enhancement over the number of executions the rules using the neighbor refinement and 4 time enhancement using the passes refinement, moreover 7 time enhancement using the passes and neighbor refinements. In term of the time computation, this comparison shows 4 time enhancement using passes and neighbor refinements.

Table 10.3 shows the summary of the inference experiments over the time neighbor rules and the two-tier inference refinements. This comparison shows 2 time enhancement over the memory and space storage of the inference mechanism.

10.7 Conclusion

In this chapter, we evaluated all our contributions proposed to solve the issue of inference complexity. Experiments evaluated the impact of the ontology inference refinements. Firstly, we performed experiments over the temporal inference refinement, the spatial inference refinement and then, the spatio-temporal inference refinements. Overall, we discussed the enhancement made by these refinements in term of reducing time computation, memory and space storage of

TABLE 10.3 – Results of two-tier inference refinement

Rules	**Data**	**Triple * 10^3**	**Inference enhancement**
Temporal neighbor	450	1 100	3 times
$Two-tier\ inference$	450	400	

the inference complexity. Finally, we performed experiments to evaluate the proposed application domain refinement. The two-tier inference refinement reduces the considered data into interesting data organized in interesting places. With this refinement, we were able to compute the inference over all the captured data, while, in the normal inference, we were able to compute the inference over just 500 raw trajectories.

Conclusion and perspectives

Conclusion

Trajectories are usually available as raw data. The data lack semantics which is fundamental for their efficient use. Our work is based on an ontology modeling approach for semantic trajectories. The trajectory approach consists of multiple separated ontologies : a general trajectory model, a mobile object model and a semantic model. However, the domain part of the trajectory ontology focuses on mobile object's characteristics and its trajectory's activities. Then, an application domain trajectory model is proposed consisting of a domain model. This model should be integrated with the semantic trajectory model. Considering a trajectory as a spatio-temporal concept, the semantic trajectory model must map to temporal and spatial models. For mapping to a temporal model, we reuse the W3C OWL-Time ontology. For mapping to a spatial model, we reuse a spatial ontology based on the OpenGIS Simple Features Interface Standard (SFS). We apply our modeling approach to an application domain : marine mammal tracking, in particularly seal trajectories. We model the application domain with a seal ontology. We map this ontology to our trajectory ontology.

We implement the declarative and the imperative parts of the ontologies. In this implementation, we consider RDF triple store. Technically, we use Oracle Semantic Data Technologies. We use Oracle RDF data for modeling, storing, loading and querying data. While our model is based on multiple ontologies, we map our trajectory ontology to the extended models (temporal and spatial models) and to the application domain ontology using OWL and RDFS constructions. To populate the ontologies, we add seal data within the ontology. The seal trajectory data is provided as files, then we have a problem of integrating them with the ontology model. So, we load this data into a relational database, then, we use the D2RQ mapping engine to map the relational data into RDF graph triples. Moreover, we address the semantic gap between the model and the captured data. Then, application domain knowledge experts are defined in the seal trajectory ontology and connected to the semantic model in the semantic trajectory model.

The objective is to annotate data with the considered knowledge. Over huge data, an ontology inference mechanism is used as an automatic annotation. This mechanism derives new semantics from existing information using additional knowledge. These knowledge could be in a form of rules either axioms or defined by the user. Fundamental axioms are the reasoner's rules, such as RDF, RDFS, OWLPrime or others. User-defined rules are knowledge related to users' needs. The user rules are the domain, temporal and spatial knowledge. Moving object activities are applied as rules. Temporal relationships, specifically the 13 Allen relationships, are implemented as temporal rules. Spatial relationships, specifically OGC topological relationships, are implemented as spatial rules. The ontology inference mechanism is computed to annotate the data with the considered user-defined rules. These annotations are data labeled with semantic annotations as triples. We clearly address that computing the inference using axioms is completely different from using user-defined rules. Oracle presents an experiment over 133 million data using OWLPrime reasoner rules and it takes 7 hours. We preform an experiment using temporal rules, however for 450 data, it takes 16 hours. This is because Oracle inference engine does not integrate OWLPrime and user-defined rule inference components similarly. Behind the scene, Oracle first run OWLPrime until a closure occurs. Then Oracle run user-defined rules until a closure occurs, and repeat the whole process till no new triple is generated from either component. Therefore, the time computation increases as the vocabulary becomes increasingly expressive, especially

when using OWLPrime with user-defined rules. The inference mechanism therefore becomes an expensive mechanism in terms of time computations and space storage.

Our objective is to reduce this inference complexity. We try to understand technicality the work of the inference engine. We evaluate the inference mechanism using different engines. Then, we compute the inference mechanism over our model with different experiments. We compute the inference using the domain and temporal rules, then the domain and spatial rules, and finally the domain, temporal and spatial rules together. Moreover, we perform experiments to understand the way the inferred data is separated and saved in the database. The difficulty is that whenever we intervene into this schema, the whole database becomes invalid. Then, we have to delete everything and rebuild the database again. Moreover, the problem of Oracle inference mechanism is not an incremental process when using user-defined rules.

After all these experiments and in order to reduce the inference complexity, we propose three optimizations. We suggest some domain constraints, temporal and spatial neighbor refinements. Using these refinements, we modify directly the rules which influence positively the computation of the inference. Moreover, we propose a reduction of the repetition of the inference computation. In this case, the inference results are saved in our index, not on the standard inference index. Finally, we visualize seal trajectories to view the movement of a moving object and understand its behavior. From this visualization, we define a refinement specifically for the domain application. This refinement reduces the considered data into interesting data grouped into place-of-interest. We call it two-tier inference filters. In other words, two distinct operations are performed to enhance the inference : primary and secondary filter operations. The primary filter analyzes the trajectory data into places of interest. The secondary filter computes the ontology inference over these places. The inference computation filters the interesting places into domain activity places which are the results of the annotation process. In this case, we are able to consider all the captured data in the annotation process.

We can sum up the main contributions of this work as following :
- Using an ontology modeling approach for semantic trajectories integrated with external background models ;
- Connecting separated and different needed ontologies ;
- Computing the inference mechanism over semantic trajectories to enrich high-level knowledge, then answer user queries ;
- Addressing the computation complexity of the inference mechanism in terms of time computations and space storage ;
- Reducing the inference complexity by some refinements.

We evaluate our contributions on real-world trajectory data. The experimental results highlight the positive impact of the proposed refinements. In the neighbor and repetition refinements, we approximately reduce half of the time computation of the inference mechanism. In evaluating the two-tier filters, the results show that we are able in this case to consider all the captured data, not just part of them like in the normal inference mechanism. In other words, we are able to answer queries over much more data than the normal ontology inference.

Perspectives

Our approach computes the inference mechanism over our spatio-temporal trajectory data model. Our model deals with 2D+1/2 scientific problems with temporal and spatial dimensions. However, there is always a way to improve. A global enhancement could be upgrading our approach to consider any spatio-temporal data over the trajectory model. Moreover, we look forward to take into consideration three-dimensions in the trajectory model.

Based on our experiments for understanding the mechanism of the inference process, we can specify how a reasoner should compute the inference. So, we design our own inference engine taking into account the following considerations :

- A logical order to execute the rules ;
- The reflexive, symmetric, transitive rules when executing a rule. This means that when we execute a rule, the output should be its result with its reflexive, symmetric, transitive results, without taking time to re-execute its reflexive, symmetric, transitive rules ;
- The composition rules when executing the rules [14, 15].

We highlight parts in our approach that need improvement :

- In our approach, we are able to answer experts queries. Therefore, providing a query interface gives possibilities to the end user to specify an activity, an assigned moving object and all the other necessary parameters : time, zone or others. Over all, a query optimization should be provided, as well as a query transformation into SPARQL or SQL queries to be applied over our existing model ;
- Our semantic annotation process is applied over trajectories. However, the annotation process labels parts of a trajectory with the considered knowledge. These parts are the `Sequences` of a trajectory. Meaning that, the results of the inference mechanism are the `Sequences` labeled with activities of a moving object or other considered rules. However, the objective could be annotating the whole trajectory with one knowledge. For example, if a trajectory has resting sequences, traveling sequences and foraging sequences, then we need a decision about the activity of the whole trajectory. The objective of the trajectory of a mobile object in a specific day is a foraging activity ;
- In our application domain, the global objective is to determine foraging parts or foraging trajectories. So, the foraging rule could be enhanced by a global view over the whole trajectory. Therefore, we can define the following scenario and the following definitions :
  Definition 12 (Trip) *. A trip is a travel at sea between two haulouts of the seals (time spent on land). This definition is provided by the biologist Jason in [73].*
  Related to our application domain, we define three kinds of a trip, shown in Figure 10.11 :
  Definition 13 (Return trip (rock)) *. Return (rock) trip is a trip of a moving object where it returns to the same specific rock which it started from.*
  Definition 14 (Return trip (area)) *. Return (area) trip is a trip of a moving object where it returns to the same specific area which it started from.*
  Definition 15 (Travel trip) *. Travel trip is a trip of a moving object from one haulout to a different one.*

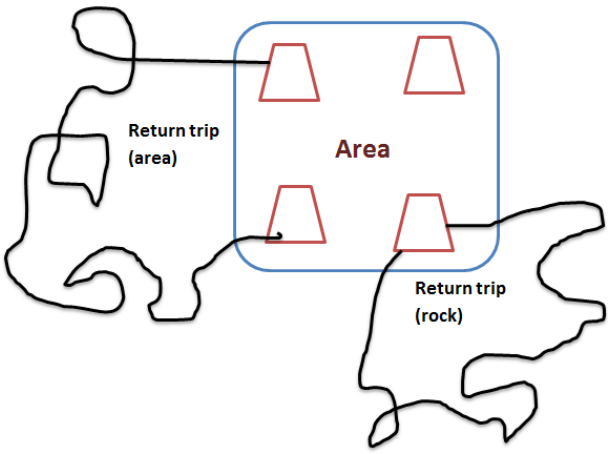

FIGURE 10.11 – kinds of a trip

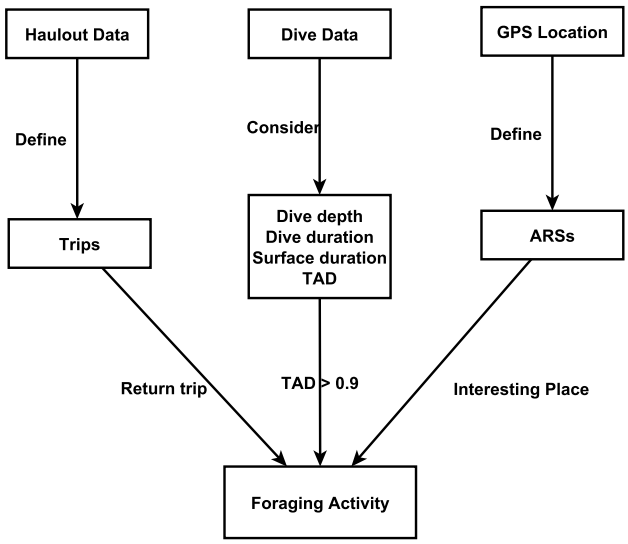

FIGURE 10.12 – Combination features for the `Foraging` activity

Overall, defining a foraging activity within a trajectory could combine three parts. Figure 10.12 presents the combination of the following parts to make a decision about the foraging activity within the whole trajectory.

– Specific features of dives related to a foraging activity, defined in Sect. 8.4 ;
– Validating dives' features in interesting places (`Area-Restricted Searches (ARSs)`), detailed in Sect. 8.5 ;
– Having return trip within the haulout data in a trajectory.

Bibliography

[1] D J. Abadi, A Marcus, S R. Madden, and K Hollenbach. Scalable semantic Web data management using vertical partitioning. In *Proceedings of the 33rd International conference on very large data bases, VLDB*, pages 411–422, 2007.

[2] Sven Abels, Liane Haak, and Axel Hahn. Identification of common methods used for ontology integration tasks. In *Proceedings of the First International Workshop on Interoperability of Heterogeneous Information Systems*, IHIS '05, pages 75–78. ACM, 2005.

[3] B. V. Aduna. The SeRQL query language. In *Journal on Data Semantics III*, Lecture Notes in Computer Science. User Guide for Sesame 2.1, 2008.

[4] Allegrograph, 2005. `http://www.franz.com/agraph/allegrograph/`.

[5] Wu Allan. Scalable RDBMS based inference engine for RDFS OWL pellet. Oracle New England Development Center, 2011.

