SemSorGrid4Env

FP7-223913

Deliverable

D3.1

Data models and languages for registries in SemsorGrid4Env

Kostis Kyzirakos, Zoi Kaoudi and
Manolis Koubarakis

March 27th, 2009

Status: Final

Scheduled Delivery Date: March 30th, 2009

Executive Summary

The objective of WP3 is to design, implement and deploy an open, dynamic and scalable registry for the Semantic Sensor Grid project. The registry to be developed will allow the description and discovery of sensors, sensor networks and related information. In this document we identify the functionalities that the registry in the SemsorGrid4Env architecture should provide. We deal with two questions that need to be answered: (i) what data model should be used to encode the metadata that the registry will store and (ii) what query language the registry should support to facilitate the users/components of the SemsorGrid4Env architecture. We have chosen to use RDF(S) as the base of our registry metadata model and SPARQL as the base of our query language. We propose the data model stRDF and the query language stSPARQL that will be supported by the SemsorGrid4Env registry. stRDF extends RDF(S) with the ability to represent spatial and temporal data so that sensor metadata can be represented and queried. stSPARQL extends SPARQL so that spatial and temporal data can be queried using a declarative and user-friendly language.

Note on Sources and Original Contributions

The SemSorGrid4Env consortium is an inter-disciplinary team, and in order to make deliverables self-contained and comprehensible to all partners, some deliverables thus necessarily include state-of-the-art surveys and associated critical assessment. Where there is no advantage to recreating such materials from first principles, partners follow standard scientific practice and occassionaly make use of their own pre-existing intellectual property in such sections. In the interests of transparency, we here identify the main sources of such pre-existing materials in this deliverable:

- N/A

Document Information

Contract Number	FP7-223913	**Acronym**	SemSorGrid4Env
Full title	SemSorGrid4Env: Semantic Sensor Grids for Rapid Application Development for Environmental Management		
Project URL	www.semsorgrid4env.eu		
Document URL			
EU Project officer	Daniel Quintart		

Deliverable	**Number**	3.1	**Name**	Data models and languages for registries in SemsorGrid4Env		
Task	**Number**	3.1	**Name**	Propose data models and languages for Semantic Sensor Grid registries		
Work package	**Number**	3				
Date of delivery	**Contractual**		28/2/2009	**Actual**	27/3/2009	
Code name	D3.1			**Status**	draft ☐	final ☑
Nature	Prototype ☐ Report ☑ Specification ☐ Tool ☐ Other ☐					
Distribution Type	Public ☑ Restricted ☐ Consortium ☐					
Authoring Partner	National and Kapodistrian University of Athens					
QA Partner	University of Manchester					
Contact Person	Kostis Kyzirakos					
	Email	kkyzir@di.uoa.gr	**Phone**	+30 210 727 5140	**Fax**	
Abstract (for dissemination)						
Keywords						

Version log/Date	**Change**	**Author**
0.1	**Template**	**A. Ibarrola**
0.2	**Contents**	**M. Koubarakis**
0.9	**Final draft to be QAed**	**M. Koubarakis, Kostis Kyzirakos, Zoi Kaoudi**
0.95	**QA**	**Alasdair Gray**
1	**Final version**	**M. Koubarakis, Kostis Kyzirakos, Zoi Kaoudi**

Project Information

This document is part of a research project funded by the IST Programme of the Commission of the European Communities as project number FP7-223913. The Beneficiaries in this project are:

Partner	Acronym	Contact
Universidad Politécnica de Madrid (Coordinator)	UPM	Prof. Dr. Asunción Gómez-Pérez Facultad de Informática Departamento de Inteligencia Artificial Campus de Montegancedo, sn Boadilla del Monte 28660 Spain #e asun@fi.upm.es #t +34-91 336-7439, #f +34-91 352-4819
The University of Manchester	UNIMAN	Prof. Carole Goble Department of Computer Science The University of Manchester Oxford Road Manchester, M13 9PL, United Kingdom #e carole@cs.man.ac.uk #t +44-161-275 61 95, #f +44-161-275 62 04
National and Kapodistrian University of Athens	NKUA	Prof. Manolis Koubarakis University Campus, Ilissia Athina GR-15784 Greece #@ koubarak@di.uoa.gr #t +30 210 7275213, #f +30 210 7275214
University of Southampton	SOTON	Prof. David De Roure University Road Southampton SO17 1BJ United Kingdom #@ dder@ecs.soton.ac.uk #t +44 23 80592418, #f +44 23 80595499
Deimos Space, S.L.	DMS	Mr. Agustín Izquierdo Ronda de Poniente 19, Edif. Fiteni VI, P 2, 2º Tres Cantos, Madrid – 28760 Spain #@agustin.izquierdo@deimos-space.com #t +34-91-8063450, #f +34-91-806-34-51
EMU Limited	EMU	Dr. Bruce Tomlinson Mill Court, The Sawmills, Durley number 1 Southampton, SO32 2EJ – United Kingdom #@ bruce.tomlinson@emulimited.com #t +44 1489 860050, #f +44 1489 860051

TechIdeas Asesores Tecnológicos, S.L.	TI	Mr. Jesús E. Gabaldón C/ Marie Curie 8-14 08042 Barcelona, Spain #@ jesus.gabaldon@techideas.es #t +34.93.291.77.27, #f ++34.93.291.76.00

Contents

List of Figures

Chapter 1

Introduction

The objective of WP3 is to design, implement and deploy an open, dynamic and scalable registry for the Semantic Sensor Grid software architecture to be defined in WP1. The registry to be developed will allow the description and discovery of sensors, sensor networks and related information.

We will design and implement the registry of the SemsorGrid4Env infrastructure based on service-oriented-architecture (SOA) technologies as they will be specified in WP1, and ideas/implementation techniques from our system Atlas (`http://atlas.di.uoa.gr/`). Atlas is a P2P network for the distributed storage, querying and update of RDF(S) metadata describing Web or Grid resources. It was developed in the FP6 project OntoGrid (`http://www.ontogrid.net/`) and was subsequently extended with more functionality in the Greek national project "Peer-to-Peer techniques for Semantic Web Services".

In WP1, a service-oriented architecture will be specified and will form the basis for the middleware of SemsorGrid4Env. A service-oriented architecture prescribes three different roles that a software component that is part of the architecture can play. A *service provider* exposes its functionality in the form of services and publishes the description of a service (i.e., *metadata* about the service) into a *service registry*. Then, a *service client* can lookup these descriptions in the service registry to discover the services that meet its requirements. Once an appropriate service is found, the client binds to the provider to consume the service. A high level view of this interaction pattern taken from [74, 16] is shown in Figure 1.1. In SemsorGrid4Env, the component that will implement the service registry will virtualize a *metadata store* and the means to populate, query and maintain metadata about available services.

The goal of the SemsorGrid4Env architecture is to extend the service-oriented architecture to cover all the extra requirements needed in semantic sensor grids. In this deliverable, we will try to identify the extra functionality that the registry in the SemsorGrid4Env architecture (and generally a registry for semantic sensor grids) should provide. Realizing a registry for semantic sensor grids is a challenging task and raises a number of interesting research questions that we will tackle

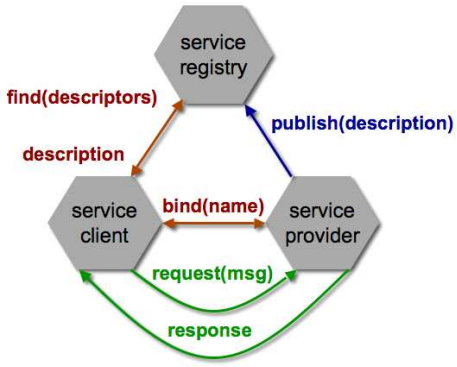

Figure 1.1: Service-oriented architecture

with our work in WP3. In this deliverable, we will deal with the first two questions that need to be answered: (i) what data model should be used to encode the metadata that the registry will store and (ii) what query language the registry should support to facilitate the users/components of the SemsorGrid4Env architecture.

Metadata for sensors and sensor networks can be distinguished into *thematic* (e.g., the sensor measures windspeed), *spatial* (e.g., the sensor is located in Chichester Bar) and *temporal* (e.g., the sensor was dead throughout the last two weeks). This has been explained clearly in previous works such as SensorML [44] and the Semantic Sensor Web vision [1, 91]. This is also apparent in the requirements of the two use cases of SemsorGrid4Env "Fire Risk Monitoring and Warning in NorthWest Spain" and "Coastal and Estuarine Flood Warning in Southern UK" described in Deliverables [20] and [14].

We have chosen to use RDF(S) [63, 10] as the base of our registry metadata model and SPARQL [26] as the base of our query language. The motivation for this choice is pragmatic; we sacrifice the expressivity of ontology languages such as OWL [68] in order to concentrate on efficient and scalable query evaluation which is possible in RDF(S). However, RDF(S) can only represent thematic metadata and needs to be extended if we want to model *spatial* and *temporal* information.

Until recently, little attention has been paid to the problem of extending RDF(S) to support spatial and/or temporal information. The first and essentially the only works that deal with representing temporal information in RDF are [34, 33, 41]. More recently, Perry has presented a spatial and temporal extension to RDF and a query language for this extension called SPARQL-ST [77]. In his work, geometries can be represented in RDF(S) using a spatial ontology based on the GeoRSS GML specification [93] and queried by adding appropriate spatial primitives to SPARQL. Thematic or spatial data in [77] can also have an associated *valid time*; for this extension, the model of [34, 33] is used.

The work presented in this deliverable has been influenced by the language SPARQL-ST [77] but our basic approach representing spatial data in RDF is

different. We follow the main ideas of *spatial constraint databases* [80, 84, 61], and represent spatial objects as *quantifier-free formulas in a first-order logic of linear constraints*. These formulas can capture a great variety of spatial geometries, e.g., k-dimensional unions of convex polygons possibly with holes, thus they give us a lot of expressive power.[1] In terms of the W3C specification of RDF, this spatial extension can be realized with the introduction of a new kind of *typed literals*: quantifier-free formula with linear constraints (the datatype of these literals can easily be defined in XML Schema).

The main contributions of this deliverable are the following:

- We survey related work in the areas most relevant to this deliverable. We start with an overview of the work accomplished already for registries in other sensor network architectures. Then, we survey the state-of-the art in query languages for temporal and spatial information for various models such as databases, XML-based models, RDF-based models, description logics and OWL (Chapter 2).

- We present the data model stRDF and the query language stSPARQL that will be supported by the SemsorGrid4Env registry (Chapter 3). stRDF extends RDF with the ability to represent spatial and temporal data so that sensor metadata can be represented and queried. stSPARQL extends SPARQL so that spatial and temporal data can be queried using a declarative and user-friendly language.

Chapter 4 presents our conclusions and discusses future work in the context of the project.

[1] There has been work on polynomial constraint databases as well, but the evaluation of queries in this case is more difficult[60].

Chapter 2

Related Work

In this chapter we present a short survey of related work in the areas most relevant to this deliverable. We start with previous work on registries for architectures related to the service-oriented architecture we will be developing in SemsorGrid4Env. Then, we discuss work on geospatial information discovery and retrieval. Finally, we survey the state-of-the art in data models and query languages for temporal and spatial information in the following research areas: (i) databases, (ii) XML, (iii) RDF, and (iv) description logics and OWL. Since our work will be based on RDF, we pay more attention to the RDF extensions that are currently the state-of-the-art regarding temporal and spatial information.

2.1 Registries

There have been several research projects of varying ambition, scale and output in the area of Sensor Internet/Web/Grid [1, 91, 29, 73, 45, 70, 44]. Below we discuss some of these projects, and point out interesting research that has been done in their context regarding registries for sensors, sensor networks and related services.

IrisNet[1] (Internet-scale Resource-Intensive Sensor Network Service) is a past project of Intel Research that was probably the first research effort to envision a world-wide sensor web built from inexpensive common computing hardware [29]. The IrisNet software lets users query globally distributed collections of high-bit-rate sensors powerfully and efficiently. IrisNet takes a database-centric approach in its design and proposes to view the sensor network as a distributed database queried in a high-level query language. Sensor service authors can specify new services through a set of XML documents and service consumers can pose XPATH queries on wide area sensor databases. In [19], the authors describe a scalable implementation of their ideas where a logical hierarchical view of the fragmented XML data is kept as a single XML document, while XPATH queries

[1]Retrieved March 1, 2009, from `http://www.intel-iris.net/`

can be evaluated over this single XML document even if the fragmentation of the data is constantly changing. Thus, in IrisNet, data produced by the sensors and metadata describing the available sensors belong conceptually to the same XML database and are queried with the same query language (XPATH).

The *SenseWeb/SensorMap*[2] project of Microsoft Research provides a common platform for users to easily publish their sensor data and enable queries over live sensor data sources. The SenseWeb platform uses as a registry a relational database, called GeoDB, to store sensor metadata (e.g., publisher name, sensor location, sensor name, sensor type, data type, unit, sensor data access methods and free text descriptions) [73, 89]. GeoDB uses a hierarchical triangular mesh [96] in order to efficiently support SQL queries over geographic data.

Hourglass[3] is a past project at Harvard that developed a data collection network for connecting a wide range of heterogeneous sensor networks, services and applications [45]. Services in Hourglass can be described using a data model based on topics and predicates such as the one in [81]. Assuming this data model, [45] states that service publication and discovery can be implemented using various distributed system technologies (e.g., DHTs) but gives no additional details of the Hourglass registry implementation.

The above projects make no use of semantics to describe sensors, sensor networks or related information. The *Semantic Sensor Web*[4] [91] proposal envisions the integration of different kinds of sensor networks, historical data sources and specialized software (e.g., for environmental modelling) in order to develop applications that have not been feasible so far. To achieve this interoperability, Sheth and colleagues propose to annotate sensor data with semantic metadata. Annotations can be expressed in RDFa[5], a W3C Recommendation which provides a set of attributes that can be used to attach semantic metadata within an XHTML document. Then, by using this metadata and the appropriate ontologies and rules, one can reason about sensor data. The work of Sheth and colleagues does not address registries for Semantic Sensor Web explicitly [91, 92, 78, 79], but their work on extending RDF with a spatial and temporal dimension, and the development of the query language SPARQL-ST [77] is a solid foundation for a data model and query language for Semantic Sensor Web registries. The same approach is followed by us in this deliverable.

The Swiss Experiment (*SwissExp*)[6] [18, 70] is a collaboration of environmental science and technology research projects of various research institutions across

[2]Retrieved March 1, 2009, from `http://atom.research.microsoft.com/sensewebv3/sensormap/`

[3]Retrieved March 1, 2009, from `http://www.eecs.harvard.edu/~syrah/hourglass/index.shtml`

[4]Retrieved March 1, 2009, from `http://knoesis.wright.edu/research/semsci/application_domain/sem_sensor/`

[5]Retrieved March 1, 2009, from `http://www.w3.org/2006/07/SWD/RDFa/`

[6]Retrieved March 1, 2009, from `http://www.swiss-experiment.ch`

Switzerland which has been created to provide a platform for large scale sensor network deployment, querying and exploitation. In [18], the Sensor Metadata Repository of SwissExp is introduced. Through a semantic wiki, all data and metadata are integrated and stored in the form of RDF and then different data sources can be queried through a Netapi[7] SPARQL endpoint to the wiki.

Finally, the SANY Sensor Anywhere - IST FP6 Integrated Project[8] is targeting at delivering a generic service-oriented infrastructure for sensor networks but also sensor services specific for environmental applications. The registry in SANY behaves as a cascading catalogue where various underlying data sources may be accessed. The registry in SANY is based on the scheme of geospatial catalogue services, and specifically of the ORCHESTRA Catalogue Service with a semantically-enriched catalogue interface which can be adjusted to any application-specific domain [99, 39]. The ORCHESTRA catalogue service acts like a broker that mediates the client requests to the underlying catalogue protocols. These protocols may be be an OGC CSW/ISO application profile[9], an OGC CSW/ebRIM application profile[9], another SANY or ORCHESTRA catalogue or a web search engine such as Yahoo. Resource discovery is made in a publish-find-bind fashion.

2.2 Discovery and retrieval of Geospatial Information

A common problem that occurs with conventional GIS technology is that multiple data sets exist in a variety of different formats and may be stored anywhere in a distributed computing environment. Catalogues are used to allow a client to find geospatial resources that are available on a server and satisfy the client's requirements, without requiring from the client to have any prior knowledge of the existing geospatial resources. According to the Open Geospatial Consortium (OGC)[10], *the term "Catalog" is used to describe the set of service interfaces which support organization, discovery, and access of geospatial information*[11]. A catalogue can be considered a database that stores *Geographic Information* (GI) about *geographic features* (objects that represent geographic entities), collections of geographic features, *catalog interfaces* and *geoprocessing services* (services for data manipulation). A service provider that want to expose some geospatial resources, publishes to the catalogue the metadata that describe the offered re-

[7]Retrieved March 1, 2009, from `http://www.w3.org/Submission/2003/SUBM-rdf-netapi-20031002/`

[8]Retrieved March 1, 2009, from `http://www.sany-ip.org/`

[9]Retrieved March 1, 2009, from `http://www.opengeospatial.org/standards/cat`

[10]Retrieved March 1, 2009, from `http://www.opengeospatial.org/`

[11]The OpenGIS Abstract Specification Topic 13: Catalog Services. Retrieved March 1, 2009, from `http://portal.opengeospatial.org/files/?artifact_id=901`

sources. A client may use the catalogue to search for and retrieve a geospatial resource that satisfies her requirements based on the metadata that describe the resource's properties. In most current geospatial catalogues, users can perform just keyword-based search, so an important question that rises is how catalogues could perform *semantic* discovery.

In [27], Klein propose an ontology-based approach that employ subsumption reasoning for discovering and retrieving Geographic Information in Spatial Data Infrastructures. The ontology-based discovery of GI is based on semantic mathmaking between the geographic feature types and a user's query. If *one* suitable GI source that provides the required information is discovered, a Web Feature Service (WFS)[12] is used to retrieve the discovered information. The WFS standard defines interfaces and operations for accessing and manipulating geographic features. A global shared vocabulary that contains the basic terms of the domain is used. The shared vocabulary is represented as an ontology (domain ontology). Each information source may use a separate application ontology that is directly derived from the domain ontology. As a result, the concepts in different application ontologies are comparable since they share the same domain ontology. The ontologies are expressed using a Description Logic (DL) notation and only the TBox language features are used: concept definition, concept inclusion and role definition.

For discovering GI, it is sufficient to describe and reason about application concepts since the proposed approach is based on semantic matchmaking between DL concepts (that represent geographic feature types) and the user's query. The query can be a concept from the application ontology or it can be defined based on terms of the shared vocabulary. A concept is considered a match if it is equal or subsumed by the query concept. For retrieving GI, it is required to provide more specific information on the feature type's structure that are represented by application concepts. The author uses *registration mappings* to describe the relationship between the structure of a feature type and an application concept. Registration mappings are used in GI retrieval to specify a query filter for a WFS query. The author do not expect the users to formulate complex DL query concepts, so an intuitive (SQL-like) query language and and a graphical user interface is provided. The query statement is translated into one or several query concepts and the suitable WFS that provide the appropriate feature type is discovered. Finally, the actual WFS query filter is automatically created based on the information provided by the registration mappings and the discovered information is retrieved.

In [27], the author proposes using spatial relations for semi-automatic semantic geodata annotation. Spatial relations may be used to define and identify spatial concepts. The first step in the proposed methodology is to identify the concept definitions from the domain ontology that involves spatial relations. Spa-

[12]Retrieved March 1, 2009, from `http://www.opengeospatial.org/standards/wfs/`

tial relations may be classified into *topological relations*, *direction relations* and *distance relations*. Topological relations are useful for concept definitions since they stay invariant under transformations. The author are indecisive on the choice of the representation language for the ontological knowledge. DL-based ontology languages are not expressive enough for describing the desired kinds of concept interdependencies, while First Order Logic provides the desired expressivity. The identified spatial relations are translated into a spatial analysis method. The spatial analysis method is a combination of primitive operators defined in the ISO 19100 series of standards (e.g., the *intersects* operation that returns true if two geometries intersect) and the OGC specifications (e.g., *Web Feature Service* Implementation specification). At the final step, a reference dataset is used to calculate relationships between well-known entities and the entities of the dataset to be annotated.

In [27], the author presents an additional rule-based methodology for semantic geodata annotation, where rules are used to partially automate the annotation process. Rules define spatial conditions for identifying geospatial concepts. The author chooses OWL-DL for the ontology representation language since it provides subsumption reasoning, needed for automatic matchmaking. Rules are used to express more complex relationships, especially by using variables. The Semantic Web Rule Language (SWRL)[13] is used to express rules that define the conditions for the classification of the features of the dataset to be annotated. The process is divided in two parts. The OWL-DL encoded ontologies are used for subsumption reasoning while the SWRL rules are checked on the instance level for classifying features in a dataset.

In [64], the authors demonstrate the use of semantic service descriptions for service chaining. Service chaining involves *service discovery*, *abstract service composition* (identification of the service chain's functionality), *concrete service composition* (identification of the service chain's messaging) and *service execution*. The authors present two prototypes: the *GeoMatchMaker* that performs the service discovery and the abstract service composition, and the *Integrated Component Designer* that performs the concrete service composition and the service execution. The services' descriptions in [64] are provided in a semantic framework. A combination of ontologies is used to describe *feature concepts* (features defined by their thematic and spatial attributes), *feature symbols* (abstract elements that define a feature in an object/field model) and *geo-operations* (OWL-S[14] based definitions of atomic GIS operations). In this semantic framework, semantic service metadata are represented with classes and/or class instances. A class definition of a service operation type may be described with a Description Logic axiom as a subclass of an other service operation type, with restrictions on the input and output parameters. Alternatively, the operations of a service may be given

[13]Retrieved March 1, 2009, from `http://www.w3.org/Submission/SWRL/`
[14]Retrieved March 1, 2009, from `http://www.daml.org/services/owl-s/`

with individuals that instantiate the concepts used in class definitions. Both class and individual definitions are encoded in OWL. The semantic descriptions of Web Services are linked with the services' syntactic descriptions using OWL-S or WSDL-S[15]. The authors assume that a common set of geo-ontologies are used by all participants for annotating the services and the semantic descriptions that are created by service providers are stored in a *Web services repository*. The GeoMatchMaker perform the geo-service discovery by identifying the service advertisements that are stored in the Web services repository and matches a service request. Depending on whether the service operations are described with class-based or individuals-based definitions, there are four possible types of concept matching. Matching between class descriptions for example is done by TBox reasoning, while matching between a class-based description and an individual-based description is done by ABox reasoning with the RacerPro[16] reasoner.

The SWING project[17] aims at deploying Semantic Web Service (SWS) technologies for geospatial data sources in a mineral resources application. Multiple WSML ontologies are created to formalize several aspects of mineral resources. The annotation and discovery of *Web Feature Services* (WFS) and *Web Processing Services* (WPS) is based on a light-weight approach that uses Datalog query containment for the required matching. The discovery is split in two steps. The geospatial part of the discovery is executed in a commercial *Geospatial Web Service Catalogue* (built by Ionic Sofrware), and the discovered service descriptions are sent to a *Web Service Execution Environment platform* where the service descriptions that satisfy the semantic query are identified. Both WFS and WPS are described in Datalog. WFS are described only by their postcondition while WPS are described by a combination of precondition and postcondition. The web services are described by means of WSML-Flight[18]; WSML-Flight allows Web Service description by Datalog rules, extended with inequality and (locally) stratified negation. Discovery queries are also described by Datalog rules and the matching is based on query containment checking: a web service is a match if it subsumes the goal query.

In [67] the author suggests an architecture for a web service catalogue which takes into account the diversity of existing web service description standards. Ontologies are used for annotating service descriptions independently from the service description standards that are used to describe a service. Schema-based service descriptions are transformed to ontology-based descriptions and stored to a centralized catalogue. The transformation process must not lead to information loss, so that the specific advantages of a schema-based service description standard are not ignored by the registry. The implemented semantic catalogue

[15]Retrieved March 1, 2009, from `http://www.w3.org/Submission/WSDL-S/`

[16]Retrieved March 1, 2009, from `http://www.racer-systems.com/`

[17]Retrieved March 1, 2009, from `http://www.swing-project.org/`

[18]Retrieved March 1, 2009, from `http://www.wsmo.org/TR/d16/d16.1/v1.0/#wsml-flight`

uses the *Web Service Modeling Ontology* (WSMO)[19] and the associated *Web Service Modeling Language* (WSML)[20]. Loss of information may occur during the transformation process, if the imported semantic service description uses a more expressive formal language than WSML. The service model used for representing service descriptions consists of a common and an extended model where primary and secondary aspects are stored respectively. The primary aspects describes the functional properties of the service and a fixed set of metadata attributes, while the secondary aspects include additional metadata (for example the spatial extent or information regarding the measured phenomena) and secondary functional properties. Combination of multiple search criteria may be used for service discovery: keywords, spatial features and semantic queries may be used to increase the precision of the search results. A declarative language is not used for querying the catalogue. Instead, semantic queries are formulated by a semantic query wizard that provides to the user predefined goals.

2.3 Database Models for Spatial and Temporal Information

In this section, we survey the work done in the vast research area of database models for spatial and temporal information. We distinguish between three different fields: spatial databases, temporal databases and spatiotemporal databases.

2.3.1 Spatial databases

Spatial database systems offer the underlying database technology for geographic information systems and other related applications, and have gained a lot of attention over the years [80]. Research in *spatial databases* has concentrated in the areas of data modeling, query languages, data structures and algorithms, and system architectures. In the area of spatial data modeling, researchers have studied possible representations of geographic data in a database. Work on query languages mainly concentrated on designing query languages for spatial DBMS (e.g., spatial extensions of the relational algebra or SQL) and graphical representations for both the input queries and the output results. Work on data structures and algorithms involved the actual implementation of the proposed spatial algebras and the system's query processing architecture. Finally, the area of system architecture dealt with integrating the spatial extensions to a standard DBMS architecture. For a nice survey of spatial database research (as of that time) the interested reader might refer to [35].

[19]Retrieved March 1, 2009, from `http://www.wsmo.org/`
[20]Retrieved March 1, 2009, from `http://www.wsmo.org/wsml/`

Most of the advanced commercial DBMS available today offer support for spatial data. For example, PostGIS[21] is an open source software which extends PostgreSQL with geographic data types, operators and indexes and follows the standards of SQL and OpenGIS implementation specification. PostGIS offers geometry types for points, polygons, multipoints, multilinestrings, multipolygons and geometrycollections, spatial predicates for determining the interactions of geometries, spatial operators for determining geospatial measurements, spatial operators for determining geospatial set operations and R-tree spatial indexes for faster query processing. Oracle Spatial[22] enables users to manage geographic and location-data in a native type within an Oracle database with providing a schema that prescribes the storage, syntax, and semantics of supported geometric data types, a spatial index system, operators, functions, and procedures for performing area-of-interest queries, spatial join queries, and other spatial analysis operations.

Various standards exist for representing and manipulating spatial data in relational systems. The ISO 13249 SQL/MM[23] define a set of types and methods for representing, processing, storing and querying spatial data. The OpenGIS Simple Features Specification for SQL[24] provide a standardized way for storing and accessing spatial data in relational and object-oriented databases. Oracle provides different datatypes for each standard e.g., the `ST_Geometry` datatype is based on the ISO SQL/MM specification while the `BLOB` datatype is used to store spatial data according to the OGC Well-Known Binary representation (fragment of the OGC's Simple Features specification). PostGIS also provides different datatypes for each standard e.g., the `ST_Geometry` datatype is based on the ISO SQL/MM specification while the `Geometry` datatype follows the OpenGIS Simple Features Specification for SQL.

Since this deliverable proposes a new spatial data model and query language, a more detailed survey of spatial data modeling work done in academia and industry might have been useful. However, due to space considerations, in the rest of this related work section, we concentrate only on spatial data models that are of immediate interest.

Spatial data models based on constraints

The first paper to propose to model spatial data in a database using constraints (and also to kick-start the area of research that we today call *constraint databases*) was [48, 49]. The fundamental breakthrough of this paper is to propose that a tuple in a relational database (e.g., $(John, Athens)$ representing the information

[21]Retrieved March 1, 2009, from `http://postgis.refractions.net/`

[22]Retrieved March 1, 2009, from `http://www.oracle.com/technology/products/spatial/index.html`

[23]Retrieved March 1, 2009, from `http://www.iso.org/iso/iso_catalogue/catalogue_tc/catalogue_detail.htm?csnumber=38651`

[24]Retrieved March 1, 2009, from `http://www.opengeospatial.org/standards/sfs/`

that John lives in Athens) can be understood as a conjunction of constraints (e.g., $x = John \wedge y = Athens$) in an appropriate constraint language (in this case the language of equality constraints over an infinite domain). Then, relations can be understood as disjunctions of conjunctions of constraints. This view of relational data allows us to represent *infinite* data (e.g., an infinitely periodic event or a geometry consisting of an infinite number of points) by a *finite constraint relation*. [48, 49] shows that constraint databases can be queried using the same languages that we have for traditional relational databases (e.g., relational algebra, relational calculus, datalog, SQL) with no extra data complexity. At the basis of this result, lies the fact that query evaluation for constraint relations can be recast as *quantifier elimination* [12, 25] in the logical theory of the constraints used in the database and the query.

Koubarakis later extended the ideas of [48, 49] to the modeling of *indefinite* (or *incomplete*) information in databases and knowledge bases [54, 56, 55, 57]. Koubarakis developed the scheme of *indefinite constraint databases* which extends the constraint databases ideas of [48] to include indefinite information in the style of [42, 30, 2]. To express queries in this scheme, Koubarakis proposed *modal* query languages i.e., query languages with operators that give us the ability to ask "possibility" and "certainty" queries - a very useful feature in any indefinite information setting. Like in the definite information setting of [48], Koubarakis showed that there is no increase in data complexity when we query indefinite constraint databases when compared with traditional indefinite databases as studied in [42, 30, 2].

Since its inception in [48, 49], the constraint databases research area has produced many interesting results; some of them surveyed in the books [84, 61]. The data model and query languages developed in Chapter 3 of this deliverable have been influenced by the practical approach to spatial constraint databases followed by CSQL [62] and Dedale [32, 86]. Constraint databases is still an active area of research with the potential to offer a lot to the development of GIS and related application areas; see the recent manifesto in [9].

The first influential spatial constraint database system was DEDALE [32, 86]. In DEDALE, all data (spatial and non-spatial) are stored in a uniform way by using a *nested relational model* with one level of nesting. Spatial geometries are modeled by finitely representable pointsets i.e., subsets of k-dimensional real space that can be finitely represented by a quantifier-free formula in a first-order logic of linear constraints over the real numbers. [32, 86] present an algebraic language for DEDALE which extends the relational algebra with the following operations: tuple construction, constraint projection (variable elimination), restructuring and union nesting. [32, 86] also give a declarative query language with SQL syntax which offers the same expressive power as the algebra. DEDALE was implemented fully in the context of project CHOROCHRONOS [58] but the relevant code is not available any longer.

In [62], a constraint query language is developed based on SQL. CSQL pro-

vides the full query facilities of SQL for the non-spatial part of the database and extends it for handling spatial data as well. CSQL assumes the same data model as DEDALE and presents a linear constraint algebra called CALG, which extends the relational algebra with constraint projection (variable elimination), spatial selection, spatial construction and spatial aggregation. In order to implement CALG, a Constraint Abstract Data Type is proposed. CSQL has not been implemented in a real system.