[6] James F. Allen. Maintaining knowledge about temporal intervals. *Communications of the ACM*, pages 832–843, 1983.

[7] L.O. Alvares, V Bogorny, B Kuijpers, A.F. Macedo, B Moelans, and A Vaisman. A model for enriching trajectories with semantic geographical information. In *Proceedings of the 15th annual ACM international symposium on Advances in geographic information systems*, pages 1–8. ACM, 2007.

[8] David Aumueller, Hong-Hai Do, Sabine Massmann, and Erhard Rahm. Schema and ontology matching with COMA++. In *Proceedings of the 2005 ACM SIGMOD International Conference on Management of Data*, SIGMOD '05, pages 906–908. ACM, 2005.

[9] Franz Baader, Diego Calvanese, Deborah L. McGuinness, Daniele Nardi, and Peter F. Patel-Schneider. *The Description Logic Handbook : Theory, Implementation, and Applications*. Cambridge University Press, 2003.

[10] Franz Baader, Ian Horrocks, and Ulrike Sattler. Description logics. In *Handbook on Ontologies*, International Handbooks on Information Systems, pages 3–28. Springer Berlin Heidelberg, 2004.

[11] M Baglioni, J Macedo, C Renso, and M Wachowicz. An ontology-based approach for the semantic modelling and reasoning on trajectories. In *Advances in Conceptual Modeling - Challenges and Opportunities*, pages 344–353. Springer Berlin/Heidelberg, 2008.

[12] Miriam Baglioni, Jose Antonio Fernandes de Macedo, Chiara Renso, Roberto Trasarti, and Monica Wachowicz. Towards semantic interpretation of movement behavior. In *Advances in GIScience*, Lecture Notes in Geoinformation and Cartography, pages 271–288. Springer Berlin Heidelberg, 2009.

[13] Chitta Baral. *Knowledge Representation, Reasoning, and Declarative Problem Solving*. Cambridge University Press, 2003.

[14] Sotiris Batsakis and Euripides G. M. Petrakis. SOWL : A framework for handling spatio-temporal information in OWL 2.0. In *RuleML Europe*, Lecture Notes in Computer Science, pages 242–249. Springer, 2011.

[15] Sotiris Batsakis and EuripidesG.M. Petrakis. Representing temporal knowledge in the semantic Web : The extended 4D fluents approach. In *Combinations of Intelligent Methods and Applications*, Smart Innovation, Systems and Technologies, pages 55–69. Springer Berlin Heidelberg, 2011.

[16] Robert Battle and Dave Kolas. Enabling the GeoSpatial semantic Web with parliament and GeoSPARQL. *Semantic Web*, pages 355–370, 2012.

[17] Konstantina Bereta, Panayiotis Smeros, and Manolis Koubarakis. Representation and querying of valid time of triples in linked Geospatial data. In *The Semantic Web : Semantics and Big Data*, Lecture Notes in Computer Science, pages 259–274. Springer Berlin Heidelberg, 2013.

[18] S Bergamaschi, S Castano, M Vincini, and D Beneventano. Semantic integration of heterogeneous information sources. *Data Knowledge Engineering*, pages 215–249, 2001.

[19] Jacob Berlin and Amihai Motro. Database schema matching using machine learning with feature selection. In *Proceedings of the 14th International Conference on Advanced Information Systems Engineering*, CAiSE '02, pages 452–466. Springer-Verlag, 2002.

[20] Tim Berners-Lee, J Hendler, and O Lassila. The semantic Web - anew form of Web content that is meaningful to computers will unleash a revoluation of new possibilities. In *Scientific American*, page 284, 2001.

[21] Christian Bizer and Richard Cyganiak. D2RQ - lessons learned. *W3C Workshop on RDF Access to Relational Databases*, 2007.

[22] V Bogorny, C Heuser, and L Alvares. A conceptual data model for trajectory data mining. In *Geographic Information Sci.*, pages 1–15. Springer Berlin/Heidelberg, 2010.

[23] Azedine Boulmakoul, Lamia Karim, and Ahmed Lbath. Moving object trajectories meta-model and spatio-temporal queries. In *International Journal of Database Management Systems (IJDMS)*, pages 35–54, 2012.

[24] Jeen Broekstra, Arjohn Kampman, and Frank Harmelen. Sesame : A generic architecture for storing and querying RDF and RDF Schema. In *The Semantic Web — ISWC 2002*, Lecture Notes in Computer Science, pages 54–68. Springer Berlin Heidelberg, 2002.

[25] E. Camossi, P. Villa, and L. Mazzola. Semantic-based anomalous pattern discovery in moving object trajectories. *CoRR*, pages 1–20, 2013.

[26] Eliseo Clementini, Paolino Di Felice, and Peter van Oosterom. A small set of formal topological relationships suitable for end-user interaction. In *Proceedings of the Third International Symposium on Advances in Spatial Databases*, SSD 93, pages 277–295. Springer-Verlag, 1993.

[27] Eliseo Clementini, Jayant Sharma, and Max J. Egenhofer. Modelling topological spatial relations : Strategies for query processing. *Computers and Graphics*, pages 815–822, 1994.

[28] E. F. Codd. Data models in database management. *SIGPLAN Not.*, pages 112–114, 1980.

[29] Olteanu Dan, Steinhaus Robin, Furche Tim, and Ferm Emanuel. G-store : A storage manager for graph data, 2010. `http://g-store.sourceforge.net/`.

[30] Beckett David. RDF 1.1 N-Triples, a line-based syntax for an RDF graph. W3C Recommendation, 2008.

[31] ISO/IEC 9075-2 :1999. In *Information Technology Database Languages SQL Part 2 : Foundation (SQL/Foundation)*, 1999.

[32] María del Carmen Suárez-Figueroa, Asunción Gómez-Pérez, Enrico Motta, and Aldo Gangemi. *Ontology Engineering in a Networked World*. Springer, 2012.

[33] Pfoser Dieter and Theodoridis Yannis. Generating semantics-based trajectories of moving objects. In *International workshop on emerging technologis for Geo-based applications*, pages 59–76, 2000.

[34] AnHai Doan and Alon Y. Halevy. Semantic-integration research in the database community. *AI Mag.*, pages 83–94, 2005.

[35] M. J. Egenhofer and J.R. Herring. A mathematical framework for the definition of topological relationships. In *Fourth International Symposium on Spatial Data Handling*, pages 803–813, 1990.

[36] M. Egenhofer and J. Herring. Categorizing binary topological relationships between regions, lines, and points in Geographic databases. Technical Report. Department of Surveying Engineering, University of Maine, 1991.

[37] Thomas Eiter, Georg Gottlob, and Heikki Mannila. Disjunctive datalog. *ACM Transactions on Database Systems*, pages 364–418, 1997.

[38] Thomas Eiter, Giovambattista Ianni, Axel Polleres, Roman Schindlauer, and Hans Tompits. Reasoning with rules and ontologies. In *In Reasoning Web 2006*, pages 93–127. Springer, 2006.

[39] Rahm Erhard and Bernstein Philip. A survey of approaches to automatic schema matching. *The VLDB Journal*, pages 334–350, 2001.

[40] Prud'hommeaux Eric and Seaborne Andy. SPARQL query language for RDF. W3C recommendation, 2008. `http://www.w3.org/TR/rdf-sparql-query/`.

[41] Laurent Etienne. *Motifs spatio-temporels de trajectoires d'objets mobiles, de l'extraction à la détection de comportements inhabituels. Application au trafic maritime.* PhD thesis, Bretagne university, 2011.

[42] J Euzenat and P Shvaiko. *Ontology matching*. Springer-Verlag, 2007.

[43] M. A. Fedak, P. Lovell, and S. M. Grant. Two approaches to compressing and interpreting time-depth information as collected by time-depth recorders and satellite-linked data recorders. *Mar Mamm Sci*, pages 94–110, 2001.

[44] Dieter Fensel, Katia Sycara, and John Mylopoulos. Web ontology language requirements w.r.t expressiveness of taxonomy and axioms in medicine. In *The Semantic Web - ISWC 2003*, Lecture Notes in Computer Science, pages 180–194. Springer Berlin Heidelberg, 2003.

[45] Filament : Graph management toolkits project, 2010. `http://filament.sourceforge.net/`.

[46] Charles L. Forgy. Expert systems. chapter Rete : A Fast Algorithm for the Many Pattern/Many Object Pattern Match Problem, pages 324–341. IEEE Computer Society Press, 1990.

[47] Antoniou G., Damnasio C. V., Grosof B., Horrocks I., Kifer M., Maluszynski J., and Patel-Schneider P. F. Combining rules and ontologies : A survey. Technical Report. Linköping University, 2005.

[48] Michael Gelfond and Vladimir Lifschitz. Classical negation in logic programs and disjunctive databases. *New Generation Computing*, pages 365–385, 1991.

[49] GeoPKDD. Geographic privacy-aware knowledge discovery and delivery. In *Coordinator : KDDLAB, Knowledge Discovery nad Delivery Laboratory, ISTI-CNR and University of Pisa.*, 2005. `http://www.geopkdd.eu`.

[50] A. Miller George. WordNet : A lexical database for english. *Communications of the ACM*, pages 39–41, 1995.

[51] Gooddata developers, 2007. `https://developer.gooddata.com/docs/data-modeling`.

[52] Graph Connect, 2012. Conference for graph databases and applications, and features Neo4j. `http://www.graphconnect.com/`.

[53] Bernardo Cuenca Grau, Ian Horrocks, Boris Motik, Bijan Parsia, Peter Patel-Schneider, and Ulrike Sattler. OWL : 2 : The next step for OWL. In *In, Web Semantics : Science, Services, and Agents on the World Wide Web*, W3C Recommendation, pages 309–322, 2008.

[54] Karvounarakis Gregory, Christophides Vassilis, Plexousakis Dimitris, and Alexaki Sofia. Querying community Web portals. Technical report, Technical report, Institute of Computer Science, FORTH, Heraklion, Greece, 2000. `http://www.ics.forth.gr/proj/isst/RDF/RQL/rql.pdf`.

[55] N.L. Griffin and F. D. Lewis. A rule-based inference engine which is optimal and VLSI implementable. Technical report. University of Kentucky, Department of Computer Science, 1989.

[56] Thomas R. Gruber. A translation approach to portable ontology specifications. *Knowledge Acquis.*, pages 199–220, 1993.

[57] B. Guc, M. May, Y. Saygin, and C. Korner. Semantic annotation of GPS trajectories. In *11th AGILE International Conference on Geographic Information Science, Girona, Spain*, pages 1–9, 2008.

[58] Ralf Hartmut Guting, Michael H. Bohlen, Martin Erwig, Christian S. Jensen, Nikos A. Lorentzos, Markus Schneider, and Michalis Vazirgiannis. A foundation for representing and querying moving objects. *ACM Trans. Database Syst.*, pages 1–42, 2000.

[59] Ralf Hartmut Güting. Graphdb : Modeling and querying graphs in databases. In *Proceedings of the 20th International Conference on Very Large Data Bases*, VLDB '94, pages 297–308. Morgan Kaufmann Publishers Inc., 1994.

[60] Ralf Hartmut Güting. An introduction to spatial database systems. *The VLDB Journal*, pages 357–399, 1994.

[61] Volker Haarslev and Ralf Möller. Racer system description. In *Proceedings of the First International Joint Conference on Automated Reasoning*, IJCAR '01, pages 701–706. Springer-Verlag, 2001.

[62] Alon Y. Halevy. Structures, semantics and statistics. In *Proceedings of the Thirtieth international conference on Very large data bases - Volume 30*, VLDB 04, pages 4–6, 2004.

[63] Jonathan Hayes and Claudio Gutierrez. Bipartite graphs as intermediate model for RDF. In *The Semantic Web – ISWC 2004*, Lecture Notes in Computer Science, pages 47–61. Springer Berlin Heidelberg, 2004.

[64] Horst H.J. semantics in completeness, decidability and complexity of entailment for RDF Schema and a semantic extension involving the OWL vocabulary. *Journal of Web Semantics 3*, pages 79–115, 2005.

[65] J. R. Hobbs and P. Fang. Time ontology in OWL. W3C recommendation, 2006. `http://www.w3.org/2006/time`.

[66] Rinke Hoekstra, Joost Breuker, Marcello Di Bello, and Alexander Boer. The LKIF core ontology of basic legal concepts. In *Proceedings of the Workshop on Legal Ontologies and Artificial Intelligence Techniques (LOAIT)*, 2007.