Another relevant constraint database system is MLPQ/PReSTO[25] developed by Peter Revesz and his team at the University of Nebraska. MLPQ/PReSTO is based on the constraint database model of [48] and offers SQL and Datalog-based query languages. It also has some nice graphical and animation capabilities that allow its easy use in various GIS applications. To the best of our knowledge, MLPQ/PReSTO is the only constraint database system that is available today.

2.3.2 Temporal databases

The field of databases has also seen a lot of work on *temporal databases* including work on foundations, query languages, indexing, optimization, and so on. [95] proposed the following classification of DBMS depending on their support for time:

- *Snapshot databases.* All conventional databases belong to this category. The information in these databases corresponds to a snapshot of the world at a particular point in time. Updates are destructive and old information is not retained. The only kind of temporal information supported by such systems is *user-defined time* (e.g., the date of birth of a person can be stored as a string).

- *Rollback databases.* Rollback databases maintain a complete record of the evolution of the database. Intuitively, they can be understood as sequences of snapshot databases indexed by *transaction time* i.e., the time a transaction changing the current database state committed (or the time the system starts believing the new information about the world). In rollback databases, updates are performed with respect to the current state of the database while queries can inquire about past states as well.

- *Historical databases.* Rollback databases keep a record of the changes as registered in the database. Historical databases, on the other hand, keep a record of the changes as they take place in the real world. Intuitively, they can be understood as sequences of snapshot databases indexed by *valid time* i.e., the time a piece of information is true in the real world. In historical databases, the users can pose queries to find out about the world at any

[25]Retrieved March 1, 2009, from `http://cse.unl.edu/~revesz/MLPQ/mlpq.htm`

point in time. Any errors about the past can be corrected but no record of past database states is kept.

- *Bitemporal databases.* Bitemporal databases combine the advantages of rollback and historical databases. Intuitively, they are sequences of historical databases indexed by transaction time. The user is allowed to inquire about the world at some point in time according to the information present in the database at some other point in time. The user is also allowed to modify the current state of the database in order to add, delete, or modify historical information about the world. [94] presents some early work on a query language for a bitemporal database management system.

The work on temporal query languages concentrated mostly on the language TSQL2 [3], a consensus temporal query language developed by a committee of leading temporal database researchers as an extension to SQL-92. TSQL2 could have influenced the SQL3 standard [83] as discussed in `http://www.cs.arizona.edu/people/rts/sql3.html`, but it does not appear to have influenced implementations of current commercial DBMS. A different approach to introducing time in databases, which is simpler than the TSQL approach, is offered by Date, Darwin and Lorentzos in [17]. Some (dated) surveys of work on temporal query languages and temporal database management include [13, 46].

As one might expect, constraint databases is also very useful for the representation of temporal information, and this approach has been followed by several temporal database researchers. See, for example [97] for an approach that extends SQL, [47] for a constraint-based model of periodic data and [54, 55] for how to model indefinite temporal information using constraints.

2.3.3 Spatiotemporal databases

Database researchers have also studied *spatiotemporal databases* which deal with geometries changing over time, e.g., as in the case of *moving objects*. Important research in spatiotemporal databases was carried out in project CHOROCHRONOS whose results have been summarized in [58]. CHOROCHRONOS achieved results in many subareas of spatiotemporal databases: user interfaces, design methodologies, data models and query languages, query processing algorithms, storage structures and indexes and prototype systems [58]. One of the highlights of the CHOROCHRONOS work has been work on spatiotemporal data models and query languages. Two competing data models were proposed: one based on abstract data types for moving objects developed by Güting and colleagues [36] and another based on constraints developed by Grumbach, Koubarakis, Rigaux, Scholl and colleagues [32, 86, 31, 59]. Both approaches are explained in great depth in the books [37, 85].

Finally, another interesting approach to a data model and query language for moving objects which also captures uncertainty is presented in [98].

2.4 XML-based Languages for Spatial and Temporal Information

Recently there has also been work on temporal XML by database researchers. For example, in [28], a temporal query language for XML, called τXQuery, is presented in which the authors add valid time support by minimally extending the syntax and semantics of XQuery. In another relevant paper [87], the authors present a data model for tracking historical information in an XML document, a way for summarizing and indexing temporal XML documents, and TXPath, a temporal XML query language that extends XPath 2.0.

To the best of our knowledge there has not been so far significant work in extending the XML data model and related query languages with geospatial data, although such work is in progress at the University of Twente in the context of MonetDB[26] and possibly other DBMS. We expect more interesting work in this area to appear soon. However, since XML is the technology of choice for exchanging information on the Web, there has been significant previous work in *using* XML to represent geospatial features. This work is relevant to this deliverable and it is surveyed below.

The Geography Markup Language (GML) [43] is the most common XML-based encoding standard for the representation of geo-information. GML was developed by the OpenGIS Consortium[27] and is based on the OGC Abstract Specification[28], the conceptual foundation for OGC interoperability specifications. GML provides XML schemas for defining a variety of concepts that are of use in geography: geographic features, geometry, coordinate reference systems, topology, time and units of measurement. Initially, the GML abstract model was based on RDF but later the consortium introduced XML schemas in GML structure in order to facilitate the integration of existing spatial databases, whose relational schema can be defined more easily with XML Schemas. The GML *profiles* are logical restrictions of GML that might be of use to applications that do not want to use the whole of GML. GML profiles can be expressed through an XML document, an XML schema, or both. Some of the profiles that have been proposed for public use are: (i) Point Profile, (ii) GML Simple Features Profile, (iii) a GML profile for JPG, and (iv) a GML profile for RSS. It should be noted that GML profiles are different from application schemas. The profiles are part of the GML namespaces (Open GIS GML) and define restricted subsets of GML. Applications schemas are XML vocabularies that are application-specific and are valid inside the application-specific namespaces.

GeoRSS [93] is a specification that enables RSS feeds to encode location. GeoRSS-Simple and GeoRSS-GML are two different encodings of GeoRSS. GeoRSS-

[26]Retrieved March 1, 2009, from `http://monetdb.cwi.nl/`

[27]Retrieved March 1, 2009, from `http://www.opengeospatial.org/`

[28]Retrieved March 1, 2009, from `http://www.opengeospatial.org/standards/as/`

Simple is a very lightweight format that developers and users can quickly and easily add to their existing feeds with little effort. It supports basic geometries (point, line, box, polygon) and covers the typical use cases when encoding locations. For a more feature-rich option, GeoRSS GML is a formal GML profile, and supports a greater range of features, notably coordinate reference systems other than WGS-84[29] latitude/longitude. Both encodings can be serialized in various flavors of RSS, or RDF, or XHTML.

The constantly growing use of sensor networks created the need for a common framework for describing sensor-related information. SensorML [44] is an approved OGC standard developed as part of the Sensor Web Enablement (SWE) initiative[30]. SensorML can be used to define the geometric, dynamic, and observational characteristics of a sensor using an XML encoding. Such sensor descriptions can be made available on the Web along with sensor observations, allowing applications to automatically discover and use remote sensors. SensorML is designed to support a wide range of sensors including both dynamic and stationary platforms, and both in-situ and remote sensors. The purpose of a sensor description language such as SensorML is to provide general sensor information in support of data discovery, support the processing and analysis of the sensor measurements, support the geolocation of the measured data, provide performance characteristics (e.g., accuracy, threshold, etc.) and, finally, support the archiving of fundamental properties and assumptions regarding the sensors. SensorML provides information about observation characteristics such as physical properties measured (e.g., radiometry, temperature, concentration, etc.), quality characteristics (e.g., accuracy, precision) and response characteristics (e.g., spectral curve, temporal response, etc.). It also provides geometry characteristics such as size, shape, spatial weight function of individual samples and geometric and temporal characteristics of sample collections. Finally, SensorML is also used for encoding metadata such as overall information about the sensor, and history and reference information supporting the SensorML document.

2.5 RDF Extensions for Spatial and Temporal Information

Up to now little attention has been paid on extending RDF and RDF Schema for spatial and temporal data representation and reasoning. In this section, we present the extensions to RDF(S) that have been proposed in order to enable the representation of spatial information, temporal information or both.

[29]Retrieved March 1, 2009, from `http://en.wikipedia.org/wiki/WGS84`

[30]Retrieved March 1, 2009, from `http://www.opengeospatial.org/projects/groups/sensorweb`

2.5.1 Spatial information in RDF

The use of RDF to represent spatial data in the Semantic Web is proposed in [53] where the prototype system SPAUK is presented. In SPAUK, geometric attributes of a resource (e.g., location of a gas station) are represented in RDF by introducing a blank node for the geometry, specifying the geometry using GML vocabulary [43], and associating the blank node with the resource using GeoRSS vocabulary [93]. Queries are expressed in the SPARQL query language utilizing appropriate geometric vocabularies and ontologies (e.g., the topological relationships of RCC [15]). The main assumption of this work is that SPARQL should not be extended with new features for querying spatial data; instead, the existing features of SPARQL together with spatial vocabularies should be utilized.

[53] does not specify a semantics for query answering. From the example given in [53], we conjecture that such a semantics needs to rely on a model theory (or an axiomatization) which combines the model theory (or axiomatization) of the spatial vocabularies/ontologies used (e.g., RCC) and the RDF model theory (or an equivalent axiomatization). We expect such a semantics to be complicated and will certainly be an extension of the standard semantics for SPARQL query evaluation. Thus, although [53] does not change the SPARQL syntax, the semantics of query answering are not the standard ones. This is not said explicitly in [53] but it can be deduced from the comments regarding how complicated the spatial query processing part of SPAUK is, and what will happen if the spatial ontologies used are extended in ways that the system does not understand (pages 798-799 of [53]).

SPAUK has been implemented by storing RDF triples in Jena and using an in-memory grid file to index the geometries.

Kolas has revisited the problem of defining a Semantic Web data model and query language for spatial data in [51]. This paper assumes the RDF-based spatial data representation of [53] and discusses various ways to exploit what is already available in SPARQL to pose queries. The options compared are: (i) to use SPARQL as in [53], (ii) to introduce a new PREMISE clause in SPARQL that could be used to introduce spatial geometries that can be used in a query, and (iii) to use some form of the DESCRIBE query form of SPARQL for asking queries about geometries. [51] chooses the last option as the most appropriate. However, it is not clear to us that using something like DESCRIBE which is not given a formal semantics in the SPARQL W3C specification[31] (the answer to a DESCRIBE query depends on the data source; there is no semantics for this kind of query) is a good way to solve the problem of querying spatial data in the Semantic Web.

Finally, the only native RDF store that provides some kind of spatial sup-

[31] Retrieved March 1, 2009, from `http://www.w3.org/TR/rdf-sparql-query/`

port is AllegroGraph RDFStore[32]. AllegroGraph RDFStore provides a very basic mechanism for the efficient storage and retrieval of geospatial data. It deals with data elements which have coordinates in a two-dimensional region. Support is provided both for Cartesian coordinate systems (i.e., a flat plane) and for spherical coordinate systems (e.g., the surface of the earth or the celestial sphere). Before adding any geospatial data into the store, it is necessary to define the geospatial subtypes the store will use. A geospatial subtype is either spherical or Cartesian, has specific X and Y ranges, and a specific Y strip width. Spatial data and queries are then formed declaratively using AllegroGraph's Prolog facilities (SPARQL currently cannot be used).

2.5.2 Temporal information in RDF

The first works that proposed to introduce temporal features in RDF were by Gutierrez and colleagues [33, 34, 41]. In their proposal, a framework to incorporate valid time in RDF is introduced. Extending the concept of RDF triple, a *temporal triple* is an RDF triple with an additional temporal label (a natural number). For example, $(s, p, o)[t]$ is a temporal triple which denotes the fact that the triple (s, p, o) is valid at time t. Triples valid at time intervals are then defined by sets of triples valid at time points. Finally, a *temporal RDF graph* is defined as a set of temporal RDF triples. [33, 34, 41] study the semantics of the proposed extension to RDF, define appropriate query languages for the extension and present results on the complexity of query answering.

Another related paper presented lately is the work by Pugliese et. al [82]. In contrast to the work by Gutierrez and colleagues [33, 34], this work mainly focuses on the indexing of temporal RDF graphs using the tGRIN index. tGRIN is a specialized index for temporal RDF that is physically stored in an RDBMS. tGRIN can be used to index temporal RDF graphs as defined by Gutierrez et al. in [33, 34] but also some case of indeterminate temporal information. The results of this paper show that the tGRIN index is superior in terms of performance than augmenting well-known RDF stores (e.g., Jena, Sesame) with temporal indexes like R+ trees, SR-trees, ST-index, and MAP21.

Available RDF storage and query systems have only recently started to consider ways to introduce temporal RDF in their implementations. The only native RDF store which currently supports some form of temporal RDF is the AllegroGraph RDFStore[32]. AllegroGraph RDFStore supports the storage and retrieval of temporal data including datetimes, time points, and time intervals. Once data has been encoded, applications can perform queries involving a broad range of temporal constraints on data, including relations between points and datetimes, intervals and datetimes, two points, two intervals, and points and intervals. AllegroGraph RDFStore allows users to query temporal RDF graphs using its Prolog

[32]Retrieved March 1, 2009, from `http://agraph.franz.com/allegrograph/`

querying facilities while support for SPARQL is left for the future.

2.5.3 Spatial data with a temporal dimension in RDF

Recently, interesting research on spatial and temporal information in the Semantic Web has been carried out by Amit Sheth's group [77, 79, 78, 38, 92, 6]. In their more recent work [77], an extension of SPARQL, called SPARQL-ST, is defined that allows one to query spatial and non-spatial data with a time dimension[33]. The main idea of [77] is to incorporate spatial features in the temporal RDF graphs of [33]. These spatial features are modeled with a spatial ontology based on the GeoRSS GML specification [93]. The main new concept of SPARQL-ST is the introduction of two new types of variables namely spatial variables and temporal variables. Spatial variables (denoted by a % prefix) represent complex spatial features rather than a simple URI, and the concept of SPARQL mappings has been extended to map a spatial variable to a set of triples that represent the required spatial information. Similarly, temporal variables (denoted by a # prefix) are mapped to time intervals and can appear in the fourth position of a temporally extended triple in the style of [33]. Furthermore, in SPARQL-ST two special filters are introduced: `SPATIAL FILTER` and `TEMPORAL FILTER`. These filters are used to filter the results with spatial and temporal constraints (e.g., relations from the RCC calculus [15] for the spatial part and Allen's interval calculus [4] for the temporal part). In order to enable the realization of this query language, the used spatial and temporal operators need to be implemented. Both spatial and temporal operators are implemented using Oracle's extensibility framework, while the strictly RDF concepts are implemented using Oracle's RDF storage and inferencing capabilities. For more details on the implementation please see [77, 79].

2.6 Spatial and Temporal Information in Description Logics and OWL

There has also been interesting work on temporal and spatial extensions of description logics. The idea here is to introduce a concrete domain to model time or space together with appropriate concepts, roles and features [66]. Work here has concentrated mostly on issues of semantics and reasoning in these logics. See for example the papers [8, 65] for recent work in this area, and [7] for a somewhat outdated survey of the area. Regarding spatial extensions of description logics,

[33]We avoid using the word "spatiotemporal" for the RDF extension and the language SPARQL-ST presented in [77]. In this respect, we follow [58, 37] and others who prefer to use this term for phenomena that are truly spatiotemporal e.g., moving objects. Query languages for moving objects support constructs for velocity, accelaration, etc. so they are more expressive than SPARQL-ST.

there has been much less work than for temporal extensions. See for example, the papers [66, 50].

Recently, works on temporal ontologies using OWL have started to appear. [40] presents OWL-Time[34], an ontology of temporal concepts designed for representing temporal knowledge on the Web. This ontology provides a vocabulary for expressing knowledge about qualitative relations among instants and intervals, information about durations, and information about dates and times. Another relevant effort in this area is the language TOWL [71] which is an extension of OWL with a concrete interval domain and appropriate classes and properties for the representation of interval knowledge in the Web.

Finally, there have been a lot of research activities recently aiming towards a Semantic Web enriched with geospatial information. For example, the Geospatial Semantic Web [22, 11, 90] is a vision that proposes to combine geospatial information with semantics so that users are enabled to retrieve more precisely the geographic data that they need. Papers in this area have only recently started to appear. In [52], the authors propose several kinds of geospatial ontologies that could be used for the Geospatial Semantic Web, while in a most recent paper [72], a system is presented which provides a spatiotemporal ontology modeling and semantic query environment compatible with OWL-DL.

2.7 Summary

In this chapter, we presented a short survey of related work in the areas most relevant to this deliverable. We started with previous work on registries for architectures related to the service-oriented architecture we will be developing in SemsorGrid4Env. Then, we discussed work on geospatial Web service discovery and, finally, we surveyed the state-of-the art in various data models and query languages for temporal and spatial information.

[34]Retrieved March 1, 2009, from `http://www.w3.org/TR/owl-time/`

Chapter 3

Data Model and Query Language

In this chapter we will present how to extend RDF(S) with the ability to represent spatial and temporal data so that sensor metadata can be represented and queried in an efficient way. We will also present how to extend SPARQL so that spatial and temporal data can be queried using a declarative and user-friendly language.

The most common form of a SPARQL query is:

```
select Variables
where GroupGraphPatterns
```

where *Variables* is a list of the projection variables and *GroupGraphPatterns* is a combination of graph patterns that is matched against an RDF graph. A graph pattern may contain triple patterns, conjunctions and disjunctions of graph patterns and filters that restrict the values of possible matchings. In our proposed extension of SPARQL, we introduce spatial variables to refer to spatial geometries, temporal variables to refer to the valid time of a triple, spatial and temporal filter expressions to restrict the value of possible spatial and temporal matchings, expressions in the `select` clause that define new spatial geometries and quad-patterns that extends triple patterns with a temporal variable that refers to the valid time of a triple.

Following the spirit of constraint databases [84, 61], and especially the work on CSQL [62], we develop a constraint data model, called *stRDF*, that can be used to represent thematic and spatial data that might change over time.[1] stRDF will be the data model for the registry of the SemsorGrid4Env architecture. In our approach, a registry is just a database with thematic, spatial and temporal data. Then, we develop the query language *stSPARQL* to query stRDF databases.

[1] In fact, it is possible to start by extending RDF to the scheme (i.e., the family of data models) *cRDF* where "c" stands for "constraints" as understood in constraint databases [84, 61] and AI constraint satisfaction [88]. Then, the data model *stRDF* will simply be an instance of this scheme. We leave this general development for a future paper so that we do not diverge from the purpose and the intended audience of this deliverable.

A subset of this query language will be offered by the registry following the specifications to be developed in Workpackage WP1.

3.1 The stRDF Data Model

The main contribution of stRDF is to bring to the RDF world the benefits of constraint databases and constraint-based reasoning so that spatial and temporal data can be represented in RDF using constraints. In this way, application areas with a rich spatial and temporal component (such as the Semantic Sensor Internet/Web/Grid) can be tackled using Semantic Web technologies. In the context of SemsorGrid4Env, one challenging use of stRDF is in the registry for sensor metadata.

To develop stRDF, we follow closely the ideas of constraint databases [84, 61] and especially the work on CSQL [62]. First, we define the formulae that we allow as constraints. Then, we develop stRDF in two steps. The first step is to define *sRDF* which extends RDF with the ability to represent spatial data. Then, we extend sRDF to stRDF so that thematic and spatial data with a temporal dimension can be represented.

3.1.1 Linear constraints

Constraints will be expressed in the first-order language $\mathcal{L} = \{\leq, +\} \cup \mathbb{Q}$ over the structure $\mathcal{Q} = \langle \mathbb{Q}, \leq, +, (q)_{q\in\mathbb{Q}} \rangle$ of the linearly ordered set of the rational numbers, denoted by $\mathbb{Q}$, with rational constants and addition[2]. The atomic formulae of this language are *linear equations* and *inequalities* of the form: $\sum_{i=1}^{p} a_i x_i \Theta a_0$, where Θ is a predicate among $=$ or $\leq$, the x_i's denote variables and the a_i's are integer constants. Note that rational constants can always be avoided in linear equations and inequalities. The multiplication symbol is used as an abbreviation i.e., $a_i x_i$ stands for $x_i + \ldots + x_i$ (a_i times).

We now define semi-linear subsets of $\mathbb{Q}^k$, where k is a positive integer.

Definition 1. *Let S be a subset of $\mathbb{Q}^k$. S is called* semi-linear *if there is a quantifier free formula $\phi(x_1, \ldots, x_k)$ of $\mathcal{L}$ where $x_1, \ldots, x_k$ are variables such that $(a_1, \ldots, a_k) \in S$ iff $\phi(a_1, \ldots, a_k)$ is true in the structure $\mathcal{Q}$.*

Spatial objects with k dimensions are represented by formulae that are Boolean combinations of linear constraints over k variables.

We will use $\varnothing$ to denote the empty subset of $\mathbb{Q}^k$ represented by any inconsistent formula of $\mathcal{L}$.

[2]The theoretical development is the same if we consider the set of reals instead of the set of rationals.

3.1.2 The sRDF data model

We now define sRDF. As in theoretical treatments of RDF [75, 76], we assume the existence of pairwise-disjoint countably infinite sets I, B and L that contain IRIs, blank nodes and literals respectively. In sRDF, we also assume the existence of an infinite sequence of sets $C_1, C_2, \ldots$ that are pairwise-disjoint with I,B and L. The elements of each $C_k, k = 1, 2, \ldots$ are the quantifier-free formulae of the first-order language $\mathcal{L}$ with k free variables. We denote with C the infinite union $C_1 \cup C_2 \cup \cdots$.

Definition 2. *An* sRDF triple *is an element of the set*

$$(I \cup B) \times I \times (I \cup B \cup L \cup C).$$

If (s, p, o) is an sRDF triple, s will be called the subject, p the predicate and o the object of the triple. An sRDF graph *is a set of sRDF triples.*

In the above definition, the standard RDF notion of a triple is extended, so that the object of a triple can be a quantifier-free formula with linear constraints. According to Definition 1 such a quantifier-free formula with k free variables is a finite representation of a (possibly infinite) semi-linear subset of $\mathbb{Q}^k$. Semi-linear subsets of $\mathbb{Q}^k$ can capture a great variety of spatial geometries, e.g., k-dimensional unions of convex polygons possibly with holes, thus they give us a lot of expressive power. However, they cannot be used to represent other geometries that need higher-degree polynomials e.g., circles[3].

Example 1. *The following are sRDF triples[4]:*

$$(ex{:}lp1, rdf{:}type, ex{:}LandParcel)$$

$$(ex{:}lp1, ex{:}landUse, \text{“}forest\text{”})$$

$$(ex{:}lp1, ex{:}hasPoints, \text{“}0 \leq x \leq 5 \wedge 1 \leq y \leq 4\text{”})$$

The above triples define a land parcel, its use and its 2-dimensional geometry using a conjunction of linear constraints. The last triple is not a standard RDF triple since its object is an element of set C.

In terms of the W3C specification of RDF, sRDF can be realized as an extension of RDF with a new kind of *typed literals*: quantifier-free formulae with linear constraints. The datatype of these literals is `strdf:SemiLinearPointSet` and can be defined using XML Schema. The sensor ontology to be developed in Workpackage WP4 can also use this datatype to refer to the geometry of various spatial objects.

We now move on to define stRDF.

[3]There has been work on polynomial constraint databases as well, but the evaluation of queries in this case is more difficult[60].

[4]In this and remaining examples of this chapter, we will be using namespaces but we will not define them explicitly.

3.1.3 stRDF

We will now extend sRDF with time. Since [95], database researchers have differentiated among user-defined time, valid time and transaction time. RDF (and therefore sRDF) supports user-defined time since triples are allowed to have as objects literals of the following XML Schema datatypes: `xsd:dateTime`, `xsd:time`, `xsd:date`, `xsd:gYearMonth`, `xsd:gYear`, `xsd:gMonthDay`, `xsd:gDay`, `xsd:gMonth`.

stRDF extends sRDF with the ability to represent the *valid time* of a triple (i.e., the time that the triple was valid in reality) using the approach of Gutierrez et al. [34, 33] where the notion of temporal RDF graphs is introduced. This approach has also been followed in the definition of the data model and language for SPARQL-ST [77]. Currently, our proposal does not support transaction time (but we are ready to support it as soon as the need arises in the project).

We will consider time as a discrete, linearly ordered set of *time points*. A *time interval* $[a, b]$ is an ordered pair of time points such that $a \leq b$. The following definition is essentially from [34].

Definition 3. *An* stRDF quad *is an sRDF triple* (a, b, c) *with a fourth component* t *which is a temporal label (a natural number). We will use the notation* (a, b, c, t)*, where* t *denotes the time point that the triple was valid in the real world. The expression* $(a, b, c, [t_1, t_2])$ *is just syntactic sugar for the set of quads* $\{(a, b, c, t) |\ t_1 \leq t \leq t_2\}$*. An* stRDF graph *is a set of stRDF quads.*

We will now define a declarative query language for querying stRDF graphs.

3.2 The Language stSPARQL

In this section we give the syntax of the query language stSPARQL by means of examples. A more formal definition of the language and its grammar is presented later in Section 3.3.

In our examples, we will use three different datasets. Two of our data sets have been inspired by the use cases of the project: they describe sensors and sensor networks. The third dataset is a classical example from GIS and it is used to illustrate the more advanced spatial features of the language that may turn out not to be of interest in this project but are of interest in the general Geospatial Semantic Web area. All the queries that we present involve spatial data thus we use the following classification of [35] in naming them.

- Spatial selection (e.g., point query, window query).

  The term spatial selection is used to describe a selection based on a spatial predicate. A *point query* asks for spatial features of a dataset that contain a user-specified point location. A *window query* asks for spatial features of

Figure 3.1: Deployment of sensors

a dataset that are enclosed by a user-specified area. A typical example of a window query is "Find all sensors (point locations) inside a user defined window (rectangle)".

- Spatial join.

  A spatial join is a join which compares two spatial objects through a spatial predicate. A spatial predicate can be a combination of topological, metric, or directional relations between two spatial objects. A typical example of a spatial join is "Find all sensors (point locations) and the land parcels (polygonal areas) they are contained in".

- Spatial function application.

  In this type of query, we utilize operations that compute new spatial objects from spatial objects that exist in a dataset. A typical example of a spatial function application is clipping, an example of which is: "Find all sensor networks that have a sensor (point location) inside a user-defined bounding box (rectangle), and return the part of the network links (polyline) lying inside this bounding box".

We also give examples of queries that involve time; these are just temporal selections.

3.2.1 Queries for a sample sensor dataset

Consider a dataset containing information about two sensors located at Southampton and one sensor located in Chichester Bar (this dataset is based on the use case of Workpackage WP7). The sensors located in Southampton measure wind speed and wind direction, while the sensor located in Chichester Bar measures only the wind speed. The sensor located in Chichester Bar was dead from 10am to 11am on 18/01/2009. In Figure 3.1 we can see a map showing the the deployment of these sensors.

The above sensors can be described using the following stRDF triples[5]:

```
sotonmet:windspeed rdf:type sit:Sensor
sotonmet:windspeed sit:sensor_type sit:WindSpeed
sotonmet:windspeed sit:location "lon=50.8839 and lat=-1.3936"

sotonmet:winddir rdf:type sit:Sensor
sotonmet:winddir sit:sensor_type sit:WindDir
sotonmet:winddir sit:location "lon=50.8839 and lat=-1.3936"

chimet:windspeed rdf:type sit:Sensor
chimet:windspeed sit:sensor_type sit:WindSpeed
chimet:windspeed sit:location "lon=50.77556 and lat=-0.94611"

chimet:windspeed sit:state "dead" [2009-01-18T10:00:00+00:00,
                                   2009-01-18T11:00:00+00:00]
```

Example queries

1. *Spatial selection (window query).* Find all sensors inside the rectangle R(50.744072, -1.577019, 50.917442, -0.79875); show their exact positions.

```
 select ?SENS, %POS
where {?SENS rdf:type sit:Sensor .
       ?SENS sit:location %POS .
       s-filter(%POS inside
                      "50.744072 <= lon <= 50.917442 and
                       -1.577019 <= lan <= -0.79875")}
```

   Query result. The result of this query is given below and visualized in Figure 3.2. Visualization of results of spatial queries is an important issue that has been discussed in various spatial database papers e.g., [23, 21]. In our case we envisage visualization of queries to take place in the application services that will use the registry. Therefore, we only give the visualization of query results in this document for presentation purposes.

?SENS	%POS
sotonmet:windspeed	"lon=50.8839 and lat=-1.3936"
sotonmet:winddir	"lon=50.8839 and lat=-1.3936"
chimet:windspeed	"lon=50.77556 and lat=-0.94611"

[5]We use an appropriately modified RDF/N3 notation to describe these triples. Namespace definitions are omitted.

Figure 3.2: Query results

Let us now explain the new features of stSPARQL by referring to the above example. stSPARQL has a new kind of variables: *spatial variables* that are prefixed by the character `%` (in this, we follow the convention of SPARQL-ST [77]). Spatial variables can be used in basic graph patterns (e.g., `?SENS sit:location %POS`) to refer to spatial literals denoting semi-linear point sets. They can also be used in *spatial filters*, a new kind of filter expressions introduced by stSPARQL that is used to constrain spatial variables and their *projections*. Spatial filters are introduced by the keyword `S-FILTER`. Projections of spatial variables (e.g., `%POS[1]`) can also be used; they denote the projections of the corresponding point sets on the appropriate dimensions, and are written using the notation `Variable "[" Dimension1 "," ... "," DimensionN "]"`.

As we will explain in Section 3.3, a spatial filter is an expression that may contain topological spatial constraints between two spatial objects. Egenhofer and Franzosa identify in [24] the following mutually exclusive and pairwise disjoint topological relations: *disjoint, touch, equals, contains, covers, inside, covered by, overlap boundary disjoint, overlap boundary intersect*. These topological relations can be used as operators in a spatial filter expression e.g., `s-filter(%GEO1 inside %GEO2)`. The operator `inside` returns true when the interior and boundary of `%GEO1` is completely contained in the interior of `%GEO2`. The complete set of operators allowed in stSPARQL are given in Section 3.3.

2. *Spatial selection (window query).* Find all sensors that measure wind speed inside the rectangle R(50.744072, -1.577019 , 50.917442, -0.79875); show their exact positions.

```
select ?SENS, %POS
where {?SENS rdf:type sit:Sensor .
       ?SENS sit:sensor_type sit:WindSpeed .
       ?SENS sit:location %POS .
```

Figure 3.3: Query results

```
        s-filter(%POS inside
                        "50.744072 <= lon <= 50.917442 and
                         -1.577019 <= lan <= -0.79875")}
```

Query result. The result of this query is given below and visualized in Figure 3.3.

?SENS	**%POS**
sotonmet:windspeed	"lon=50.8839 and lat=-1.3936"
chimet:windspeed	"lon=50.77556 and lat=-0.94611"

3. *Spatial and temporal selection.* Find all sensors that measure wind speed and are inside the rectangle R(50.744072, -1.577019, 50.917442, -0.79875) and were dead at 10:30 on Sunday the 18th; show their exact positions.

```
select ?SENS, %POS
where {?SENS rdf:type sit:Sensor .
       ?SENS sit:sensor_type sit:WindSpeed .
       ?SENS sit:location %POS .
       ?SENS sit:state "dead" #TIME .
       s-filter(%POS inside
                        "50.744072 <= lon <= 50.917442 and
                         -1.577019 <= lan <= -0.79875") .
       t-filter(#TIME contains 2009-01-18T10:30:00+00:00)}
```

Query result. The result of this query is given below and visualized in Figure 3.4.