[67] Ian Horrocks, Ulrike Sattler, and Stephan Tobies. Practical reasoning for expressive description logics. In *Proceedings of the 6th International Conference on Logic Programming and Automated Reasoning*, LPAR '99, pages 161–180. Springer-Verlag, 1999.

[68] Ian Horrocks, Peter F. Patel-Schneider, and Frank Van Harmelen. From SHIQ and RDF to OWL : The making of a Web Ontology Language. *Journal of Web Semantics*, 2003.

[69] Ian Horrocks, Peter F. Patel-Schneider, Harold Boley, Said Tabet, Benjamin Grosof, and Mike Dean. SWRL : A semantic web rule language combining OWL and RuleML. W3C member submission, World Wide Web Consortium, 2004. `http://www.w3.org/Submission/SWRL`.

[70] Yingjie Hu, Krzysztof Janowicz, David Carral, Simon Scheider, Werner Kuhn, Gary Berg-Cross, Pascal Hitzler, Mike Dean, and Dave Kolas. A Geo-ontology design pattern for semantic trajectories. In *Spatial Information Theory*, Lecture Notes in Computer Science, pages 438–456. Springer International Publishing, 2013.

[71] Horrocks Ian, Fensel Dieter, Broekstra Jeen, Decker Stefan, Erdmann Michael, Goble Carole, Harlemen Frank van, Klein Michel, Staab Steffen, Studer Rudi, and Motta Enrico. The ontology inference layer OIL, 2000.

[72] ISO/TC_211. Geographic information – temporal schema, ISO 19108, 2002.

[73] Jason and Matthiopoulos. The use of space by animals as a function of accessibility and preference. *Ecological Modelling*, pages 239–268, 2003.

[74] R. H Jerry and P Feng. An ontology of time for the semantic Web. In *ACM Transactions on Asian Language Information Processing*, pages 66–85, 2004. `http://www.w3.org/2006/time`.

[75] Herring John R. OpenGIS implementation standard for Geographic information - simple feature access - part 2 : SQL option. *Open Geospatial Consortium Inc.*, 2011. `http://www.opengeospatial.org/legal`.

[76] Sequeda Juan, Tirmizi Syed Hamid, Corcho Oscar, and Miranker Daniel P. Survey of directly mapping SQL databases to the semantic Web. *Knowledge Eng. Review*, pages 445–486, 2011.

[77] Yong-Bin Kang, Yuan-Fang Li, and Shonali Krishnaswamy. Predicting reasoning performance using ontology metrics. In *The Semantic Web – ISWC 2012*, Lecture Notes in Computer Science, pages 198–214. Springer Berlin Heidelberg, 2012.

[78] Ryden Keith. OpenGIS implementation standard for Geographic information - simple feature access - part 1 :common architecture. *Open Geospatial Consortium Inc.*, 2010. `http://www.opengeospatial.org/pt/06-103r4`.

[79] Wilkinson Kevin, Sayers Craig, Kuno Harumi, and Reynolds Dave. Efficient RDF storage and retrieval in Jena2. In *Exploitting hyperlinks 349*, pages 35–43, 2003.

[80] Atanas Kiryakov, Damyan Ognyanov, and Dimitar Manov. OWLIM : a pragmatic semantic repository for OWL. In *Proceedings of the 2005 International Conference on Web Information Systems Engineering*, WISE'05, pages 182–192. Springer-Verlag, 2005.

[81] AndrewS. Knell and EdwardA. Codling. Classifying area-restricted search (ARS) using a partial sum approach. *Theoretical Ecology*, pages 325–339, 2012.

[82] Janowicz Krzysztof, Scheider Simon, Pehle Todd, and Hart Glen. Geospatial semantics and linked spatiotemporal data, past, present, and future. *Semantic Web - Interoperability, Usability, Applicability an IOS Press Journal*, pages 321–332, 2012.

[83] Logical data model. `http://www.learndatamodeling.com/ldm.php`.

[84] C. Lecluse, P. Richard, and F. Velez. O2, an object-oriented data model. In *Proceedings of the 1988 ACM SIGMOD International Conference on Management of Data*, SIGMOD '88, pages 424–433. ACM, 1988.

[85] V. I. Levenshtein. Binary codes capable of correcting deletions, insertions and reversals. *Soviet Physics Doklady*, pages 707–710, 1966.

[86] J. Lieberman, R. Singh, and C. Goad. W3C Geospatial ontologies. Technical Report. W3C incubator group report, 2007.

[87] Vladimir Lifschitz. Answer set planning. In *Logic Programming and Nonmonotonic Reasoning*, Lecture Notes in Computer Science, pages 23–37. Springer Berlin Heidelberg, 1999.

[88] Predoiu Livia and Stuckenschmidt Heiner. Probabilistic extensions of semantic Web languages - A survey. In *The Semantic Web for Knowledge and Data Management : Technologies and Practices*. Idea Group Inc, 2008.

[89] Nair Manoj. Spatial data in SQL server. *InfoSYS, White Paper*, 2011.

[90] Norbert Martínez-Bazan, Victor Muntés-Mulero, Sergio Gómez-Villamor, Jordi Nin, Mario-A. Sánchez-Martínez, and Josep-L. Larriba-Pey. Dex : High-performance exploration on large graphs for information retrieval. In *Proceedings of the Sixteenth ACM Conference on Conference on Information and Knowledge Management*, CIKM '07, pages 573–582. ACM, 2007.

[91] P Matthew. *A framework to support spatial, temporal and thematic analytics over semantic Web data*. PhD thesis, Wright State University, 2008.

[92] D. L. McGuinness and F. VanHarmelen. OWL Web Ontology Language overview. W3C recommendation, 2004.

[93] W Mefteh, J Malki, and A Bouju. De l'expérience à un patron de conception pour les trajectoires. In *SDH-SAGEO (Spatial Data Handling-Spatial Analysis and GEOmatics)*, pages 1–16, 2008.

[94] Wafa Mefteh. *Approche ontologique pour la modelisation et le raisonnement sur les trajectoires. Prise en compte des aspects thematiques, temporels et saptiaux.* PhD thesis, La Rochelle university, 2013.

[95] Xiaofeng Meng and Zhiming Ding. DSTTMOD : A discrete spatio-temporal trajectory based moving object database system. In *Database and Expert Systems Applications*, Lecture Notes in Computer Science, pages 444–453. Springer Berlin Heidelberg, 2003.

[96] Compton Michael, Barnaghi Payam, Bermudez Luis, Garcia-Castro Raul, Corcho Oscar, Cox Simon, Graybeal John, Hauswirth Manfred, Henson Cory, Herzog Arthur, Huang Vincent, Janowicz Krzysztof, Kelsey W. David, Le Phuoc Danh, Lefort Laurent, Leggieri Myriam, Neuhaus Holger, Nikolov Andriy, Page Kevin, Passant Alexandre, Sheth Amit, and Taylor Kerry. The SSN ontology of the W3C semantic sensor network incubator group. *Web Semantics : Science, Services and Agents on the World Wide Web*, pages 25 – 32, 2012.

[97] Sun Microsystems. The Upper Cyc Ontology in XTM. Technical report, Sun Microsystems, 2001.

[98] MODAP. Mobility, data mining and privacy. *EU Coordination Action, FET OPEN*, 2009. `http://www.modap.org/`.

[99] MOVE. European science foundation. European Cooperation in Science and Technologies (COST) Action IC0903 : Knowledge Discovery from Moving Objects (MOVE), 2009-2013. http://www.move-cost.info.

[100] Ian Niles and Adam Pease. Towards a standard upper ontology. In *Proceedings of the international conference on Formal Ontology in Information Systems - Volume 2001*, FOIS '01, pages 2–9. ACM, 2001.

[101] Natalya F. Noy. Semantic integration : A survey of ontology-based approaches. *SIGMOD Rec.*, pages 65–70, 2004.

[102] Oracle Spatial developer guide 11G Release 2 (11.2). Oracle documentation, 1996. http://docs.oracle.com/cd/E11882.

[103] Oracle Database Semantic Technologies Developers guide 11G Release 2. Oracle documentation, 2012. http://www.oracle.com/technology/tech/semantic-technologies.

[104] Erling Orri. Advances in Virtuoso RDF triple storage (bitmap indexing). 2006. http://virtuoso.openlinksw.com/dataspace/dav/wiki/Main.

[105] W Ouri, S Prasad, X Bo, Z Jutai, and C Sam. DOMINO : Databases for moving objects tracking. In *In ACM SIGMOD International Conference on Management of Data (SIGMod-99)*, pages 547–549, 1999.

[106] Christine Parent, Stefano Spaccapietra, and Esteban Zimanyi. *Conceptual Modeling for Traditional and Spatio-Temporal Applications : The MADS Approach*. Springer-Verlag New York, Inc., 2006.

[107] C Parent, S Spaccapietra, C Renso, G Andrienko, N Andrienko, V Bogorny, M Damiani, A Gkoulalas-divanis, J Macedo, N Pelekis, Y Theodoridis, and Z Yan. Semantic trajectories modeling and analysis. EU, FET OPEN, 2009 - 2012 Programme, Coordination Action type Project MODAP. ACM Computing Surveys, 2013. http://www.modap.org.

[108] Duarte Nuno Peralta, Helena Sofia Andrade N. P. Pinto, and Nuno J. Mamede. Reusing a time ontology. In *ICEIS (3)*, Enterprise Information Systems, pages 121–128. Kluwer Academic Publishers, 2003.

[109] Matthew Perry, Amit Sheth, Ismailcem Budak Arpinar, and Farshad Hakimpour. Geospatial and temporal semantic analytics. In *Information System Journal*, GIS 06, pages 1–14, 2002.

[110] Matthew Perry, Farshad Hakimpour, and Amit Sheth. Analyzing theme, space, and time : an ontology-based approach. In *Proceedings of the 14th annual ACM international symposium on Advances in geographic information systems*, GIS 06, pages 147–154. ACM, 2006.

[111] Simone Paolo Ponzetto and Roberto Navigli. Large-scale taxonomy mapping for restructuring and integrating wikipedia. In *Proceedings of the 21st International Jont Conference on Artifical Intelligence*, IJCAI'09, pages 2083–2088. Morgan Kaufmann Publishers Inc., 2009.

[112] Paul Ramsey, Sandro Santilli, Regina Obe, Mark Cave-Ayland, and Park Bborie. OSGeo project. In *Spatial and Geographic objects for PostgreSQL*, 2012. http://postgis.refractions.net.

[113] David A. Randell, Zhan Cui, and Anthony G. Cohn. A spatial logic based on regions and connection. In *Proceedings of the third International Conference on Principles of Knowledge Representation and Reasoning (KR'92)*, pages 165–176, 1992.

[114] Resource Description Framework (RDF) Concepts and Abstract Syntax. World wide web consortium, 2004. `http://www.w3.org/TR/2004/REC-rdf-concepts-20040210/`.

[115] Resource Description Framework (RDF) Schema Specification 1.0. World wide web consortium, 2004. `http://www.w3.org/TR/2000/CR-rdf-schema-20000327/`.

[116] Oracle® Spatial and Graph RDF Semantic Graph Developer's Guide 12C Release 1 (12.1). Oracle documentation, 2014. `http://docs.oracle.com/database/121/RDFRM`.

[117] Satya S. Sahoo, Wolfgang Halb, Sebastian Hellmann, Kingsley Idehen, Ted Thibodeau, and Soren Auer. A survey of current approaches for mapping of relational databases to RDF. W3C RDB2RDF incubator group, 2009.

[118] SEEK. EU marie curie project N295179, program people IRSES 2011 scheme - SEEK. Semantic enrichment of trajectory knowledge discovery, 2012-2015. `http://www.seek-project.eu`.

[119] Marianne Shaw, Landon T. Detwiler, James F. Brinkley, and Dan Suciu. A dataflow graph transformation language and query rewriting system for RDF ontologies. In *Proceedings of the 24th International Conference on Scientific and Statistical Database Management*, SSDBM'12, pages 544–561. Springer-Verlag, 2012.

[120] Pavel Shvaiko and Jérôme Euzenat. A survey of schema-based matching approaches. In *Journal on Data Semantics IV*, Lecture Notes in Computer Science, pages 146–171. Springer Berlin Heidelberg, 2005.

[121] Avi Silberschatz, Henry F. Korth, and S. Sudarshan. Data models. *ACM Comput. Surv.*, pages 105–108, 1996.

[122] Swapna Singh and Ragini Karwayun. A comparative study of inference engines. In *Proceedings of the 2010 Seventh International Conference on Information Technology : New Generations*, ITNG '10, pages 53–57. IEEE Computer Society, 2010.