Figure 3.4: Query results

?SENS	%POS
chimet:windspeed	"lon=50.77556 and lat=-0.94611"

The above query demonstrates the features of stSPARQL that are used to query the valid times of triples. stSPARQL offers one more new kind of variables in addition to spatial ones: *temporal variables* that are prefixed by the character `#` (in this, we also follow the convention of SPARQL-ST [77]). Temporal variables can be used as the last term in a new kind of basic graph pattern called *quad pattern* to refer to the valid time of a triple. For example, `SENS state dead #TIME` is a quad pattern of the above query and `#TIME` is a temporal variable. Temporal variables can also appear in `T-FILTER` expressions, a new kind of filter that can be used in stSPARQL to constrain the valid time of triples. The syntax of `t-filter` expressions is presented in Section 3.3

3.2.2 Queries for a sample sensor network dataset

Consider now the following dataset containing information about sensors and their networks. Each sensor belongs to a sensor network and has a known position. For some sensor networks, we may have information regarding the links between each sensor described as a polyline.

```
ex:network1 rdf:type ex:SensorNetwork
ex:network1 ex:has_sensor ex:sensor1
ex:network1 ex:has_sensor ex:sensor2

ex:network2 rdf:type ex:SensorNetwork
ex:network2 ex:has_sensor ex:sensor3
ex:network2 ex:has_sensor ex:sensor4
ex:network2 ex:has_links "y - x = 0 and 300 <= x <= 305 and 300 <= y <= 305"
```

```
ex:sensor1 rdf:type ex:Sensor
ex:sensor1 ex:located_at "x=10 and y=10"

ex:sensor2 rdf:type ex:Sensor
ex:sensor2 ex:located_at "x=12 and y=13"

ex:sensor3 rdf:type ex:Sensor
ex:sensor3 ex:located_at "x=300 and y=300"

ex:sensor4 rdf:type ex:Sensor
ex:sensor4 ex:located_at "x=305 and y=305"
```

Example queries

1. *Query with OPTIONAL and spatial function application (clipping).* Find all sensors inside the rectangle $R(11, 11, 303, 303)$; show their exact position and optionally the links that the respective sensor network has inside the given box.

```
select ?SENSOR, %SENS_POS, %NETWORK_LINKS INTER
                          "11 <= x <= 303  and 11 <= y <=303"
where {?NETWORK rdf:type ex:SensorNetwork .
       ?NETWORK ex:has_sensor ?SENSOR .
       ?SENSOR rdf:type ex:Sensor .
       ?SENSOR ex:located_at %SENS_POS .
       s-filter(%SENS_POS inside
                             "11 <= x <= 303 and 11 <= y <= 303") .
       optional {?NETWORK ex:has_links %NETWORK_LINKS .
                 s-filter(%NETWORK_LINKS anyinteract
                          "11 <= x <= 303  and 11 <= y <=303")}}
```

Query result. The result of this query is displayed below.

?SENSOR	%SENS_POS	%RESULT
ex:sensor2	"x=12 and y=12"	
ex:sensor3	"x=300 and y=300"	"x - y = 0 and 300 <= x <= 305 and 300 <= y <= 305 and 11 <= x <= 303 and 11 <= y <= 303"

Optional graph patterns in SPARQL allow information to be added to what has already been collected if this information is available, instead of rejecting solutions if some part of a triple pattern does not match. If an optional graph pattern matches a graph, new bindings are defined and added to one or more solutions. If an optional graph pattern does not match a graph, the solution is left unchanged. Unlike SQL left outer joins, optional graph patterns in SPARQL may be complex graph patterns with many joins, may contain new variables or references to variables previously declared and may contain multiple `filter`, `s-filter` or `t-filter` expressions.

The previous query first finds matching sensors and their location. If there is a triple with predicate `ex:has_links` for a sensor network and the object of that triple satisfies the relevant `s-filter` expression, a solution will contain the intersection of the object of that triple with the window defined in the `select` clause. In this example, only a single triple pattern is given in the `OPTIONAL` part but, in general, the `OPTIONAL` part may contain many joins, spatial or temporal filters etc.

In the `select` clause of an stSPARQL query we allow expressions like `GEO_1 INTER GEO_2` or `GEO_1 UNION GEO_2` where `GEO_1`, `GEO_2` are spatial variables or user-defined spatial objects specified as quantifier-free formulae with linear constraints. These expressions compute new spatial objects e.g., the intersection or the union of two regions respectively. The complete set of operators allowed is given in Section 3.3. In stSPARQL we handle empty sets of spatial values explicitly (as in CSQL[62]) instead of eliminating them implicitly (as in Dedale[31]). Elimination of triples that have an empty set of spatial values must be done explicitly in an `s-filter` expression. In this example, if the last `s-filter` expression was removed, then links that have no intersection with the user-defined window are also returned.

The computation of the intersection of the network links geometry (polyline) with the user-defined window (rectangle) can be done easily : we just conjoin the two formulae with an *and* ($\wedge$). The formula in the result defines all pairs of (x, y) values that satisfy both the constraints defining the geometry of the network links and the user-defined window. The result obtained in this way can contain redundancy (e.g., constraints like `(300<=x<=305 and 11<=x<=303)` which reduces to `(300<=x<=303)`. Consequently, *simplification* must be carried out at some point in the query evaluation process in order to eliminate redundancies. This simplification operation, applied to the initial result given above, would yield the following table:

?SENSOR	%SENS_POS	%RESULT
ex:sensor2	"x=12 and y=12"	
ex:sensor3	"x=300 and y=300"	"x - y = 0 and 300 <= x <= 303 and 300 <= y <= 303"

2. *Query with OPTIONAL and spatial selection (windowing).* Find all sensors inside the rectangle $R(11, 11, 303, 303)$; show their exact position and optionally the links of the sensor network.

```
select ?SENSOR, %SENS_POS, %NETWORK_LINKS
where {?NETWORK rdf:type ex:SensorNetwork .
       ?NETWORK ex:has_sensor ?SENSOR .
       ?SENSOR rdf:type ex:Sensor .
       ?SENSOR ex:located_at %SENS_POS .
       s-filter(%SENS_POS inside
                        "11 <= x <= 303 and 11 <= y <= 303") .
       optional {?NETWORK ex:has_links %NETWORK_LINKS .
                 s-filter (%NETWORK_LINKS anyinteract
                        "11 <= x <= 303 and 11 <= y <=303")}}
```

Query result. The result of this query is given below.

?SENSOR	%SENS_POS	%NETWORK_LINKS
ex:sensor2	"x=12 and y=12"	
ex:sensor3	"x=300 and y=300"	"y - x = 0 and 300 <= x <= 305 and 300 <= y <= 305"

The previous query demonstrates a more complex spatial filter. The query first finds sensors that belong to a sensor network and are located inside the user-defined window. The optional part of the query asks for the geometries of the sensor networks' links. If we have the information regarding the links of a sensor network and there is some spatial interaction of the links with the user-defined window, we add to the solution the full description of the sensor network's links.

As we will explain in Section 3.3, a spatial filter is an expression that may contain topological spatial constraints between two spatial objects. Additionally to the operators identified by Egenhofer and Franzosa in [24], we

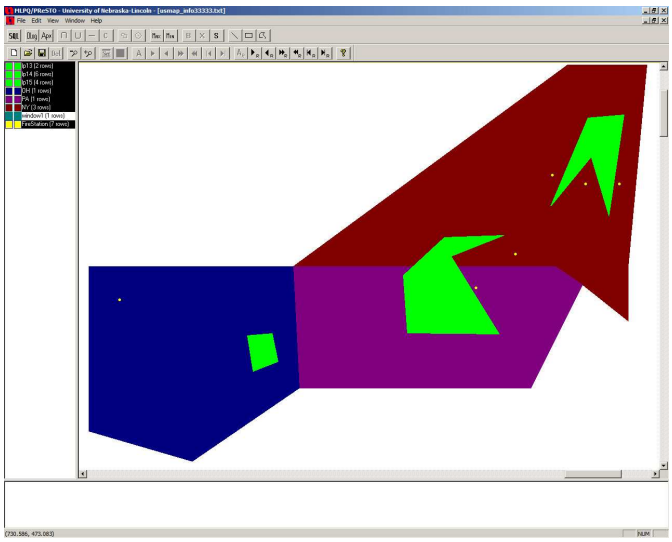

Figure 3.5: Land parcels, states and fire stations

also allow the operator `anyinteract` that returns true when two geometries are not disjoint. In this example we wanted to check whether two spatial objects had any spatial interaction, so we defined a spatial filter where geometries of network links that are disjoint with the user-defined window are not included in the query result.

The difference with the previous query is that we now ask for the complete geometry of the sensor network's links if there is any spatial interaction with user-defined query while in the previous query we computed the intersection of the geometry of the sensor network's links with the user-defined window.

3.2.3 A typical example from GIS

Let us now consider a dataset from a typical GIS application that describes land parcels that represent the ground occupancy or the land use of a certain geographic region, such as cropland, pasture or forest. These land parcels may reside inside a state or may intersect several states. Fire stations, where firefighting apparatus (i.e, fire engines, fire extinguishers, and other fire extinguishing equipment) is stored, are located near the forests. In Figure 3.5 we can see the states of Ohio, Pennsylvania and New York that are visualized with a blue, purple and maroon polygon respectively, some green polygons that represent the forests in the area and some yellow points that represent the fire stations.

The stRDF description of the states that have a geometry that is expressed using linear constraints is the following:

```
ex:s1 rdf:type ex:State
ex:s1 ex:has_name "New York"
```

```
ex:s1 ex:has_geometry
      "(-66x+90y<-8748 and 26y<12974 and -66x+6y > -52380
        and 110y>47630) or
       (24y < 10392 and -6x+15y > 1497 and 6x+9y > 8751) or
       (-6x+15y <1497 and 18x < 14994 and 12x+15y > 16221)"

ex:s2 rdf:type ex:State
ex:s2 ex:has_name "Pennsylvania"
ex:s2 ex:has_geometry
      "86y < 37238 and 6x+9y < 8751 and -34x+17y > -20553
       and 76y > 29868 and 40x+2y > 29786"

ex:s3 rdf:type ex:State
ex:s3 ex:has_name "Ohio"
ex:s3 ex:has_geometry
      "67y < 29011 and 40x+2y < 29786 and -24x+35y > -3645
       and 10x+34y > 19446 and 54x > 35424"
```

The stRDF description of land parcels that have a geometry that is expressed using linear constraints and have a specific land use is the following:

```
ex:lp13 rdf:type ex:LandParcel
ex:lp13 ex:land_use "forest"
ex:lp13 ex:has_geometry
        "(x+0.200002y<=798.372742 and x-2.499986y<=-286.348328
         and -x+0.499998y<=-510.679077) or
         (x-0.499998y<=510.679077 and -x-0.156864y<=-772.448792
         and -x+11.666931y<=4078.833252)"

ex:lp14 rdf:type ex:LandParcel
ex:lp14 ex:land_use "forest"
ex:lp14 ex:has_geometry
        "(-x+11.600311y<=4767.707031 and -y<=-481.657867
           and x-11.400317y<=-4671.174805) or
         (-x+11.400317y<=4671.174805 and -x+0.999850y<=-338.292084
           and x-9.666733y<=-3834.414063) or
         (-x+9.666733y<=3834.414063 and -x+0.428571y<=-613.336975
           and x-0.808219y<=441.383148) or
         (-x+0.808219y<=-441.383148 and -y<=-452.929535
           and x-0.801370y<=444.688965) or
         (-x+0.801370y<=-444.688965 and x-0.833331y<=430.212616
           and x-0.764708y<=462.384857) or
         (-x+0.764708y<=-462.384857 and -x-0.294739y<=-959.076416
```

```
           and x-0.147238y<=760.422974)"

ex:lp15 rdf:type ex:LandParcel
ex:lp15 ex:land_use "forest"
ex:lp15 ex:has_geometry
        "(-x+1.085713y<=-292.161499 and -x+27.500151y<=11395.946289
           and x-2.513515y<=-321.850525) or
         ( x-2.513515y<=-321.850525 and -x+2.941169y<=511.387146
          and -x+2.150007y<=165.527252) or
         (-x+2.150007y<=165.527252 and x+0.621624y<=1046.101440
          and -x-1.648150y<=-1467.831543) or
         ( x+1.648150y<=1467.831543 and -x-85.004036y<=-35713.109375
          and -x-0.075471y<=-791.517029)"

ex:lp16 rdf:type ex:LandParcel
ex:lp16 ex:land_use "forest"

ex:lp17 rdf:type ex:LandParcel
```

The stRDF description of the fire stations that have a geometry that is expressed using linear constraints is the following:

```
ex:fs1 rdf:type ex:FireStation
ex:fs1 ex:has_location "x=796 and y=437"

ex:fs2 rdf:type ex:FireStation
ex:fs2 ex:has_location "x=818 and y=425"

ex:fs3 rdf:type ex:FireStation
ex:fs3 ex:has_location "x=783 and y=426"

ex:fs4 rdf:type ex:FireStation
ex:fs4 ex:has_location "x=819 and y=460"

ex:fs5 rdf:type ex:FireStation
ex:fs5 ex:has_location "x=808 and y=463"

ex:fs6 rdf:type ex:FireStation
ex:fs6 ex:has_location "x=830 and y=460"

ex:fs7 rdf:type ex:FireStation
ex:fs7 ex:has_location "x=666 and y=422"
```

The land parcels, the states and the fire stations described above have been visualized in Figure 3.5 using the MLPQ/PReSTO System[6], a constraint database system developed at the University of Nebraska-Lincoln by Peter Revesz and his group. We used the graphical user interface of the MLPQ/PReSTO system to sketch the land parcels and get the constraints that represent the land parcels' geometries. We also used the *Intersection* operator of the MLPQ/PReSTO system, to calculate the intersection of two spatial objects and get the constraints that represent the resulting spatial object.

Example queries

1. *Spatial selection (point query).* Which land parcel contains the point (710,400)?

```
select ?LP, ?USE
where {?LP rdf:type ex:LandParcel .
       ?LP ex:has_use ?USE .
       ?LP ex:has_geometry %GEO .
       s-filter(%GEO contains "x=710 and y=400")}
```

Query result. The result of this query is given below.

?LP	?USE
ex:lp13	"forest"

2. *Query with OPTIONAL.* Find all land parcels, their use and optionally their geometry.

```
select ?LP, ?USE, %GEO
where {?LP rdf:type ex:LandParcel .
       ?LP ex:has_use ?USE .
       optional {?LP ex:has_geometry %GEO}}
```

Query result. The result of this query is given below.

[6]Retrieved March 1, 2009, from `http://cse.unl.edu/~revesz/MLPQ/mlpq.htm`

?LP	?USE	%GEO
ex:lp13	"forest"	"(x+0.200002y<=798.372742 and x-2.499986y<=-286.348328 and -x+0.499998y<=-510.679077) or (x-0.499998y<=510.679077 and -x-0.156864y<=-772.448792 and -x+11.666931y<=4078.833252)"
ex:lp16	"forest"	
ex:lp15	"forest"	"(-x+1.085713y<=-292.161499 and -x+27.500151y<=11395.946289 and x-2.513515y<=-321.850525) or (x-2.513515y<=-321.850525 and -x+2.941169y<=511.387146 and -x+2.150007y<=165.527252) or (-x+2.150007y<=165.527252 and x+0.621624y<=1046.101440 and -x-1.648150y<=-1467.831543) or (x+1.648150y<=1467.831543 and -x-85.004036y<=-35713.109375 and -x-0.075471y<=-791.517029)"
ex:lp14	"forest"	"(-x+11.600311y<=4767.707031 and -y<=-481.657867 and x-11.400317y<=-4671.174805) or (-x+11.400317y<=4671.174805 and -x+0.999850y<=-338.292084 and x-9.666733y<=-3834.414063) or (-x+9.666733y<=3834.414063 and -x+0.428571y<=-613.336975 and x-0.808219y<=441.383148) or (-x+0.808219y<=-441.383148 and -y<=-452.929535 and x-0.801370y<=444.688965) or (-x+0.801370y<=-444.688965 and x-0.833331y<=430.212616 and x-0.764708y<=462.384857) or (-x+0.764708y<=-462.384857 and -x-0.294739y<=-959.076416 and x-0.147238y<=760.422974)"

Note that the land parcel `ex:lp16` is a solution even though it does not have a geometry defined, while land parcel `ex:lp17` is not a solution since

it does not have a specific land use.

3. *Query with intersection of areas.* Find all land parcels that are forests and intersect the state of New York; compute this intersection.

```
select ?LP, %GEO1 INTER %GEO2
where {?LP rdf:type ex:LandParcel .
       ?LP ex:has_use "forest" .
       ?LP ex:has_geometry %GEO1 .
       ?S rdf:type ex:State .
       ?S ex:has_name "New York".
       ?S ex:has_geometry %GEO2 .
       s-filter(%GEO1 anyinteract %GEO2) }
```

Query result. The result of this query is displayed below and visualized in Figure 3.6.

?LP	**%GEO**
ex:lp14	(-x+11.600311y<=4767.707031 and -y<=-481.657867 and x-11.400317y<=-4671.174805) or (-x+11.400317y<=4671.174805 and -x+0.999850y<=-338.292084 and x-9.666733y<=-3834.414063) or (-x+9.666733y<=3834.414063 and -x+0.428571y<=-613.336975 and x-0.808219y<=441.383148) or (-x+0.808219y<=-441.383148 and -y<=-452.929535 and x-0.801370y<=444.688965) or (-x+0.801370y<=-444.688965 and x-0.833331y<=430.212616 and x-0.764708y<=462.384857) or (-x+0.764708y<=-462.384857 and -x-0.294739y<=-959.076416 and x-0.147238y<=760.422974)
ex:lp15	(-x+1.085713y<=-292.161499 and -x+27.500151y<=11395.946289 and x-2.513515y<=-321.850525 and -x+1.363636y<-132.545456 and y<499 and x-0.090909y<793.636353 and-y<-433) or (x-2.513515y<=-321.850525 and -x+2.941169y<=511.387146 and -x+2.150007y<=165.527252 and -x+1.363636y<-132.545456 and y<499 and x-0.090909y<793.636353 and -y<-433) or (-x+2.150007y<=165.527252 and x+0.621624y<=1046.101440 and -x-1.648150y<=-1467.831543 and -x+1.363636y<-132.545456 and y<499 and x-0.090909y<793.636353 and -y<-433)

In this example the whole of land parcel `ex:lp14` is an answer to the query since it resides inside the state of New York, while for the land parcel

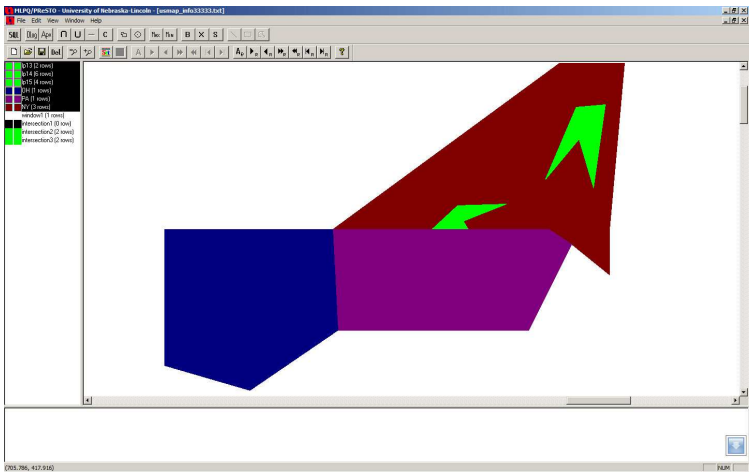

Figure 3.6: Query results

`ex:lp15` only the part that intersects the New York state's geometry is an answer to the query.

4. *Spatial function application (e.g., clipping).* Find the URIs and parts of all land parcels that are forests and intersect the rectangle $R(710, 405, 770, 425)$; compute this intersection.

```
select ?LP, %GEO INTER
                    "710 <= x <= 770 and 405 <= y <= 425"
where {?LP rdf:type ex:LandParcel .
       ?LP ex:has_use "forest" .
       ?LP ex:has_geometry %GEO .
       s-filter(%GEO anyinteract
                    "710 <= x <= 770 and 405 <= y <= 425")}
```

Query result. In Figure 3.7 we can see the rectangle $R(710, 405, 770, 425)$ that is defined by the user. The result of this query is displayed below and visualized in Figure 3.8 where we can see the new spatial objects that are created from the intersection of the land parcels' geometries with the user-defined rectangle.

?LP	%GEO
ex:lp13	(x+0.200002y<=798.372742 and x-2.499986y<=-286.348328 and -x+0.499998y<=-510.679077 and -x<-710 and x<770 and -y<-405 and y<425) or (x-0.499998y<=510.679077 and -x-0.156864y<=-772.448792 and -x+11.666931y<=4078.833252 and -x<-710 and x<770 and -y<-405 and y<425)
ex:lp15	(-x<-710 and x<770 and -y<-405 and y<425 and -x+2.150007y<=165.527252 and x+0.621624y<=1046.101440 and -x-1.648150y<=-1467.831543) or (-x<-710 and x<770 and -y<-405 and y<425 and x+1.648150y<=1467.831543 and -x-85.004036y<=-35713.109375 and -x-0.075471y<=-791.517029)

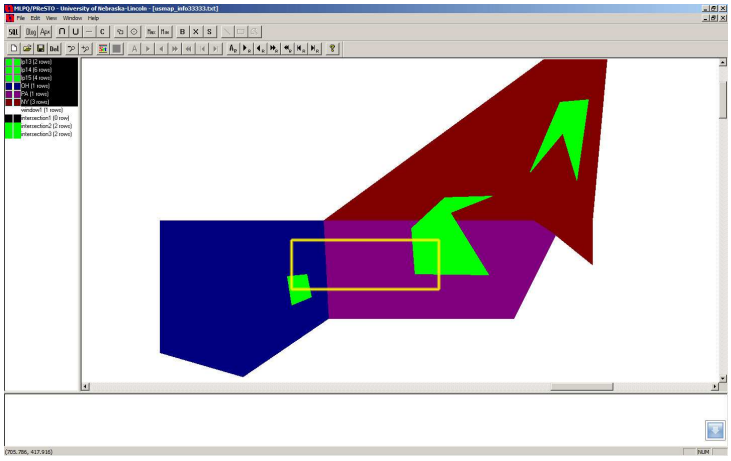

Figure 3.7: User-defined window for a clipping query

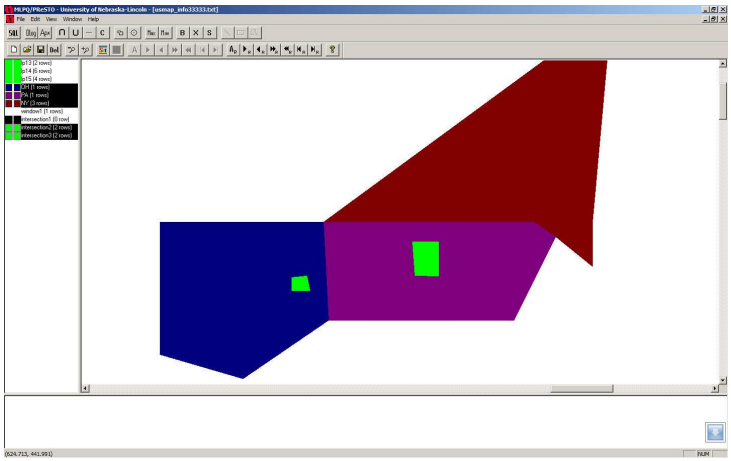

Figure 3.8: Clipping query results

5. *Query with OPTIONAL and spatial function application (e.g., clipping).* Find the URIs and optionally the parts of all land parcels that are forests and optionally have a geometry and this geometry is contained in the rectangle $R(710, 405, 770, 425)$.

```
select ?LP, %GEO INTER "710 <= x <= 770 and 405 <= y <= 425"
where {?LP rdf:type ex:LandParcel .
       ?LP ex:has_use "forest" .
       optional {?LP ex:has_geometry %GEO .
       s-filter (%GEO anyinteract
                      "710 <= x <= 770 and 405 <= y <= 425")}}
```

Query result. The result of the above query is displayed below.

?LP	**%GEO**
ex:lp13	(x+0.200002y<=798.372742 and x-2.499986y<=-286.348328 and -x+0.499998y<=-510.679077 and -x<-710 and x<770 and -y<-405 and y<425) or (x-0.499998y<=510.679077 and -x-0.156864y<=-772.448792 and -x+11.666931y<=4078.833252 and -x<-710 and x<770 and -y<-405 and y<425)
ex:lp15	(-x<-710 and x<770 and -y<-405 and y<425 and -x+2.150007y<=165.527252 and x+0.621624y<=1046.101440 and -x-1.648150y<=-1467.831543) or (-x<-710 and x<770 and -y<-405 and y<425 and x+1.648150y<=1467.831543 and -x-85.004036y<=-35713.109375 and -x-0.075471y<=-791.517029)
ex:lp16	

We can see that the land parcel `ex:lp16` that does not have a defined geometry is included in the results since the existence of a geometry description is optional in this query, but the land parcel `ex:lp14` that has a defined geometry is not included in the results since it does not intersect in the user-specified rectangle.

6. *Spatial function application (e.g., AREA).* Find the area covered by each land parcel that is contained in the rectangle $R(710, 405, 770, 425)$.

```
select ?LP, AREA(%GEO INTER
                "710 <= x <= 770 and 405 <= y <= 425")
where {?LP rdf:type ex:LandParcel .
       ?LP ex:has_geometry %GEO .
       s-filter(%GEO anyinteract
                "710 <= x <= 770 and 405 <= y <= 425")}
```

Query result. The result of this query is displayed below, where we can see the URIs of the land parcels contained by the window defined in the query, and the area of each land parcel.

?LP	**AREA**
ex:lp13	88.625366
ex:lp15	644.926392

7. *Spatial analysis and topological operations (e.g., BUFFER).* In case of a fire emergency in a forest, near-by fire stations should be alerted. Find the URIs of the fire stations that are located within 100km from each forest.

```
select ?LP, ?FS
where {?LP rdf:type ex:LandParcel .
       ?LP ex:land_use "forest" .
       ?LP ex:has_geometry %GEO .
       ?FS rdf:type ex:FireStation .
       ?FS ex:has_location %FS_LOC .
       s-filter(BUFFER(%GEO, 100) contains %FS_LOC)}
```

Query result. The result of this query is displayed below and visualized in Figure 3.9.

?LP	**?FS**
ex:lp15	ex:fs1
ex:lp15	ex:fs3
ex:lp14	ex:fs2
ex:lp14	ex:fs4
ex:lp14	ex:fs5
ex:lp14	ex:fs6

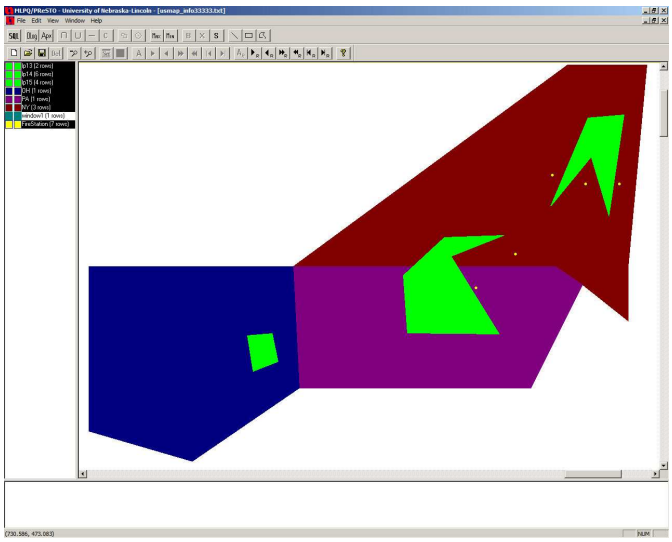

Figure 3.9: Query results

This query uses the *Buffer* spatial operation which involves a spatial object and a distance. The result of this spatial operation is a new geometry around the initial geometry. In this query all fire stations that are located 100km (or less) away from a forest will be selected.

3.3 Syntax of stSPARQL

The syntax of stSPARQL is given below in BNF notation by extending the syntax of SPARQL as given in [5]:

```
stSPARQLquery  ::= SelectClause WhereClause
SelectClause   ::= "select" ((Var | SpatialGeometryExpr | TemporalExpr)+ | "*")
WhereClause    ::= "where" GroupGP
GroupGP        ::= "{" BasicGP?  ((GPNotTriples | Filter) "."? BasicGP?)* "}"
BasicGP        ::= TriplePattern | QuadPattern
QuadPattern    ::= TriplePattern VarT
GPNotTriples   ::= OptionalGP | GroupOrUnionGP | GraphGP
OptionalGP     ::= "optional" GroupGP
GraphGP        ::= "graph" VarOrIRIref GroupGP
GroupOrUnionGP ::= GroupGP ( "union" GroupGP )*
Filter         ::= StandardFilter | SpatialFilter | TemporalFilter
StandardFilter ::= "filter" "(" StandardConstraint ")"
SpatialFilter  ::= "s-filter" "(" SpatialConstraint ")"
TemporalFilter ::= "t-filter" "(" TemporalConstraint ")"
```

We now discuss in more detail some stSPARQL constructs introduced by the above syntax.

3.3.1 Syntax for query variables

Query variables in stSPARQL are introduced by the non-terminal `Var` in the above grammar. Variables appearing in an stSPARQL query have global scope. Non-spatial variables are prefixed with `?` or `$` as in SPARQL. The `?` or `$` is not part of the variable name. Spatial variables are prefixed by the character `%` and temporal variables are prefixed by the character `#` (in this, we follow the convention of SPARQL-ST [77]). Spatial variables can be used in graph patterns or spatial constraints to refer to *spatial literals* denoting semi-linear point sets. Temporal variables can be used in graph patterns or temporal constraints to refer to the valid time of a triple. The possible cases for query variables are given in the following grammar rules:

```
Var             ::= VarN | VarS | VarT
VarN            ::= "?" Varname | "$" Varname
VarS            ::= "%" Varname
VasT            ::= "#" Varname
```

3.3.2 Syntax for triple patterns and quad patterns

Triple patterns in SPARQL are written as a sequence of three elements: subject, predicate and object. stSPARQL allows spatial variables or literals in the object of a triple pattern.

stSPARQL introduces a new kind of basic graph pattern called *quad pattern*, denoted by the non-terminal `QuadPattern` in the stSPARQL grammar given earlier. Temporal variables can be used as the last term in a quad pattern to refer to the valid time of a triple. The syntax for quad patterns is as follows:

```
QuadPattern     ::= TriplePattern VarT
```

3.3.3 Syntax for expressions that represent spatial geometries

In stSPARQL we allow expressions like `GEO1 OP GEO2` where `GEO1`, `GEO2` are spatial variables, applications of spatial functions or user-defined spatial geometries specified as quantifier-free formulae with linear constraints. These expressions may compute new spatial geometries e.g., `GEO1 INTER GEO2` computes the intersection of `GEO1` and `GEO2`. New spatial geometries may also be produced from spatial functions like `BB(GEO)` that construct a new spatial geometry that represent the minimum bounding box of the existing spatial geometry `GEO`. The

possible expressions that represent spatial geometries are given in the following grammar rules:

```
SpatialGeometryExpr ::= SpatialGeometry | SpatialGeometryProj
SpatialGeometry     ::= VarS | ConstructSpatialGeo
                        | SemiLinearPointSetExpr
ConstructSpatialGeo ::= SpatialGeometry SpatialOperator SpatialGeometry
                        | ConstructFunction
SpatialOperator     ::= "INTER" | "UNION" | "MINUS"
ConstructFunction   ::= "BUFFER" "(" SpatialGeometry "," Integer ")"
                        | "BB" "(" SpatialGeometry ")"  /* Bounding Box */
                        | "BD" "(" SpatialGeometry ")"  /* Boundary */
```