[123] P. Sistla, O. Wolfson, S. Chamberlain, and S. Dao. Querying the uncertain position of moving objects. In *Temporal Databases : Research and Practice*, pages 310–337. Springer, 1998.

[124] Alexaki Sofia, Christophides Vassilis, Karvounarakis Gregory, Plexousakis Dimitris, and Tolle Karsten. The ICS-FORTH RDFSuite : Managing voluminous RDF description bases, 2001. `http://139.91.183.30:9090/RDF`.

[125] Das Souripriya and Srinivasan Jagannathan. Database technologies for RDF. In *Reasoning Web. Semantic Technologies for Information Systems, 5th International Summer School 2009*, Lecture Notes in Computer Science, pages 205–221. Springer, 2009.

[126] S Spaccapietra, C Parent, M Damiani, J Demacedo, F Porto, and C Vangenot. A conceptual view on trajectories. *Data and Knowledge Engineering*, pages 126–146, 2008.

[127] ISO/IEC 13249-3 :2002 FDIS. In *Information technology Database languages SQL Multimedia and Application Packages Part 3 : Spatial*, 2002.

[128] Markus Stocker and Evren Sirin. PelletSpatial : A hybrid RCC-8 and RDF/OWL reasoning and query engine. In *OWLED*, CEUR Workshop Proceedings. CEUR-WS.org, 2008.

[129] Stephens Susie M., Rung Johan, and Lopez Xavier. Graph data representation in Oracle database 10G : Case studies in life science. *IEEE data engineering bull*, pages 61–67, 2004.

[130] R. Gruber Thomas. Ontolingua : A mechanism to support portable ontologies. Technical report, Stanford University, Knowledge Systems Laboratory, 1992.

[131] Willem Robert van Hage, Veronique Malaise, Gerben de Vries, Guus Schreiber, and Maarten van Someren. Combining ship trajectories and semantics with the simple event model (SEM). In *Proceedings of the 1st ACM international workshop on Events in multimedia*, EiMM 09, pages 73–80. ACM, 2009.

[132] Arnaud Vandecasteele and Aldo Napoli. An enhanced spatial reasoning ontology for maritime anomaly detection. In *International Conference on System Of Systems Engineering - IEEE SOSE*, GIS 06, pages 247–252, 2012.

[133] Arnaud Vandecasteele. *Expert knowledge ontological modeling for the analyse of abnormal behaviour : application for maritime surveillance.* PhD thesis, Mines ParisTech university, 2012.

[134] Paola Villa and Elena Camossi. A description logic approach to discover suspicious itineraries from maritime container trajectories. In *GeoSpatial Semantics*, Lecture Notes in Computer Science, pages 182–199. Springer Berlin Heidelberg, 2011.

[135] Rouaa Wannous, Jamal Malki, Alain Bouju, and Cécile Vincent. Modelling mobile object activities based on trajectory ontology rules considering spatial relationship rules. In *Modeling Approaches and Algorithms for Advanced Computer Applications*, Studies in Computational Intelligence, pages 249–258. Springer International Publishing, 2013.

[136] Rouaa Wannous, Jamal Malki, Alain Bouju, and Cécile Vincent. Time integration in semantic trajectories using an ontological modelling approach. In *New Trends in Databases and Information Systems*, Advances in Intelligent Systems and Computing, pages 187–198. Springer Berlin Heidelberg, 2013.

[137] Michael Widenius and David Axmark. *MySQL Reference Manual.* Reilly Media, Incorporate, 2002.

[138] Ouri Wolfson, Bo Xu, Sam Chamberlain, and Liqin Jiang. Moving objects databases : Issues and solutions. In *Proceedings of the 10th International Conference on Scientific and Statistical Database Management*, SSDBM '98, pages 111–122. IEEE Computer Society, 1998.

[139] Zhe Wu, George Eadon, Souripriya Das, Eugene Inseok Chong, Vladimir Kolovski, Melliyal Annamalai, and Jagannathan Srinivasan. Implementing an inference engine for RDFS/OWL constructs and user-defined rules in Oracle. In *Proceedings of the 2008 IEEE 24th International Conference on Data Engineering*, ICDE 08, pages 1239–1248. IEEE Computer Society, 2008.

[140] Zhixian Yan, J Macedo, C Parent, and S Spaccapietra. Trajectory ontologies and queries. *Transactions in GIS*, pages 75–91, 2008.

[141] Zhixian Yan, C Parent, S Spaccapietra, and D Chakraborty. A hybrid model and computing platform for spatio-semantic trajectories. In *The Semantic Web : Research and Applications*, pages 60–75. Springer Berlin/Heidelberg, 2010.

[142] Zhixian Yan, D Chakraborty, C Parent, S Spaccapietra, and K Aberer. SeMiTri : A framework for semantic annotation of heterogeneous trajectories. In *Proceedings of the 14th International Conference on Extending Database Technology*, pages 259–270. ACM, 2011.

[143] Zhixian Yan, Dipanjan Chakraborty, Christine Parent, Stefano Spaccapietra, and Karl Aberer. Semantic trajectories : Mobility data computation and annotation. *ACM Transactions on Intelligent Systems and Technology*, pages 1–49, 2013.

[144] Zhixian Yan. Towards semantic trajectory data analysis : A conceptual and computational approach. In *VLDB PhD Workshop*, 2009.

[145] Hongyu Zhang, Yuan-Fang Li, and Hee Beng Kuan Tan. Measuring design complexity of semantic Web ontologies. *Journal of Systems and Software*, pages 803–814, 2010.

[146] Donia Zheni, Ali Frihida, HendaBen Ghezala, and Christophe Claramunt. A semantic approach for the modeling of trajectories in space and time. In *Advances in Conceptual Modeling - Challenging Perspectives*, Lecture Notes in Computer Science, pages 347–356. Springer Berlin Heidelberg, 2009.

[147] Yin Zhu, VincentWenchen Zheng, and Qiang Yang. Activity recognition from trajectory data. In *Computing with Spatial Trajectories*, pages 179–212. Springer New York, 2011.

Appendix rules

1 Temporal rules

1.1 IntervalAfter rule

```
DELETE FROM mdsys.semr_owlTime_rb
WHERE upper(rule_name) = upper('intervalAfter_rule');

INSERT INTO mdsys.semr_owlTime_rb VALUES(
  'intervalAfter_rule',
  '(?x rdf:type :ProperInterval) (?x :hasEnd ?xEnd)
   (?xEnd :inXSDDateTime ?xEndDateTime) (?y rdf:type :ProperInterval)
   (?y :hasBeginning ?yBegin) (?yBegin :inXSDDateTime ?yBeginDateTime)',
   '(yBeginDateTime > xEndDateTime)',
   '(?y :intervalAfter ?x)',
  SEM_ALIASES(SEM_ALIAS('','http://www.w3.org/2006/time#')));
```

Code 10.2 – Declarative part of `intervalAfter rule` (NT format)

1.2 IntervalBefore rule

```
DELETE FROM mdsys.semr_owlTime_rb
WHERE upper(rule_name) = upper('intervalBefore_rule');

INSERT INTO mdsys.semr_owlTime_rb VALUES(
  'intervalBefore_rule',
  '(?x rdf:type :ProperInterval)
   (?x :hasEnd ?xEnd)
   (?xEnd :inXSDDateTime ?xEndDateTime)
   (?x :hasBeginning ?xBegin)
   (?xBegin :inXSDDateTime ?xBeginDateTime)
   (?y rdf:type :ProperInterval)
   (?y :hasBeginning ?yBegin)
   (?yBegin :inXSDDateTime ?yBeginDateTime)
   (?y :hasEnd ?yEnd)
   (?yEnd :inXSDDateTime ?yEndDateTime)',
  '((timeIntervalLengthInSeconds(dateTime2TimeStamp(xEndDateTime),
                            dateTime2TimeStamp(yBeginDateTime))<0)
    and
   (timeIntervalLengthInSeconds(dateTime2TimeStamp(xBeginDateTime),
                            dateTime2TimeStamp(xEndDateTime))+
    timeIntervalLengthInSeconds(dateTime2TimeStamp(xEndDateTime),
                            dateTime2TimeStamp(yBeginDateTime))<0)
    and
   (timeIntervalLengthInSeconds(dateTime2TimeStamp(xEndDateTime),
                            dateTime2TimeStamp(yBeginDateTime))+
    timeIntervalLengthInSeconds(dateTime2TimeStamp(yBeginDateTime),
```

```
                            dateTime2TimeStamp(yEndDateTime))<0)
   and
   (timeIntervalLengthInSeconds(dateTime2TimeStamp(xBeginDateTime),
                            dateTime2TimeStamp(xEndDateTime))+
    timeIntervalLengthInSeconds(dateTime2TimeStamp(xEndDateTime),
                            dateTime2TimeStamp(yBeginDateTime))+
    timeIntervalLengthInSeconds(dateTime2TimeStamp(yBeginDateTime),
                            dateTime2TimeStamp(yEndDateTime))<0))',
  '(?y :intervalBefore ?x)',
 SEM_ALIASES(SEM_ALIAS('','http://www.w3.org/2006/time#')));
```

Code 10.3 – Declarative part of `intervalBefore rule` (NT format)

1.3 IntervalContains rule

```
DELETE FROM mdsys.semr_owlTime_rb
WHERE upper(rule_name) = upper('intervalContains_rule');

INSERT INTO mdsys.semr_owlTime_rb VALUES(
  'intervalContains_rule',
  '(?x rdf:type :ProperInterval)
   (?x :hasEnd ?xEnd)
   (?xEnd :inXSDDateTime ?xEndDateTime)
   (?x :hasBeginning ?xBegin)
   (?xBegin :inXSDDateTime ?xBeginDateTime)
   (?y rdf:type :ProperInterval)
   (?y :hasBeginning ?yBegin)
   (?yBegin :inXSDDateTime ?yBeginDateTime)
   (?y :hasEnd ?yEnd)
   (?yEnd :inXSDDateTime ?yEndDateTime)',
  '((timeIntervalLengthInSeconds(dateTime2TimeStamp(xEndDateTime),
                            dateTime2TimeStamp(yBeginDateTime))<0)
    and
   (timeIntervalLengthInSeconds(dateTime2TimeStamp(xBeginDateTime),
                            dateTime2TimeStamp(xEndDateTime))+
    timeIntervalLengthInSeconds(dateTime2TimeStamp(xEndDateTime),
                            dateTime2TimeStamp(yBeginDateTime))<0)
    and
   (timeIntervalLengthInSeconds(dateTime2TimeStamp(xEndDateTime),
                            dateTime2TimeStamp(yBeginDateTime))+
    timeIntervalLengthInSeconds(dateTime2TimeStamp(yBeginDateTime),
                            dateTime2TimeStamp(yEndDateTime))>0))',
  '(?y :intervalContains ?x)',
 SEM_ALIASES(SEM_ALIAS('','http://www.w3.org/2006/time#')));
```

Code 10.4 – Declarative part of `intervalContains rule` (NT format)

1.4 IntervalDuring rule

```
DELETE FROM mdsys.semr_owlTime_rb
WHERE upper(rule_name) = upper('intervalDuring_rule');

INSERT INTO mdsys.semr_owlTime_rb VALUES(
  'intervalDuring_rule',
  '(?x rdf:type :ProperInterval)
   (?x :hasEnd ?xEnd)
   (?xEnd :inXSDDateTime ?xEndDateTime)
   (?x :hasBeginning ?xBegin)
   (?xBegin :inXSDDateTime ?xBeginDateTime)
   (?y rdf:type :ProperInterval)
   (?y :hasBeginning ?yBegin)
```

```
  (?yBegin :inXSDDateTime ?yBeginDateTime)
  (?y :hasEnd ?yEnd)
  (?yEnd :inXSDDateTime ?yEndDateTime)',
 '((timeIntervalLengthInSeconds(dateTime2TimeStamp(xEndDateTime),
                            dateTime2TimeStamp(yBeginDateTime))<0)
   and
  (timeIntervalLengthInSeconds(dateTime2TimeStamp(xBeginDateTime),
                            dateTime2TimeStamp(xEndDateTime))+
   timeIntervalLengthInSeconds(dateTime2TimeStamp(xEndDateTime),
                            dateTime2TimeStamp(yBeginDateTime))>0)
   and
  (timeIntervalLengthInSeconds(dateTime2TimeStamp(xEndDateTime),
                            dateTime2TimeStamp(yBeginDateTime))+
   timeIntervalLengthInSeconds(dateTime2TimeStamp(yBeginDateTime),
                            dateTime2TimeStamp(yEndDateTime))<0))',
 '(?y :intervalDuring ?x)',
 SEM_ALIASES(SEM_ALIAS('','http://www.w3.org/2006/time#')));
```