3.3.4 Syntax for projection operators

A projection operator projects out one or more (but not all) dimensions of a spatial geometry. The projection of a spatial variable (e.g., `%POS[1]`) denotes the projection of the corresponding point sets on the appropriate dimensions. The syntax of a projection operator is given by the following grammar rules:

```
SpatialGeometryProj ::= SpatialGeometry "[" Dimension ("," Dimension)* "]"
Dimension           ::= Integer
```

3.3.5 Syntax for spatial literals

In stSPARQL, semi-linear point sets are denoted by spatial literals. The syntax for spatial literals is as follows:

```
SemiLinearPointSetExpr ::= '"' Formula '"'
Formula                ::= Formula "and" Formula
Formula                ::= Formula "or" Formula
Formula                ::= "(" Formula ")"
Formula                ::= LinearConstraint
LinearConstraint       ::= LCTerm (("+" | "-") LCTerm)*
                           ("=" | "<=" | "<" | ">" | ">=")
                           Integer
LCTerm                 ::= Integer Varname
```

3.3.6 Syntax for spatial filters

Spatial filters are introduced by the non-terminal `s-filter` in the grammar for stSPARQL given earlier. The expressions that can appear in an `s-filter` clause can be used to constrain the spatial variables that appear in a query. We allow

any of the mutually exclusive and pairwise disjoint topological relations identified by Egenhofer and Franzosa in [24] to be used as a constraint in a `s-filter`. These relations relations are the following: *disjoint, touch, equals, contains, covers, inside, covered by, overlap boundary disjoint, overlap boundary intersect.* Additionally, we allow the operator `ANYINTERACT` that returns true when two geometries are not disjoint and the operator `OVERLAP` that returns true when two geometries have either the *overlap boundary disjoint* or the *overlap boundary intersect* topological relationship. These operators are offered to ease the task of writing user queries; they do not offer any new expressive power. The spatial constraints that can appear in an `s-filter` clause are defined by the following grammar rules:

```
SpatialFilter       ::= "s-filter" "(" SpatialConstraint ")"
SpatialConstraint   ::= SpatialConstraint || SpatialConstraint
SpatialConstraint   ::= SpatialConstraint && SpatialConstraint
SpatialConstraint   ::= "NOT" SpatialConstraint
SpatialConstraint   ::= "(" SpatialConstraint ")"
SpatialConstraint   ::= SpatialTopoFunction | MetricSpatialExpr
SpatialTopoFunction ::= SpatialGeoExpr TopoOperator SpatialGeoExpr
TopoOperator        ::= "ANYINTERACT" | "DISJOINT" | "TOUCH"
                        | "EQUALS" | "CONTAINS" | "COVERS"
                        | "INSIDE" | "COVEREDBY" | "OVERLAPBDDISJOINT"
                        | "OVERLAPBDINTER" | "OVERLAP"
MetricSpatialExpr   ::= "DISTANCE" "(" SpatialGeoExpr ","
                            SpatialGeoExpr ")" Comp Integer
Comp                ::= "=" | "<" | "<=" | ">" | ">="
```

3.3.7 Syntax for temporal filters

Temporal filters are introduced by the non-terminal `t-filter` in the grammar for stSPARQL given earlier. The expressions that can appear in an `t-filter` clause can be used to constrain the temporal variables that appear in a query. A temporal constraint can be a Boolean combination of atomic temporal constraints among temporal variables or constants. We allow any of the thirteen interval relations identified by Allen in [4] to be used as an atomic temporal constraint e.g., `(#t1 OVERLAPS #t2)` returns true when the two time intervals overlap. We also allow *metric temporal constraints* i.e., expressions that compare the result of a *metric temporal function* application with a constant. As in SPARQL-ST[77], we allow only one binary metric temporal function: `ELAPSEDTIME`. The function `ELAPSEDTIME` returns the duration of the time interval between the ending time of the earliest interval and the starting time of the latest interval. If the two intervals do not overlap, the function `ELAPSEDTIME` returns zero. An atomic temporal expression can be either a temporal variable, an interval or an expres-

sion like `TEMPOP(#t1, #t2, #t3)` where `TEMPOP` is either `INTERSECT` or `RANGE`. Expressions like `INTERSECT(#t1, #t2, #t3)` compute the largest interval that intersects the intervals `#t1`, `#t2` and `#t3`, and expressions like `RANGE(#t1, #t2)` compute the smallest interval that contains the intervals `#t1` and `#t2`". The possible temporal constraints allowed in `t-filter` are given by the following grammar rules:

```
TemporalFilter          ::= "t-filter" "(" TemporalConstraint ")"
TemporalConstraint      ::= TemporalConstraint || TemporalConstraint
TemporalConstraint      ::= TemporalConstraint && TemporalConstraint
TemporalConstraint      ::= "NOT" TemporalConstraint
TemporalConstraint      ::= "(" TemporalConstraint ")"
TemporalConstraint      ::= BooleanTemporalFunction | MetricTemporalExpr
BooleanTemporalFunction ::= AtomicTemporalExpr TemporalOperator
                            AtomicTemporalExpr
TemporalOperator        ::= "OVERLAPS" | "BEFORE" | "AFTER" | "EQUAL"
                            | "MEETS" | "DURING" | "STARTS" | "FINISHES"
MetricTemporalExpr      ::= "ELAPSEDTIME" "(" AtomicTemporalExpr ","
                            AtomicTemporalExpr ")" Comp Integer
AtomicTemporalExpr      ::= VarT | Interval | TemporalExpr
TemporalExpr            ::=("INTERSECT" | "RANGE") "(" VarT ("," VarT)* ")"
Comp                    ::= "=" | "<" | "<=" | ">" | ">="
```

We close this section by pointing out that the list of spatial and temporal constraints introduced above will be evaluated carefully in the next few months as the formalization of stSPARQL progresses together with its implementation and use in the project. We expect that any present omission in terms of user-needed functionality will be dealt with out causing any problems to the overall data model and query language framework.

3.4 Comparison with Related Work

Let us now compare stRDF and stSPARQL with relevant proposals in the literature.

As we have already said, our proposal follows closely the constraint databases approach to spatial and temporal databases [84, 61, 85] and in particular the CSQL language [62]. We summarize the benefits of this approach and contrast it with the approach based on data types as, for example, discussed in Ralf Güting survey [35], current standards such as SQL/MM [69] and "OpenGIS Implementation Specification for Geographic information - Simple feature access - Part 2: SQL option"[7], and commercial DBMS (e.g., Oracle Spatial). As discussed in

[7]Retrieved March 1, 2009, from `http://www.opengeospatial.org/standards/sfs`

detail in [86], the constraint database approach has the following benefits when compared with the data type approach:

- The constraint database approach is *very general* and its modeling primitives (semi-linear point sets) do not depend on the application to be supported. On the contrary, approaches based on data types rely on application-dependent choices for data types and related operations (e.g., points and rectangles for a simple sensor discovery application, points, polylines and regions for a GIS application etc.). Notice, for example, that the SQL geometry type hierarchy of the OpenGIS standard for SQL is somewhat different than the corresponding hierarchy in SQL/MM. Similarly, each commercial implementation of these two standards typically offers some new spatial data types that the system developers believe to be useful in certain application areas. Chapter 3 of the book [85] discusses in detail the difficulties involved in designing a comprehensive spatial data type system that is general enough (so that it can be useful to many applications), and at the same time have good properties such as closure, consistency etc.

- The elaboration of a query in constraint-based languages such as CSQL [62] or the query language of Dedale [86] does not depend on the data type of the spatial object. In other words, the user does not need to think about the spatial data types of the inputs, the operators available for each data type or whether operators can be composed in order to formulate a query. Also, the user does not need to think about the spatial data types of the result; the spatial objects returned by a query will always be semi-linear point sets.

The various manifestations of the linear constraint database approach to spatial databases [48, 49, 86, 62] can also be seen to have the following nice properties:

- *Finite representability.* The relations manipulated are finitely representable by quantifier-free formulae in the language $\mathcal{L}$.

- *Closure.* The result of any query is a finitely representable relation.

- *Computability.* One representation of the result of a query is computable from any given representation of the inputs to the query (there can be many representations of the inputs).

- *Tractability vs. Expressiveness.* The data complexity of evaluating queries remains low while the expressive power of the language remains at the levels we are used with in relational systems.

- *Graceful extension to deal with incomplete information.* As demonstrated in the work of Koubarakis [54, 56, 55, 57], constraint databases can naturally be extended to represent incomplete information. Although there is a

computational complexity price to pay here (due to the combinatorics of incomplete information), the extended framework is very natural and easy to use, and the properties of finite representability, closure and computability are still valid.

We believe that the same nice properties hold for stSPARQL but more work is needed in order to demonstrate these properties theoretically (this is currently the focus of our theoretical work on the foundations of stSPARQL).

Let us now compare stSPARQL with relevant spatial and temporal extensions to SPARQL that are in the literature. The closest language to stSPARQL is SPARQL-ST presented in Perry's Ph.D. thesis [77]. stSPARQL follows SPARQL-ST in adopting the model of temporal RDF graphs of [33] to represent the valid time of a triple. However, the spatial part of stSPARQL and SPARQL-ST are significantly different. SPARQL-ST assumes a particular upper ontology expressed in RDFS for modeling theme, space and time [78, 77]. The spatial part of this upper ontology uses the class `geo:SpatialRegion` and its subclasses (e.g., `geo:Polygon`) defined in GeoRSS in order to model spatial geometries (e.g., polygons). Thematic data (e.g., a city) can then be connected to their spatial geometry (e.g., a polygon) using the property `stt:located_at`. Spatial geometries in SPARQL-ST are specified by sets of RDF triples that give various details of the geometry depending on its type (e.g., for a 2-dimensional polygonal area, they give the coordinates of its boundary and the relevant coordinate reference system). SPARQL-ST provides a set of built-in spatial conditions (the topological relationships of RCC and a distance metric) that can be used in SPATIAL FILTER clauses to constrain the geometries that are returned as answers to queries.

Although a semantics for SPARQL-ST is sketched in [77], the treatment of spatial conditions in these semantics leaves much to be desired. The notion of "when a spatial condition evaluates to true" that is used to give semantics to built-in spatial conditions (page 99 of [77]) is not defined formally but is left to the intuition of the reader. When this definition is given explicitly, it will have to rely on the different types of geometries (e.g., `geo:Polygon`) allowed by the spatial ontology of [77], properties of these geometries (e.g., `geo:lrPosList`) and relevant co-ordinate systems (e.g., `geo:CRS_NAD83`). Currently, these semantics are hardwired in the implementation of SPARQL-ST presented in [77]. This means that if someone wants to use a different spatial ontology (e.g., an ontology based on the Open GIS SQL geometry types), this cannot be done unless the semantics of SPARQL-ST and its implementation are modified appropriately.

Since geometries in stRDF and stSPARQL are based on the mathematical concept of semi-linear subsets of $\mathbb{Q}^k$, stSPARQL (as opposed to SPARQL-ST) can be given an elegant semantics based on well-understood mathematical machinery from constraint databases.[8] In addition, as we have shown with exam-

[8]This formal semantics of stSPARQL is currently the focus of our work.

ples in Section 3.2, the new datatype `strdf:SemiLinearPointSet` of stRDF can be used together with spatial ontologies expressed in RDFS to give the same kind of class-based modeling capabilities offered by SPARQL-ST. Thus, stRDF and stSPARQL impose very minimal requirements to Semantic Web developers that want to use our approach: all they have to do is utilize the new datatype `strdf:SemiLinearPointSet`.

Two other papers related to our work on stSPARQL are [53, 51] by Kolas and colleagues. Compared with our work on stRDF and stSPARQL, the system SPAUK presented in [53] (and discussed in detail in Section 2.5.1 of this deliverable) has problems similar to the ones we pointed out for SPARQL-ST. First, no semantics for query evaluation are given. Secondly, even if these semantics are given in great detail, they will rely on the spatial ontologies assumed by SPAUK. Thus, any query processor that implements these semantics will need to be extended if users of the system decide to use different spatial ontologies (this is said explicitly in [53]).

[51] is an interesting paper since it discusses various ways to use SPARQL to query spatial data in a Geospatial Semantic Web context. We agree with [51] that using spatial extension functions in SPARQL (e.g., distance as defined by some IRI) is not an acceptable solution since it is ad-hoc and will create interoperability problems. We also agree with [51] that it is hard to write queries in SPARQL that find information about geometries associated with a spatial feature (e.g., a landmark) when we do not know what type of geometries will be in the answer set, thus we do not know what properties of the geometry we can query. But we strongly disagree with the suggestion of [51] for a query language depending on SPARQL's DESCRIBE query form to solve such problems. Since the DESCRIBE form is not given any semantics by the SPARQL W3C specification[9], this creates bigger problems than the ones it attempts to solve. We point out that this problem does not arise in stSPARQL because geometries are uniformly represented by linear constraints.

Finally, one might wonder about stSPARQL since, although based on the W3C standard query language for RDF, it does not strive for compatibility with other current standards such as those proposed by OGC and the family of ISO 19100 standards. We have not tried to be compliant with OGC/ISO 19100 standards because this would imply the use of OGC datatypes for various kinds of geometries, something which we find to have problems as we explain above. If one does not want to use the constraint-based approach that we adopt, one can still define a straightforward extension to SPARQL based on datatypes to achieve similar goals.

For example, one could introduce datatypes like Point, Line, Polygon etc. and specify them using XML Schema. One could take the GeoRSS ontology (or a similar ontology based on the OpenGIS Simple Features Specification) and define

[9]Retrieved March 1, 2009, from `http://www.w3.org/TR/rdf-sparql-query/`

datatypes like `georss:Point` or `georss:Box` and then introduce the appropriate operators in SPARQL e.g., introduce an operator `@` to check whether a geometry is completely contained by another geometry. Then one could describe a sensor with the following triples:

```
ex:located_at rdfs:range <georss:Point> .
ex:sensor1 rdf:type ex:Sensor .
ex:sensor1 ex:located_at "5 5"^^georss:Point .
```

Then one could write a window query that asks for all the sensors that are located inside the rectangle R(0, 0, 10, 10) as follows:

```
select ?sensor
where  {?sensor rdf:type ex:Sensor .
        ?sensor ex:located_at ?sensor_loc .
        s-filter(?sensor_loc @ "0 0 10 10"^^georss:Box) }
```

This approach would have the same problems pointed out earlier.

3.5 Summary

In this chapter we extended RDF(S) with the ability to represent spatial and temporal data so that sensor metadata can be represented and queried in an efficient way. We developed the model *stRDF* which extends RDF with the ability to represent spatial data with a temporal dimension. We also presented *stSPARQL*, a declarative query language for querying *stRDF* graphs.

Chapter 4

Conclusions and Future Work

We studied the problem of designing a data model and a query language for the registry of the SemsorGrid4Env infrastructure and generally for a registry in a Semantic Sensor Grid.

We surveyed related work in the areas most relevant to our work. We started with an overview of the work accomplished already for other registries of sensor networks. Then, we surveyed the state-of-the art in query languages for temporal and spatial information for various models such as databases, XML-based models, RDF-based models, description logics and OWL.

We proposed the data model stRDF and the query language stSPARQL that will be supported by the SemsorGrid4Env registry. stRDF extends RDF(S) with the ability to represent spatial and temporal data so that sensor metadata can be represented and queried. stSPARQL extends SPARQL so that spatial and temporal data can be queried using a declarative and user-friendly language.

In the next deliverable of this work package (D3.2: Distributed data structures and algorithms for a Semantic Sensor Grid registry), we will continue the work we started in this deliverable as follows:

- We will give an algebra for stSPARQL (in the spirit of the algebra for SPARQL given in [75]). This will be the algebraic language that will drive the stSPARQL implementation.