Code 10.5 – Declarative part of `intervalDuring rule` (NT format)

1.5 IntervalEquals rule

```
DELETE FROM mdsys.semr_owlTime_rb
WHERE upper(rule_name) = upper('intervalEquals_rule');

INSERT INTO mdsys.semr_owlTime_rb VALUES(
  'intervalEquals_rule',
  '(?x rdf:type :ProperInterval)
   (?x :hasEnd ?xEnd)
   (?xEnd :inXSDDateTime ?xEndDateTime)
   (?x :hasBeginning ?xBegin)
   (?xBegin :inXSDDateTime ?xBeginDateTime)
   (?y rdf:type :ProperInterval)
   (?y :hasBeginning ?yBegin)
   (?yBegin :inXSDDateTime ?yBeginDateTime)
   (?y :hasEnd ?yEnd)
   (?yEnd :inXSDDateTime ?yEndDateTime)',
  '((timeIntervalLengthInSeconds(dateTime2TimeStamp(xEndDateTime),
                            dateTime2TimeStamp(yBeginDateTime))<0)
    and
   (timeIntervalLengthInSeconds(dateTime2TimeStamp(xBeginDateTime),
                            dateTime2TimeStamp(xEndDateTime))+
    timeIntervalLengthInSeconds(dateTime2TimeStamp(xEndDateTime),
                            dateTime2TimeStamp(yBeginDateTime))=0)
    and
   (timeIntervalLengthInSeconds(dateTime2TimeStamp(xEndDateTime),
                            dateTime2TimeStamp(yBeginDateTime))+
    timeIntervalLengthInSeconds(dateTime2TimeStamp(yBeginDateTime),
                            dateTime2TimeStamp(yEndDateTime))=0))',
  '(?y :intervalEquals ?x)',
  SEM_ALIASES(SEM_ALIAS('','http://www.w3.org/2006/time#')));
```

Code 10.6 – Declarative part of `intervalEquals rule` (NT format)

1.6 IntervalFinishedBy rule

```
DELETE FROM mdsys.semr_owlTime_rb
WHERE upper(rule_name) = upper('intervalFinishedBy_rule');

INSERT INTO mdsys.semr_owlTime_rb VALUES(
  'intervalFinishedBy_rule',
```

```
  '(?x rdf:type :ProperInterval)
   (?x :hasEnd ?xEnd)
   (?xEnd :inXSDDateTime ?xEndDateTime)
   (?x :hasBeginning ?xBegin)
   (?xBegin :inXSDDateTime ?xBeginDateTime)
   (?y rdf:type :ProperInterval)
   (?y :hasBeginning ?yBegin)
   (?yBegin :inXSDDateTime ?yBeginDateTime)
   (?y :hasEnd ?yEnd)
   (?yEnd :inXSDDateTime ?yEndDateTime)',
  '((timeIntervalLengthInSeconds(dateTime2TimeStamp(xEndDateTime),
                              dateTime2TimeStamp(yBeginDateTime))<0)
     and
    (timeIntervalLengthInSeconds(dateTime2TimeStamp(xBeginDateTime),
                              dateTime2TimeStamp(xEndDateTime))+
     timeIntervalLengthInSeconds(dateTime2TimeStamp(xEndDateTime),
                              dateTime2TimeStamp(yBeginDateTime))<0)
     and
    (timeIntervalLengthInSeconds(dateTime2TimeStamp(xEndDateTime),
                              dateTime2TimeStamp(yBeginDateTime))+
     timeIntervalLengthInSeconds(dateTime2TimeStamp(yBeginDateTime),
                              dateTime2TimeStamp(yEndDateTime))=0))',
   '(?y :intervalFinishedBy ?x)',
  SEM_ALIASES(SEM_ALIAS('','http://www.w3.org/2006/time#')));
```

Code 10.7 – Declarative part of `intervalFinished rule` (NT format)

1.7 IntervalFinishes rule

```
DELETE FROM mdsys.semr_owlTime_rb
WHERE upper(rule_name) = upper('intervalFinishes_rule');

INSERT INTO mdsys.semr_owlTime_rb VALUES(
  'intervalFinishes_rule',
  '(?x rdf:type :ProperInterval)
   (?x :hasEnd ?xEnd)
   (?xEnd :inXSDDateTime ?xEndDateTime)
   (?x :hasBeginning ?xBegin)
   (?xBegin :inXSDDateTime ?xBeginDateTime)
   (?y rdf:type :ProperInterval)
   (?y :hasBeginning ?yBegin)
   (?yBegin :inXSDDateTime ?yBeginDateTime)
   (?y :hasEnd ?yEnd)
   (?yEnd :inXSDDateTime ?yEndDateTime)',
  '((timeIntervalLengthInSeconds(dateTime2TimeStamp(xEndDateTime),
                              dateTime2TimeStamp(yBeginDateTime))<0)
     and
    (timeIntervalLengthInSeconds(dateTime2TimeStamp(xBeginDateTime),
                              dateTime2TimeStamp(xEndDateTime))+
     timeIntervalLengthInSeconds(dateTime2TimeStamp(xEndDateTime),
                              dateTime2TimeStamp(yBeginDateTime))>0)
     and
    (timeIntervalLengthInSeconds(dateTime2TimeStamp(xEndDateTime),
                              dateTime2TimeStamp(yBeginDateTime))+
     timeIntervalLengthInSeconds(dateTime2TimeStamp(yBeginDateTime),
                              dateTime2TimeStamp(yEndDateTime))=0))',
   '(?y :intervalFinishes ?x)',
  SEM_ALIASES(SEM_ALIAS('','http://www.w3.org/2006/time#')));
```

Code 10.8 – Declarative part of `intervalFinishes rule` (NT format)

1.8 IntervalMeets rule

```
DELETE FROM mdsys.semr_owlTime_rb
WHERE upper(rule_name) = upper('intervalMeets_rule');

INSERT INTO mdsys.semr_owlTime_rb VALUES(
  'intervalMeets_rule',
  '(?x rdf:type :ProperInterval)
   (?x :hasEnd ?xEnd)
   (?xEnd :inXSDDateTime ?xEndDateTime)
   (?x :hasBeginning ?xBegin)
   (?xBegin :inXSDDateTime ?xBeginDateTime)
   (?y rdf:type :ProperInterval)
   (?y :hasBeginning ?yBegin)
   (?yBegin :inXSDDateTime ?yBeginDateTime)
   (?y :hasEnd ?yEnd)
   (?yEnd :inXSDDateTime ?yEndDateTime)',
  '((timeIntervalLengthInSeconds(dateTime2TimeStamp(xEndDateTime),
                              dateTime2TimeStamp(yBeginDateTime))<0)
    and
   (timeIntervalLengthInSeconds(dateTime2TimeStamp(xBeginDateTime),
                              dateTime2TimeStamp(xEndDateTime))+
    timeIntervalLengthInSeconds(dateTime2TimeStamp(xEndDateTime),
                              dateTime2TimeStamp(yBeginDateTime))<0)
    and
   (timeIntervalLengthInSeconds(dateTime2TimeStamp(xEndDateTime),
                              dateTime2TimeStamp(yBeginDateTime))+
    timeIntervalLengthInSeconds(dateTime2TimeStamp(yBeginDateTime),
                              dateTime2TimeStamp(yEndDateTime))<0)
   and
   (timeIntervalLengthInSeconds(dateTime2TimeStamp(xBeginDateTime),
                              dateTime2TimeStamp(xEndDateTime))+
    timeIntervalLengthInSeconds(dateTime2TimeStamp(xEndDateTime),
                              dateTime2TimeStamp(yBeginDateTime))+
    timeIntervalLengthInSeconds(dateTime2TimeStamp(yBeginDateTime),
                              dateTime2TimeStamp(yEndDateTime))=0))',
  '(?y :intervalMeets ?x)',
  SEM_ALIASES(SEM_ALIAS('','http://www.w3.org/2006/time#')));
```

Code 10.9 – Declarative part of `intervalMeets rule` (NT format)

1.9 IntervalMetBy rule

```
DELETE FROM mdsys.semr_owlTime_rb
WHERE upper(rule_name) = upper('intervalMetBy_rule');

INSERT INTO mdsys.semr_owlTime_rb VALUES(
  'intervalMetBy_rule',
  '(?x rdf:type :ProperInterval) (?x :hasEnd ?xEnd)
   (?xEnd :inXSDDateTime ?xEndDateTime) (?y rdf:type :ProperInterval)
   (?y :hasBeginning ?yBegin) (?yBegin :inXSDDateTime ?yBeginDateTime)',
   '(yBeginDateTime = xEndDateTime)',
   '(?y :intervalMetBy ?x)',
  SEM_ALIASES(SEM_ALIAS('','http://www.w3.org/2006/time#')));
```

Code 10.10 – Declarative part of `intervalMetBy rule` (NT format)

1.10 IntervalOverlappedBy rule

```
DELETE FROM mdsys.semr_owlTime_rb
WHERE upper(rule_name) = upper('intervalOverlappedBy_rule');
```

```
INSERT INTO mdsys.semr_owlTime_rb VALUES(
  'intervalOverlappedBy_rule',
  '(?x rdf:type :ProperInterval)
   (?x :hasEnd ?xEnd)
   (?xEnd :inXSDDateTime ?xEndDateTime)
   (?x :hasBeginning ?xBegin)
   (?xBegin :inXSDDateTime ?xBeginDateTime)
   (?y rdf:type :ProperInterval)
   (?y :hasBeginning ?yBegin)
   (?yBegin :inXSDDateTime ?yBeginDateTime)
   (?y :hasEnd ?yEnd)
   (?yEnd :inXSDDateTime ?yEndDateTime)',
  '((timeIntervalLengthInSeconds(dateTime2TimeStamp(xEndDateTime),
                              dateTime2TimeStamp(yBeginDateTime))<0)
     and
    (timeIntervalLengthInSeconds(dateTime2TimeStamp(xBeginDateTime),
                              dateTime2TimeStamp(xEndDateTime))+
     timeIntervalLengthInSeconds(dateTime2TimeStamp(xEndDateTime),
                              dateTime2TimeStamp(yBeginDateTime))>0)
     and
    (timeIntervalLengthInSeconds(dateTime2TimeStamp(xEndDateTime),
                              dateTime2TimeStamp(yBeginDateTime))+
     timeIntervalLengthInSeconds(dateTime2TimeStamp(yBeginDateTime),
                              dateTime2TimeStamp(yEndDateTime))>0))',
   '(?y :intervalOverlappedBy ?x)',
  SEM_ALIASES(SEM_ALIAS('','http://www.w3.org/2006/time#')));
```

Code 10.11 – Declarative part of `intervalOverlappedBy rule` (NT format)

1.11 IntervalOverlaps rule

```
DELETE FROM mdsys.semr_owlTime_rb
WHERE upper(rule_name) = upper('intervalOverlaps_rule');

INSERT INTO mdsys.semr_owlTime_rb VALUES(
  'intervalOverlaps_rule',
  '(?x rdf:type :ProperInterval)
   (?x :hasEnd ?xEnd)
   (?xEnd :inXSDDateTime ?xEndDateTime)
   (?x :hasBeginning ?xBegin)
   (?xBegin :inXSDDateTime ?xBeginDateTime)
   (?y rdf:type :ProperInterval)
   (?y :hasBeginning ?yBegin)
   (?yBegin :inXSDDateTime ?yBeginDateTime)
   (?y :hasEnd ?yEnd)
   (?yEnd :inXSDDateTime ?yEndDateTime)',
  '((timeIntervalLengthInSeconds(dateTime2TimeStamp(xEndDateTime),
                              dateTime2TimeStamp(yBeginDateTime))<0)
     and
    (timeIntervalLengthInSeconds(dateTime2TimeStamp(xBeginDateTime),
                              dateTime2TimeStamp(xEndDateTime))+
     timeIntervalLengthInSeconds(dateTime2TimeStamp(xEndDateTime),
                              dateTime2TimeStamp(yBeginDateTime))<0)
     and
    (timeIntervalLengthInSeconds(dateTime2TimeStamp(xEndDateTime),
                              dateTime2TimeStamp(yBeginDateTime))+
     timeIntervalLengthInSeconds(dateTime2TimeStamp(yBeginDateTime),
                              dateTime2TimeStamp(yEndDateTime))<0)
    and
    (timeIntervalLengthInSeconds(dateTime2TimeStamp(xBeginDateTime),
                              dateTime2TimeStamp(xEndDateTime))+
     timeIntervalLengthInSeconds(dateTime2TimeStamp(xEndDateTime),
                              dateTime2TimeStamp(yBeginDateTime))+
     timeIntervalLengthInSeconds(dateTime2TimeStamp(yBeginDateTime),
```

```
                              dateTime2TimeStamp(yEndDateTime))>0))',
  '(?y :intervalOverlaps ?x)',
 SEM_ALIASES(SEM_ALIAS('','http://www.w3.org/2006/time#')));
```

Code 10.12 – Declarative part of `intervalOverlaps rule` (NT format)

1.12 IntervalStartedBy rule

```
DELETE FROM mdsys.semr_owlTime_rb
WHERE upper(rule_name) = upper('intervalStartedBy_rule');

INSERT INTO mdsys.semr_owlTime_rb VALUES(
  'intervalStartedBy_rule',
  '(?x rdf:type :ProperInterval)
   (?x :hasEnd ?xEnd)
   (?xEnd :inXSDDateTime ?xEndDateTime)
   (?x :hasBeginning ?xBegin)
   (?xBegin :inXSDDateTime ?xBeginDateTime)
   (?y rdf:type :ProperInterval)
   (?y :hasBeginning ?yBegin)
   (?yBegin :inXSDDateTime ?yBeginDateTime)
   (?y :hasEnd ?yEnd)
   (?yEnd :inXSDDateTime ?yEndDateTime)',
  '((timeIntervalLengthInSeconds(dateTime2TimeStamp(xEndDateTime),
                              dateTime2TimeStamp(yBeginDateTime))<0)
    and
    (timeIntervalLengthInSeconds(dateTime2TimeStamp(xBeginDateTime),
                              dateTime2TimeStamp(xEndDateTime))+
    timeIntervalLengthInSeconds(dateTime2TimeStamp(xEndDateTime),
                              dateTime2TimeStamp(yBeginDateTime))=0)
    and
   (timeIntervalLengthInSeconds(dateTime2TimeStamp(xEndDateTime),
                              dateTime2TimeStamp(yBeginDateTime))+
    timeIntervalLengthInSeconds(dateTime2TimeStamp(yBeginDateTime),
                              dateTime2TimeStamp(yEndDateTime))>0))',
  '(?y :intervalStartedBy ?x)',
 SEM_ALIASES(SEM_ALIAS('','http://www.w3.org/2006/time#')));
```

Code 10.13 – Declarative part of `intervalStartedBy rule` (NT format)

1.13 IntervalStarts rule

```
DELETE FROM mdsys.semr_owlTime_rb
WHERE upper(rule_name) = upper('intervalStarts_rule');

INSERT INTO mdsys.semr_owlTime_rb VALUES(
  'intervalStarts_rule',
  '(?x rdf:type :ProperInterval)
   (?x :hasEnd ?xEnd)
   (?xEnd :inXSDDateTime ?xEndDateTime)
   (?x :hasBeginning ?xBegin)
   (?xBegin :inXSDDateTime ?xBeginDateTime)
   (?y rdf:type :ProperInterval)
   (?y :hasBeginning ?yBegin)
   (?yBegin :inXSDDateTime ?yBeginDateTime)
   (?y :hasEnd ?yEnd)
   (?yEnd :inXSDDateTime ?yEndDateTime)',
  '((timeIntervalLengthInSeconds(dateTime2TimeStamp(xEndDateTime),
                              dateTime2TimeStamp(yBeginDateTime))<0)
    and
   (timeIntervalLengthInSeconds(dateTime2TimeStamp(xBeginDateTime),
```

```
                            dateTime2TimeStamp(xEndDateTime))+
    timeIntervalLengthInSeconds(dateTime2TimeStamp(xEndDateTime),
                            dateTime2TimeStamp(yBeginDateTime))=0)
    and
   (timeIntervalLengthInSeconds(dateTime2TimeStamp(xEndDateTime),
                            dateTime2TimeStamp(yBeginDateTime))+
    timeIntervalLengthInSeconds(dateTime2TimeStamp(yBeginDateTime),
                            dateTime2TimeStamp(yEndDateTime))<0))',
  '(?y :intervalStarts ?x)',
 SEM_ALIASES(SEM_ALIAS('','http://www.w3.org/2006/time#')));
```

Code 10.14 – Declarative part of `intervalStarts rule` (NT format)

2 Refinement temporal rules

2.1 intervalAfter refinement rule

```
DELETE FROM mdsys.semr_owlTime_rb
WHERE upper(rule_name) = upper('intervalAfterRef_rule');

INSERT INTO mdsys.semr_owlTime_rb VALUES(
  'intervalAfterRef_rule',
  '(?x rdf:type ot:ProperInterval) (?x ot:hasEnd ?xEnd)
   (?xEnd ot:inXSDDateTime ?xEndDateTime) (?y rdf:type ot:ProperInterval)
   (?y ot:hasBeginning ?yBegin) (?yBegin ot:inXSDDateTime ?yBeginDateTime)',
   '((timeIntervalLengthInSeconds(dateTime2TimeStamp(xEndDateTime),
                            dateTime2TimeStamp(yBeginDateTime))<300)
       and
       (yBeginDateTime > xEndDateTime))',
   '(?y ot:intervalAfterRef ?x)',
  SEM_ALIASES(SEM_ALIAS('ot','http://www.w3.org/2006/time#')));
```

Code 10.15 – Declarative part of `intervalAfter refinement` rule (NT format)

2.2 intervalBefore refinement rule

```
DELETE FROM mdsys.semr_owlTime_rb
WHERE upper(rule_name) = upper('intervalBeforeRef_rule');

INSERT INTO mdsys.semr_owlTime_rb VALUES(
  'intervalBeforeRef_rule',
  '(?x rdf:type :ProperInterval)
   (?x :hasEnd ?xEnd)
   (?xEnd :inXSDDateTime ?xEndDateTime)
   (?x :hasBeginning ?xBegin)
   (?xBegin :inXSDDateTime ?xBeginDateTime)
   (?y rdf:type :ProperInterval)
   (?y :hasBeginning ?yBegin)
   (?yBegin :inXSDDateTime ?yBeginDateTime)
   (?y :hasEnd ?yEnd)
   (?yEnd :inXSDDateTime ?yEndDateTime)',
  '((timeIntervalLengthInSeconds(dateTime2TimeStamp(xEndDateTime),
                            dateTime2TimeStamp(yBeginDateTime))<0)
    and
   (timeIntervalLengthInSeconds(dateTime2TimeStamp(xBeginDateTime),
                            dateTime2TimeStamp(xEndDateTime))+
    timeIntervalLengthInSeconds(dateTime2TimeStamp(xEndDateTime),
                            dateTime2TimeStamp(yBeginDateTime))<0)
    and
```

```
   (timeIntervalLengthInSeconds(dateTime2TimeStamp(xEndDateTime),
                            dateTime2TimeStamp(yBeginDateTime))+
    timeIntervalLengthInSeconds(dateTime2TimeStamp(yBeginDateTime),
                            dateTime2TimeStamp(yEndDateTime))<0)
   and
   (timeIntervalLengthInSeconds(dateTime2TimeStamp(xBeginDateTime),
                            dateTime2TimeStamp(xEndDateTime))+
    timeIntervalLengthInSeconds(dateTime2TimeStamp(xEndDateTime),
                            dateTime2TimeStamp(yBeginDateTime))+
    timeIntervalLengthInSeconds(dateTime2TimeStamp(yBeginDateTime),
                            dateTime2TimeStamp(yEndDateTime))<0)
      and
      (timeIntervalLengthInSeconds(dateTime2TimeStamp(yEndDateTime),
                            dateTime2TimeStamp(xBeginDateTime))<300))',
  '(?y :intervalBeforeRef ?x)',
 SEM_ALIASES(SEM_ALIAS('','http://www.w3.org/2006/time#')));
```

Code 10.16 – Declarative part of `intervalBefore refinement` rule (NT format)

2.3 intervalContains refinement rule

```
DELETE FROM mdsys.semr_owlTime_rb
WHERE upper(rule_name) = upper('intervalContainsRef_rule');

INSERT INTO mdsys.semr_owlTime_rb VALUES(
  'intervalContainsRef_rule',
  '(?x rdf:type :ProperInterval)
   (?x :hasEnd ?xEnd)
   (?xEnd :inXSDDateTime ?xEndDateTime)
   (?x :hasBeginning ?xBegin)
   (?xBegin :inXSDDateTime ?xBeginDateTime)
   (?y rdf:type :ProperInterval)
   (?y :hasBeginning ?yBegin)
   (?yBegin :inXSDDateTime ?yBeginDateTime)
   (?y :hasEnd ?yEnd)
   (?yEnd :inXSDDateTime ?yEndDateTime)',
  '((timeIntervalLengthInSeconds(dateTime2TimeStamp(xEndDateTime),
                            dateTime2TimeStamp(yBeginDateTime))<0)
    and
    (timeIntervalLengthInSeconds(dateTime2TimeStamp(xBeginDateTime),
                            dateTime2TimeStamp(xEndDateTime))+
    timeIntervalLengthInSeconds(dateTime2TimeStamp(xEndDateTime),
                            dateTime2TimeStamp(yBeginDateTime))<0)
    and
    (timeIntervalLengthInSeconds(dateTime2TimeStamp(xEndDateTime),
                            dateTime2TimeStamp(yBeginDateTime))+
    timeIntervalLengthInSeconds(dateTime2TimeStamp(yBeginDateTime),
                            dateTime2TimeStamp(yEndDateTime))>0)
        and
        ((timeIntervalLengthInSeconds(dateTime2TimeStamp(xBeginDateTime),
                            dateTime2TimeStamp(yBeginDateTime))<300)
              or
              (timeIntervalLengthInSeconds(dateTime2TimeStamp(xEndDateTime),
                            dateTime2TimeStamp(yEndDateTime))<300)))',
  '(?y :intervalContainsRef ?x)',
 SEM_ALIASES(SEM_ALIAS('','http://www.w3.org/2006/time#')));
```

Code 10.17 – Declarative part of `intervalContains refinement` rule (NT format)

2.4 intervalDuring refinement rule

```
DELETE FROM mdsys.semr_owlTime_rb
WHERE upper(rule_name) = upper('intervalDuringRef_rule');

INSERT INTO mdsys.semr_owlTime_rb VALUES(
  'intervalDuringRef_rule',
  '(?x rdf:type :ProperInterval)
   (?x :hasEnd ?xEnd)
   (?xEnd :inXSDDateTime ?xEndDateTime)
   (?x :hasBeginning ?xBegin)
   (?xBegin :inXSDDateTime ?xBeginDateTime)
   (?y rdf:type :ProperInterval)
   (?y :hasBeginning ?yBegin)
   (?yBegin :inXSDDateTime ?yBeginDateTime)
   (?y :hasEnd ?yEnd)
   (?yEnd :inXSDDateTime ?yEndDateTime)',
  '((timeIntervalLengthInSeconds(dateTime2TimeStamp(xEndDateTime),
                              dateTime2TimeStamp(yBeginDateTime))<0)
    and
   (timeIntervalLengthInSeconds(dateTime2TimeStamp(xBeginDateTime),
                              dateTime2TimeStamp(xEndDateTime))+
    timeIntervalLengthInSeconds(dateTime2TimeStamp(xEndDateTime),
                              dateTime2TimeStamp(yBeginDateTime))>0)
    and
   (timeIntervalLengthInSeconds(dateTime2TimeStamp(xEndDateTime),
                              dateTime2TimeStamp(yBeginDateTime))+
    timeIntervalLengthInSeconds(dateTime2TimeStamp(yBeginDateTime),
                              dateTime2TimeStamp(yEndDateTime))<0)
        and
   ((timeIntervalLengthInSeconds(dateTime2TimeStamp(yEndDateTime),
                              dateTime2TimeStamp(xEndDateTime)) < 300)
             or
    (timeIntervalLengthInSeconds(dateTime2TimeStamp(yBeginDateTime),
                              dateTime2TimeStamp(xBeginDateTime))<300)))',
  '(?y :intervalDuringRef ?x)',
  SEM_ALIASES(SEM_ALIAS('','http://www.w3.org/2006/time#')));
```

Code 10.18 – Declarative part of `intervalDuring refinement` rule (NT format)

2.5 intervalEquals refinement rule

```
DELETE FROM mdsys.semr_owlTime_rb
WHERE upper(rule_name) = upper('intervalEqualsRef_rule');

INSERT INTO mdsys.semr_owlTime_rb VALUES(
  'intervalEqualsRef_rule',
  '(?x rdf:type :ProperInterval)
   (?x :hasEnd ?xEnd)
   (?xEnd :inXSDDateTime ?xEndDateTime)
   (?x :hasBeginning ?xBegin)
   (?xBegin :inXSDDateTime ?xBeginDateTime)
   (?y rdf:type :ProperInterval)
   (?y :hasBeginning ?yBegin)
   (?yBegin :inXSDDateTime ?yBeginDateTime)
   (?y :hasEnd ?yEnd)
   (?yEnd :inXSDDateTime ?yEndDateTime)',
  '((timeIntervalLengthInSeconds(dateTime2TimeStamp(xEndDateTime),
                              dateTime2TimeStamp(yBeginDateTime))<0)
    and
   (timeIntervalLengthInSeconds(dateTime2TimeStamp(xBeginDateTime),
                              dateTime2TimeStamp(xEndDateTime))+
    timeIntervalLengthInSeconds(dateTime2TimeStamp(xEndDateTime),
                              dateTime2TimeStamp(yBeginDateTime))=0)
    and
   (timeIntervalLengthInSeconds(dateTime2TimeStamp(xEndDateTime),
```

```
                              dateTime2TimeStamp(yBeginDateTime))+
    timeIntervalLengthInSeconds(dateTime2TimeStamp(yBeginDateTime),
                              dateTime2TimeStamp(yEndDateTime))=0))',
  '(?y :intervalEqualsTime ?x)',
 SEM_ALIASES(SEM_ALIAS('','http://www.w3.org/2006/time#')));
```

Code 10.19 – Declarative part of `intervalEquals refinement` rule (NT format)

2.6 intervalFinishedBy refinement rule

```
DELETE FROM mdsys.semr_owlTime_rb
WHERE upper(rule_name) = upper('intervalFinishedByRef_rule');

INSERT INTO mdsys.semr_owlTime_rb VALUES(
  'intervalFinishedByRef_rule',
  '(?x rdf:type :ProperInterval)
   (?x :hasEnd ?xEnd)
   (?xEnd :inXSDDateTime ?xEndDateTime)
   (?x :hasBeginning ?xBegin)
   (?xBegin :inXSDDateTime ?xBeginDateTime)
   (?y rdf:type :ProperInterval)
   (?y :hasBeginning ?yBegin)
   (?yBegin :inXSDDateTime ?yBeginDateTime)
   (?y :hasEnd ?yEnd)
   (?yEnd :inXSDDateTime ?yEndDateTime)',
  '((timeIntervalLengthInSeconds(dateTime2TimeStamp(xEndDateTime),
                              dateTime2TimeStamp(yBeginDateTime))<0)
    and
   (timeIntervalLengthInSeconds(dateTime2TimeStamp(xBeginDateTime),
                              dateTime2TimeStamp(xEndDateTime))+
    timeIntervalLengthInSeconds(dateTime2TimeStamp(xEndDateTime),
                              dateTime2TimeStamp(yBeginDateTime))<0)
    and
   (timeIntervalLengthInSeconds(dateTime2TimeStamp(xEndDateTime),
                              dateTime2TimeStamp(yBeginDateTime))+
    timeIntervalLengthInSeconds(dateTime2TimeStamp(yBeginDateTime),
                              dateTime2TimeStamp(yEndDateTime))=0))',
  '(?y :intervalFinishedByRef ?x)',
 SEM_ALIASES(SEM_ALIAS('','http://www.w3.org/2006/time#')));
```

Code 10.20 – Declarative part of `intervalFinishedBy refinement` rule (NT format)

2.7 intervalFinishes refinement rule

```
DELETE FROM mdsys.semr_owlTime_rb
WHERE upper(rule_name) = upper('intervalFinishesRef_rule');

INSERT INTO mdsys.semr_owlTime_rb VALUES(
  'intervalFinishesRef_rule',
  '(?x rdf:type :ProperInterval)
   (?x :hasEnd ?xEnd)
   (?xEnd :inXSDDateTime ?xEndDateTime)
   (?x :hasBeginning ?xBegin)
   (?xBegin :inXSDDateTime ?xBeginDateTime)
   (?y rdf:type :ProperInterval)
   (?y :hasBeginning ?yBegin)
   (?yBegin :inXSDDateTime ?yBeginDateTime)
   (?y :hasEnd ?yEnd)
   (?yEnd :inXSDDateTime ?yEndDateTime)',
  '((timeIntervalLengthInSeconds(dateTime2TimeStamp(xEndDateTime),
                              dateTime2TimeStamp(yBeginDateTime))<0)
```

```
    and
   (timeIntervalLengthInSeconds(dateTime2TimeStamp(xBeginDateTime),
                              dateTime2TimeStamp(xEndDateTime))+
    timeIntervalLengthInSeconds(dateTime2TimeStamp(xEndDateTime),
                              dateTime2TimeStamp(yBeginDateTime))>0)
    and
   (timeIntervalLengthInSeconds(dateTime2TimeStamp(xEndDateTime),
                              dateTime2TimeStamp(yBeginDateTime))+
    timeIntervalLengthInSeconds(dateTime2TimeStamp(yBeginDateTime),
                              dateTime2TimeStamp(yEndDateTime))=0))',
  '(?y :intervalFinishesRef ?x)',
 SEM_ALIASES(SEM_ALIAS('','http://www.w3.org/2006/time#')));
```

Code 10.21 – Declarative part of `intervalFinishes refinement` rule (NT format)

2.8 intervalMeets refinement rule

```
DELETE FROM mdsys.semr_owlTime_rb
WHERE upper(rule_name) = upper('intervalMeetsRef_rule');

INSERT INTO mdsys.semr_owlTime_rb VALUES(
  'intervalMeetsRef_rule',
  '(?x rdf:type :ProperInterval)
   (?x :hasEnd ?xEnd)
   (?xEnd :inXSDDateTime ?xEndDateTime)
   (?x :hasBeginning ?xBegin)
   (?xBegin :inXSDDateTime ?xBeginDateTime)
   (?y rdf:type :ProperInterval)
   (?y :hasBeginning ?yBegin)
   (?yBegin :inXSDDateTime ?yBeginDateTime)
   (?y :hasEnd ?yEnd)
   (?yEnd :inXSDDateTime ?yEndDateTime)',
  '((timeIntervalLengthInSeconds(dateTime2TimeStamp(xEndDateTime),
                              dateTime2TimeStamp(yBeginDateTime))<0)
    and
   (timeIntervalLengthInSeconds(dateTime2TimeStamp(xBeginDateTime),
                              dateTime2TimeStamp(xEndDateTime))+
    timeIntervalLengthInSeconds(dateTime2TimeStamp(xEndDateTime),
                              dateTime2TimeStamp(yBeginDateTime))<0)
    and
   (timeIntervalLengthInSeconds(dateTime2TimeStamp(xEndDateTime),
                              dateTime2TimeStamp(yBeginDateTime))+
    timeIntervalLengthInSeconds(dateTime2TimeStamp(yBeginDateTime),
                              dateTime2TimeStamp(yEndDateTime))<0)
   and
   (timeIntervalLengthInSeconds(dateTime2TimeStamp(xBeginDateTime),
                              dateTime2TimeStamp(xEndDateTime))+
    timeIntervalLengthInSeconds(dateTime2TimeStamp(xEndDateTime),
                              dateTime2TimeStamp(yBeginDateTime))+
    timeIntervalLengthInSeconds(dateTime2TimeStamp(yBeginDateTime),
                              dateTime2TimeStamp(yEndDateTime))=0))',
  '(?y :intervalMeetsRef ?x)',
 SEM_ALIASES(SEM_ALIAS('','http://www.w3.org/2006/time#')));
```

Code 10.22 – Declarative part of `intervalMeets refinement` rule (NT format)

2.9 intervalMetBy refinement rule

```
DELETE FROM mdsys.semr_owlTime_rb
WHERE upper(rule_name) = upper('intervalMetByRef_rule');
```

```
INSERT INTO mdsys.semr_owlTime_rb VALUES(
  'intervalMetByRef_rule',
  '(?x rdf:type :ProperInterval) (?x :hasEnd ?xEnd)
   (?xEnd :inXSDDateTime ?xEndDateTime) (?y rdf:type :ProperInterval)
   (?y :hasBeginning ?yBegin) (?yBegin :inXSDDateTime ?yBeginDateTime)',
   '(yBeginDateTime = xEndDateTime)',
   '(?y :intervalMetByRef ?x)',
  SEM_ALIASES(SEM_ALIAS('','http://www.w3.org/2006/time#')));
```

Code 10.23 – Declarative part of `intervalMetBy refinement` rule (NT format)

2.10 intervalOverlappedBy refinement rule

```
DELETE FROM mdsys.semr_owlTime_rb
WHERE upper(rule_name) = upper('intervalOverlappedByRef_rule');

INSERT INTO mdsys.semr_owlTime_rb VALUES(
  'intervalOverlappedByRef_rule',
  '(?x rdf:type :ProperInterval)
   (?x :hasEnd ?xEnd)
   (?xEnd :inXSDDateTime ?xEndDateTime)
   (?x :hasBeginning ?xBegin)
   (?xBegin :inXSDDateTime ?xBeginDateTime)
   (?y rdf:type :ProperInterval)
   (?y :hasBeginning ?yBegin)
   (?yBegin :inXSDDateTime ?yBeginDateTime)
   (?y :hasEnd ?yEnd)
   (?yEnd :inXSDDateTime ?yEndDateTime)',
  '((timeIntervalLengthInSeconds(dateTime2TimeStamp(xEndDateTime),
                              dateTime2TimeStamp(yBeginDateTime))<0)
    and
   (timeIntervalLengthInSeconds(dateTime2TimeStamp(xBeginDateTime),
                              dateTime2TimeStamp(xEndDateTime))+
    timeIntervalLengthInSeconds(dateTime2TimeStamp(xEndDateTime),
                              dateTime2TimeStamp(yBeginDateTime))>0)
    and
   (timeIntervalLengthInSeconds(dateTime2TimeStamp(xEndDateTime),
                              dateTime2TimeStamp(yBeginDateTime))+
    timeIntervalLengthInSeconds(dateTime2TimeStamp(yBeginDateTime),
                              dateTime2TimeStamp(yEndDateTime))>0)
       and
       (timeIntervalLengthInSeconds(dateTime2TimeStamp(xEndDateTime),
                              dateTime2TimeStamp(yEndDateTime))< 300))',
   '(?y :intervalOverlappedByRef ?x)',
  SEM_ALIASES(SEM_ALIAS('','http://www.w3.org/2006/time#')));
```

Code 10.24 – Declarative part of `intervalOverlappedBy refinement` rule (NT format)

2.11 intervalOverlaps refinement rule

```
DELETE FROM mdsys.semr_owlTime_rb
WHERE upper(rule_name) = upper('intervalOverlapsRef_rule');

INSERT INTO mdsys.semr_owlTime_rb VALUES(
  'intervalOverlapsRef_rule',
  '(?x rdf:type :ProperInterval)
   (?x :hasEnd ?xEnd)
   (?xEnd :inXSDDateTime ?xEndDateTime)
   (?x :hasBeginning ?xBegin)
   (?xBegin :inXSDDateTime ?xBeginDateTime)
   (?y rdf:type :ProperInterval)
```

```
  (?y :hasBeginning ?yBegin)
  (?yBegin :inXSDDateTime ?yBeginDateTime)
  (?y :hasEnd ?yEnd)
  (?yEnd :inXSDDateTime ?yEndDateTime)',
 '((timeIntervalLengthInSeconds(dateTime2TimeStamp(xEndDateTime),
                            dateTime2TimeStamp(yBeginDateTime))<0)
   and
  (timeIntervalLengthInSeconds(dateTime2TimeStamp(xBeginDateTime),
                            dateTime2TimeStamp(xEndDateTime))+
   timeIntervalLengthInSeconds(dateTime2TimeStamp(xEndDateTime),
                            dateTime2TimeStamp(yBeginDateTime))<0)
   and
  (timeIntervalLengthInSeconds(dateTime2TimeStamp(xEndDateTime),
                            dateTime2TimeStamp(yBeginDateTime))+
   timeIntervalLengthInSeconds(dateTime2TimeStamp(yBeginDateTime),
                            dateTime2TimeStamp(yEndDateTime))<0)
  and
  (timeIntervalLengthInSeconds(dateTime2TimeStamp(xBeginDateTime),
                            dateTime2TimeStamp(xEndDateTime))+
   timeIntervalLengthInSeconds(dateTime2TimeStamp(xEndDateTime),
                            dateTime2TimeStamp(yBeginDateTime))+
   timeIntervalLengthInSeconds(dateTime2TimeStamp(yBeginDateTime),
                            dateTime2TimeStamp(yEndDateTime))>0)
      and
      (timeIntervalLengthInSeconds(dateTime2TimeStamp(xBeginDateTime),
                            dateTime2TimeStamp(yEndDateTime))< 300))',
 '(?y :intervalOverlapsRef ?x)',
SEM_ALIASES(SEM_ALIAS('','http://www.w3.org/2006/time#')));
```

Code 10.25 – Declarative part of `intervalOverlaps refinement` rule (NT format)

2.12 intervalStartedBy refinement rule

```
DELETE FROM mdsys.semr_owlTime_rb
WHERE upper(rule_name) = upper('intervalStartedByRef_rule');

INSERT INTO mdsys.semr_owlTime_rb VALUES(
  'intervalStartedByRef_rule',
  '(?x rdf:type :ProperInterval)
   (?x :hasEnd ?xEnd)
   (?xEnd :inXSDDateTime ?xEndDateTime)
   (?x :hasBeginning ?xBegin)
   (?xBegin :inXSDDateTime ?xBeginDateTime)
   (?y rdf:type :ProperInterval)
   (?y :hasBeginning ?yBegin)
   (?yBegin :inXSDDateTime ?yBeginDateTime)
   (?y :hasEnd ?yEnd)
   (?yEnd :inXSDDateTime ?yEndDateTime)',
  '((timeIntervalLengthInSeconds(dateTime2TimeStamp(xEndDateTime),
                            dateTime2TimeStamp(yBeginDateTime))<0)
    and
   (timeIntervalLengthInSeconds(dateTime2TimeStamp(xBeginDateTime),
                            dateTime2TimeStamp(xEndDateTime))+
    timeIntervalLengthInSeconds(dateTime2TimeStamp(xEndDateTime),
                            dateTime2TimeStamp(yBeginDateTime))=0)
    and
   (timeIntervalLengthInSeconds(dateTime2TimeStamp(xEndDateTime),
                            dateTime2TimeStamp(yBeginDateTime))+
    timeIntervalLengthInSeconds(dateTime2TimeStamp(yBeginDateTime),
                            dateTime2TimeStamp(yEndDateTime))>0))',
  '(?y :intervalStartedByRef ?x)',
 SEM_ALIASES(SEM_ALIAS('','http://www.w3.org/2006/time#')));
```

Code 10.26 – Declarative part of `intervalStartedBy refinement` rule (NT format)

2.13 intervalStarts refinement rule

```
DELETE FROM mdsys.semr_owlTime_rb
WHERE upper(rule_name) = upper('intervalStartsRef_rule');

INSERT INTO mdsys.semr_owlTime_rb VALUES(
  'intervalStartsRef_rule',
  '(?x rdf:type :ProperInterval)
   (?x :hasEnd ?xEnd)
   (?xEnd :inXSDDateTime ?xEndDateTime)
   (?x :hasBeginning ?xBegin)
   (?xBegin :inXSDDateTime ?xBeginDateTime)
   (?y rdf:type :ProperInterval)
   (?y :hasBeginning ?yBegin)
   (?yBegin :inXSDDateTime ?yBeginDateTime)
   (?y :hasEnd ?yEnd)
   (?yEnd :inXSDDateTime ?yEndDateTime)',
  '((timeIntervalLengthInSeconds(dateTime2TimeStamp(xEndDateTime),
                                 dateTime2TimeStamp(yBeginDateTime))<0)
     and
    (timeIntervalLengthInSeconds(dateTime2TimeStamp(xBeginDateTime),
                                 dateTime2TimeStamp(xEndDateTime))+
     timeIntervalLengthInSeconds(dateTime2TimeStamp(xEndDateTime),
                                 dateTime2TimeStamp(yBeginDateTime))=0)
     and
    (timeIntervalLengthInSeconds(dateTime2TimeStamp(xEndDateTime),
                                 dateTime2TimeStamp(yBeginDateTime))+
     timeIntervalLengthInSeconds(dateTime2TimeStamp(yBeginDateTime),
                                 dateTime2TimeStamp(yEndDateTime))<0))',
   '(?y :intervalStartsRef ?x)',
  SEM_ALIASES(SEM_ALIAS('','http://www.w3.org/2006/time#')));
```

Code 10.27 – Declarative part of `intervalStarts refinement` rule (NT format)

3 Spatial rules

3.1 AnyInteract rule

```
DELETE FROM mdsys.semr_owlSpatialOnto_rb
WHERE upper(rule_name) = upper('anyInteract_rule');

INSERT INTO mdsys.semr_owlSpatialOnto_rb VALUES(
 'anyInteract_rule',
 '(?spObj1 rdf:type os:Geometry)(?spObj1 os:srid ?sridSpObj1)(?spObj1 os:wkt ?strSpObj1)
       (?spObj2 rdf:type os:Geometry)(?spObj2 os:srid ?sridSpObj2)(?spObj2 os:wkt ?strSpObj2)',
  '(evalSpatialRelation(spObj1,strSpObj1,spObj2,strSpObj2,sridSpObj2,''ANYINTERACT'') = 1)',
  '(?spObj1 os:anyInteract ?spObj2)',
  SEM_ALIASES(SEM_ALIAS('os','http://l3i.univ-larochelle.fr/Sido/owlOGCSpatial#')));
```

Code 10.28 – Declarative part of `anyInteract` rule (NT format)

3.2 Contains rule

```
DELETE FROM mdsys.semr_owlSpatialOnto_rb
WHERE upper(rule_name) = upper('contains_rule');

INSERT INTO mdsys.semr_owlSpatialOnto_rb VALUES(
  'contains_rule',
```

```
  '(?spObj1 rdf:type os:Geometry)(?spObj1 os:srid ?sridSpObj1)(?spObj1 os:wkt ?strSpObj1)
       (?spObj2 rdf:type os:Geometry)(?spObj2 os:srid ?sridSpObj2)(?spObj2 os:wkt ?strSpObj2)',
  '(evalSpatialRelation(spObj1,strSpObj1,spObj2,strSpObj2,sridSpObj2,''CONTAINS'') = 1)',
  '(?spObj1 os:contains ?spObj2)',
  SEM_ALIASES(SEM_ALIAS('os','http://l3i.univ-larochelle.fr/Sido/owlOGCSpatial#')));
```

Code 10.29 – Declarative part of `contains` rule (NT format)

3.3 CoveredBy rule

```
DELETE FROM mdsys.semr_owlSpatialOnto_rb
WHERE upper(rule_name) = upper('coveredBy_rule');

INSERT INTO mdsys.semr_owlSpatialOnto_rb VALUES(
  'coveredBy_rule',
  '(?spObj1 rdf:type os:Geometry)(?spObj1 os:srid ?sridSpObj1)(?spObj1 os:wkt ?strSpObj1)
       (?spObj2 rdf:type os:Geometry)(?spObj2 os:srid ?sridSpObj2)(?spObj2 os:wkt ?strSpObj2)',
  '(evalSpatialRelation(spObj1,strSpObj1,spObj2,strSpObj2,sridSpObj2,''COVEREDBY'') = 1)',
  '(?spObj1 os:coveredBy ?spObj2)',
  SEM_ALIASES(SEM_ALIAS('os','http://l3i.univ-larochelle.fr/Sido/owlOGCSpatial#')));
```

Code 10.30 – Declarative part of `coveredBy` rule (NT format)

3.4 Covers rule

```
DELETE FROM mdsys.semr_owlSpatialOnto_rb
WHERE upper(rule_name) = upper('covers_rule');

INSERT INTO mdsys.semr_owlSpatialOnto_rb VALUES(
  'covers_rule',
  '(?spObj1 rdf:type os:Geometry)(?spObj1 os:srid ?sridSpObj1)(?spObj1 os:wkt ?strSpObj1)
       (?spObj2 rdf:type os:Geometry)(?spObj2 os:srid ?sridSpObj2)(?spObj2 os:wkt ?strSpObj2)',
  '(evalSpatialRelation(spObj1,strSpObj1,spObj2,strSpObj2,sridSpObj2,''COVERS'') = 1)',
  '(?spObj1 os:covers ?spObj2)',
  SEM_ALIASES(SEM_ALIAS('os','http://l3i.univ-larochelle.fr/Sido/owlOGCSpatial#')));
```

Code 10.31 – Declarative part of `covers` rule (NT format)

3.5 Equals rule

```
DELETE FROM mdsys.semr_owlSpatialOnto_rb
WHERE upper(rule_name) = upper('equals_rule');

INSERT INTO mdsys.semr_owlSpatialOnto_rb VALUES(
  'equals_rule',
  '(?spObj1 rdf:type os:Geometry)(?spObj1 os:srid ?sridSpObj1)(?spObj1 os:wkt ?strSpObj1)
       (?spObj2 rdf:type os:Geometry)(?spObj2 os:srid ?sridSpObj2)(?spObj2 os:wkt ?strSpObj2)',
  '(evalSpatialRelation(spObj1,strSpObj1,spObj2,strSpObj2,sridSpObj2,''EQUAL'') = 1)',
  '(?spObj1 os:equals ?spObj2)',
  SEM_ALIASES(SEM_ALIAS('os','http://l3i.univ-larochelle.fr/Sido/owlOGCSpatial#')));
```

Code 10.32 – Declarative part of `equals` rule (NT format)

3.6 Inside rule

```
DELETE FROM mdsys.semr_owlSpatialOnto_rb
WHERE upper(rule_name) = upper('inside_rule');

INSERT INTO mdsys.semr_owlSpatialOnto_rb VALUES(
  'inside_rule',
  '(?spObj1 rdf:type os:Geometry)(?spObj1 os:srid ?sridSpObj1)(?spObj1 os:wkt ?strSpObj1)
       (?spObj2 rdf:type os:Geometry)(?spObj2 os:srid ?sridSpObj2)(?spObj2 os:wkt ?strSpObj2)',
  '(evalSpatialRelation(spObj1,strSpObj1,spObj2,strSpObj2,sridSpObj2,''INSIDE'') = 1)',
  '(?spObj1 os:inside ?spObj2)',
  SEM_ALIASES(SEM_ALIAS('os','http://l3i.univ-larochelle.fr/Sido/owlOGCSpatial#')));
```

Code 10.33 – Declarative part of `inside` rule (NT format)

3.7 Overlaps rule

```
DELETE FROM mdsys.semr_owlSpatialOnto_rb
WHERE upper(rule_name) = upper('overlaps_rule');

INSERT INTO mdsys.semr_owlSpatialOnto_rb VALUES(
  'overlaps_rule',
  '(?spObj1 rdf:type os:Geometry)(?spObj1 os:srid ?sridSpObj1)(?spObj1 os:wkt ?strSpObj1)
       (?spObj2 rdf:type os:Geometry)(?spObj2 os:srid ?sridSpObj2)(?spObj2 os:wkt ?strSpObj2)',
  '(evalSpatialRelation(spObj1,strSpObj1,spObj2,strSpObj2,sridSpObj2,''OVERLAPS'') = 1)',
  '(?spObj1 os:overlaps ?spObj2)',
  SEM_ALIASES(SEM_ALIAS('os','http://l3i.univ-larochelle.fr/Sido/owlOGCSpatial#')));
```

Code 10.34 – Declarative part of `overlaps` rule (NT format)

3.8 Touch rule

```
DELETE FROM mdsys.semr_owlSpatialOnto_rb
WHERE upper(rule_name) = upper('touch_rule');

INSERT INTO mdsys.semr_owlSpatialOnto_rb VALUES(
  'touch_rule',
  '(?spObj1 rdf:type os:Geometry)(?spObj1 os:srid ?sridSpObj1)(?spObj1 os:wkt ?strSpObj1)
       (?spObj2 rdf:type os:Geometry)(?spObj2 os:srid ?sridSpObj2)(?spObj2 os:wkt ?strSpObj2)',
  '(evalSpatialRelation(spObj1,strSpObj1,spObj2,strSpObj2,sridSpObj2,''TOUCH'') = 1)',
  '(?spObj1 os:touch ?spObj2)',
  SEM_ALIASES(SEM_ALIAS('os','http://l3i.univ-larochelle.fr/Sido/owlOGCSpatial#')));
```

Code 10.35 – Declarative part of `touch` rule (NT format)