- We will present a preliminary implementation of stSPARQL that will serve as the basis of the SemsorGrid4Env registry implementation. We expect this preliminary implementation to deal only with a subset of stSPARQL that is of immediate use in the project and possibly be centralized (as opposed to the distributed implementation that we will eventually produce as described in the Technical Annex of the project).

- We will discuss distributed data structures and algorithms that allow us to develop a full-scale distributed implementation of stSPARQL.

Bibliography

[1] Semantic Sensor Web. Wikipedia entry, Retrieved February 10, 2008, from `http://en.wikipedia.org/wiki/Semantic_Sensor_Web`.

[2] S. Abiteboul, P. Kanellakis, and G. Grahne. On the Representation and Querying of Sets of Possible Worlds. In *Proceedings of the ACM SIGMOD International Conference on Management of Data*, pages 34–48, 1987.

[3] Ilsoo Ahn, Gad Ariav, Don Batory, James Clifford, Curtis E. Dyreson, Ramez Elmasri, Fabio Grandi, Christian S. Jensen, Wolfgang Kaefer, Nick Kline, Krishna Kulkarni, T. Y. Cliff Leung, Nikos Lorentzos, John F. Roddick, Arie Segev, Michael D. Soo, and Suryanarayana M. Sripada. *The TSQL2 Temporal Query Language*. Kluwer Academic Publishers, 1995.

[4] James F Allen. Maintaining Knowledge about Temporal Intervals. *Communications of the ACM*, 26(11):832–843, 1983.

[5] Renzo Angles and Claudio Gutierrez. The Expressive Power of SPARQL. In *International Semantic Web Conference (ISWC 2008)*, pages 114–129, 2008.

[6] Budak I. Arpinar, Amit Sheth, Cartic Ramakrishnan, Usery, Molly Azami, and Mei-Po Kwan. Geospatial Ontology Development and Semantic Analytics. *Transactions in GIS*, 10(4):551–575, July 2006.

[7] Alessandro Artale and Enrico Franconi. A Survey of Temporal Extensions of Description Logics. *Annals of Mathematics and Artificial Intelligence*, 30(1-4):171–210, 2000.

[8] Alessandro Artale and David Toman. Decidable Reasoning over Timestamped Conceptual Models. In Franz Baader, Carsten Lutz, and Boris Motik, editors, *Proceedings of the 21st International Workshop on Description Logics (DL2008)*, volume 353 of *CEUR Workshop Proceedings*, Dresden, Germany, 2008. CEUR-WS.org.

[9] Bernd Bank, Max Egenhofer, Joos Heintz, Bart Kuijpers, and Peter Revesz. 07212 Manifesto – Constraint Databases, Geometric Elimination ang Geographic Information Systems. In Bernd Bank, Max J. Egenhofer,

and Bart Kuijpers, editors, *Constraint Databases, Geometric Elimination and Geographic Information Systems*, number 07212 in Dagstuhl Seminar Proceedings, Dagstuhl, Germany, 2007. Internationales Begegnungs- und Forschungszentrum für Informatik (IBFI), Schloss Dagstuhl, Germany.

[10] D. Brickley and R.V. Guha. Resource Description Framework (RDF) Schema Specification 1.0. Technical report, W3C Recommendation, 2000.

[11] François Bry, Bernhard Lorenz, Hans Jürgen Ohlbach, and Mike Rosner. A Geospatial World Model for the Semantic Web. In *Proceedings of Third Workshop on Principles and Practice of Semantic Web Reasoning, Dagstuhl, Germany (11th–16th September 2005)*. INRIA, 2005.

[12] C.C. Chang and H.J. Keisler. *Model Theory*. North Holland, 1973.

[13] Jan Chomicki. Temporal Query Languages: A Survey. In *ICTL '94: Proceedings of the First International Conference on Temporal Logic*, pages 506–534, London, UK, 1994. Springer-Verlag.

[14] Mike Clark, Craig Hutton, Jason Sadler, and Samantha Roe. Coastal and Estuarine Flood Warning in Southern UK: Requirements Specification. Deliverable D7.1v1, SemsorGrid4Env project.

[15] Anthony G. Cohn, Brandon Bennett, John Gooday, and Nicholas Mark Gotts. Qualitative Spatial Representation and Reasoning with the Region Connection Calculus. *Geoinformatica*, 1(3):275–316, 1997.

[16] The SemsorGrid4Env consortium. Semantic Sensor Grids for Rapid Application Development for Environmental Management (White Paper). Unpublished manuscript.

[17] Chris Date and Hugn Darwen. *Temporal Data and the Relational Model*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 2002.

[18] Nicholas Dawes, K. Ashwin Kumar, Sebastian Michel, Karl Aberer, and Michael Lehning. Sensor Metadata Management and its Application in Collaborative Environmental Research. In *4th IEEE International Conference on e-Science*, 2008.

[19] Amol Deshpande, Suman Kumar Nath, Phillip B. Gibbons, and Srinivasan Seshan. Cache-and-Query for Wide Area Sensor Databases. In *Proceedings of the 22th ACM SIGMOD International Conference on Management of Data (SIGMOD 2003)*, pages 503–514, 2003.

[20] Victor Manuel Diaz, Israel Liebana, and Agustin Izquierdo. Fire Risk Monitoring and Warning in NorhtWest Spain: Requirements Specification. Deliverable D6.1v1, SemsorGrid4Env project.

[21] Max J. Egenhofer. Spatial SQL: A Query and Presentation Language. *IEEE Transactions on Knowledge and Data Engineering*, 6(1):86–95, 1994.

[22] Max J. Egenhofer. Toward the Semantic Geospatial Web. In *GIS '02: Proceedings of the 10th ACM international symposium on Advances in geographic information systems*, pages 1–4, New York, NY, USA, 2002. ACM Press.

[23] Max J. Egenhofer and Andrew U. Frank. Towards a Spatial Query Language: User Interface Considerations. In *Proceedings of the 14th International Conference on Very Large Data Bases*, pages 124–133, San Francisco, CA, USA, 1988. Morgan Kaufmann Publishers Inc.

[24] Max J. Egenhofer and Robert D. Franzosa. Point-Set Topological Spatial Relations. In *Proceedings of International Journal of Geographical Information Systems*, volume 5, pages 161–174, 1991.

[25] H.B. Enderton. *A Mathematical Introduction to Logic*. Academic Press, 1972.

[26] Andy Seaborne Eric Prud'hommeaux. SPARQL Query Language for RDF. W3C Recommendation 15 January 2008. Retrieved March 1, 2008, from `http://www.w3.org/TR/rdf-sparql-query/`.

[27] SCHOOL = Institute for Geoinformatics, University of Muenster YEAR = 2008 Eva Klien, TITLE = "Semantic Annotation of Geographic Information". PhD thesis.

[28] Dengfeng Gao and Richard T. Snodgrass. Temporal Slicing in the Evaluation of XML Queries. In *Proceedings of the 29th International Conference on Very Large Data Bases (VLDB 2003)*, pages 632–643, 2003.

[29] PB. Gibbons, B. Karp, Y. Ke, S. Nath, and S. Seshan. IrisNet: An Architecture for a World-Wide Sensor Web. *IEEE Pervasive Computing*, 2(4), October-December 2003.

[30] G. Grahne. *The Problem of Incomplete Information in Relational Databases*, volume 554 of *Lecture Notes in Computer Science*. Springer Verlag, 1991.

[31] Stéphane Grumbach, Manolis Koubarakis, Philippe Rigaux, Michel Scholl, and Spiros Skiadopoulos. Spatio-temporal Models and Languages: An Approach Based on Constraints. In *Spatio-Temporal Databases: The CHOROCHRONOS Approach*, pages 177–201. Springer, 2003.

[32] Stéphane Grumbach, Philippe Rigaux, and Luc Segoufin. The DEDALE System for Complex Spatial Queries. In *Proceedings of the 1998 ACM SIGMOD*

international conference on Management of data (SIGMOD '98), pages 213–224, New York, NY, USA, 1998. ACM.

[33] Claudio Gutierrez, Carlos Hurtado, and Alejandro Vaisman. Introducing Time into RDF. *IEEE Transactions on Knowledge and Data Engineering*, 19(2):207–218, 2007.

[34] Claudio Gutierrez, Carlos Hurtado, and Ro Vaisman. Temporal RDF. In *European Conference on the Semantic Web (ECSW05)*, pages 93–107, 2005.

[35] Ralf Hartmut Guting. An Introduction to Spatial Database Systems. *VLDB Journal*, 3:357–399, 1994.

[36] Ralf Hartmut Güting, Michael H. Böhlen, Martin Erwig, Christian S. Jensen, Nikos A. Lorentzos, Markus Schneider, and Michalis Vazirgiannis. A Foundation for Representing and Quering Moving Objects. *ACM Transantions on Database Systems*, 25(1):1–42, 2000.

[37] Ralf Hartmut Guting and Markus Schneider. *Moving Objects Databases*. Morgan Kaufmann Publishers Inc., 2005.

[38] Farshad Hakimpour, Boanerges Aleman-meza, Matthew Perry, and Amit Sheth. Data Processing in Space, Time and Semantics Dimensions. In *Terra Cognita 2006 - Directions to the Geospatial Semantic Web, in conjunction with the Fifth International Semantic Web Conference (ISWC '06)*, 2006.

[39] Desiree Hilbring and Thomas Uslander. Catalogue Services Enabling Syntactical and Semantic Interoperability in Environmental Risk Management Architectures. In Klaus Tochtermann and Arno Scharl, editors, *Proceedings of the 20th International Conference on Informatics for Environmental Protection (EnviroInfo 2006)*, pages 39–46, Graz, Austria, September 2006. Aachen Shaker Verlag.

[40] Jerry R. Hobbs and Feng Pan. An Ontology of Time for the Semantic Web. *ACM Transactions on Asian Language Information Processing*, 3(1):66–85, 2004.

[41] Carlos A. Hurtado and Alejandro A. Vaisman. Reasoning with Temporal Constraints in RDF. In *Principles and Practice of Semantic Web Reasoning*, pages 164–178. Springer, 2006.

[42] T. Imielinski and W. Lipski. Incomplete Information in Relational Databases. *Journal of ACM*, 31(4):761–791, 1984.

[43] Open Geospatial Consortium Inc. OpenGIS Geography Markup Language Encoding Standard (GML). Approved July 2007.

[44] Open Geospatial Consortium Inc. OpenGIS Sensor Model Language (SensorML) Implementation Specification. Retrieved March 1, 2009, from `http://www.opengeospatial.org/standards/sensorml/`, 2007.

[45] J. Ledlie M. Roussopoulos M. Seltzer M. Welsh J. Shneidman, P. Pietzuch. Hourglass: An Infrastructure for Connecting Sensor Networks and Applications. Harvard Technical Report TR-21-04, 2004.

[46] Christian S. Jensen and Richard T. Snodgrass. Temporal Data Management. *IEEE Transactions on Knowledge and Data Engineering*, 11(1):36–44, 1999.

[47] Froduald Kabanza, Jean-Marc Stévenne, and Pierre Wolper. Handling Infinite Temporal Data. *Journal of Computer and System Sciences*, 51(1):3–17, 1995.

[48] P.C. Kanellakis, G.M. Kuper, and P.Z. Revesz. Constraint Query Languages. In *Proceedings of the 9th ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems*, pages 299–313, 1990.

[49] P.C. Kanellakis, G.M. Kuper, and P.Z. Revesz. Constraint Query Languages. *Journal of Computer and System Sciences*, 51:26–52, 1995.

[50] Yarden Katz and Bernardo Cuenca Grau. Representing Qualitative Spatial Information in OWL DL. In *Proceedings of the First International Workshop: OWL Experiences and Directions*, Galway, Ireland, November 2005.

[51] Dave Kolas. Supporting Spatial Semantics with SPARQL. In *Terra Cognita Workshop (Terra Cognita 2008)*, Karlsruhe, Germany, 2008.

[52] Dave Kolas, John Hebeler, and Mike Dean. Geospatial Semantic Web: Architecture of Ontologies. In *Proceedings of the First International Conference on GeoSpatial Semantics (GeoS 2005)*, volume 3799 of *Lecture Notes in Computer Science*, pages 183–194. Springer, 2005.

[53] Dave Kolas and Troy Self. Spatially Augmented Knowledgebase. In *Proceedings of the 6th International Semantic Web Conference and 2nd Asian Semantic Web Conference (ISWC/ASWC2007)*, Berlin, Heidelberg.

[54] M. Koubarakis. Representation and Querying in Temporal Databases: the Power of Temporal Constraints. In *Proceedings of the 9th International Conference on Data Engineering*, pages 327–334, April 1993.

[55] M. Koubarakis. Database Models for Infinite and Indefinite Temporal Information. *Information Systems*, 19(2):141–173, March 1994.

[56] M. Koubarakis. Foundations of Indefinite Constraint Databases. In A. Borning, editor, *Proceedings of the 2nd International Workshop on the Principles and Practice of Constraint Programming (PPCP'94)*, volume 874 of *Lecture Notes in Computer Science*, pages 266–280. Springer Verlag, 1994.

[57] M. Koubarakis. The Complexity of Query Evaluation in Indefinite Temporal Constraint Databases. *Theoretical Computer Science*, 171:25–60, January 1997. Special Issue on Uncertainty in Databases and Deductive Systems, Editor: L.V.S. Lakshmanan.

[58] Manolis Koubarakis, Timos K. Sellis, Andrew U. Frank, Stéphane Grumbach, Ralf Hartmut Güting, Christian S. Jensen, Nikos A. Lorentzos, Yannis Manolopoulos, Enrico Nardelli, Barbara Pernici, Hans-Jörg Schek, Michel Scholl, Babis Theodoulidis, and Nectaria Tryfona, editors. *Spatio-Temporal Databases: The CHOROCHRONOS Approach*, volume 2520 of *Lecture Notes in Computer Science*. Springer, 2003.

[59] Manolis Koubarakis and Spiros Skiadopoulos. Querying Temporal and Spatial Constraint Networks in PTIME. *Artificial Intelligence*, 123(1-2):223–263, 2000.

[60] Bart Kuijpers. Linear versus Polynomial Constraint Databases. In Shashi Shekhar and Hui Xiong, editors, *Encyclopedia of GIS*, pages 612–615. Springer, 2008.

[61] Gabriel Kuper, Leonid Libkin, and Jan Paredaens. *Constraint Databases*. Springer Verlag, 2000.

[62] Gabriel Kuper, Sridhar Ramaswamy, Kyuseok Shim, and Jianwen Su. A Constraint-based Spatial Extension to SQL. In *Proceedings of the 6th International Symposium on Advances in Geographic Information Systems*, 1998.

[63] O. Lassila and R. R. Swick. Resource Description Framework (RDF) Model and Syntax Specification. Technical report, W3C Recommendation, 1999.

[64] Rob Lemmens, Carlos Granell, Andreas Wytzisk, Rolf de By, Michael Gould2, and Peter van Oosterom. Semantic and syntactic service descriptions at work in geo service chaining. In *Proceedings of the 9the AGILE conference on geographic information science, shaping the future of geographic information science in Europe*, pages 51–61, April 2006.

[65] C. Lutz. Combining Interval-based Temporal Reasoning with General TBoxes. *Artificial Intelligence*, 152(2):235–274, February 2004.

[66] Carsten Lutz and Maja Milicic. A Tableau Algorithm for Description Logics with Concrete Domains and General TBoxes. *Journal of Automated Reasoning*, 38(1-3):227–259, 2007.

[67] Patrick Maue. An extensible semantic catalogue for (geospatial) web services. Diploma Thesis. Institute for Geoinformatics, University of Muenster., 2006.

[68] Deborah L. McGuinness and Frank van Harmelen. OWL Web Ontology Language. Technical report, W3C Recommendation, 2004.

[69] Jim Melton and Andrew Eisenberg. Sql multimedia and application packages (sql/mm). *SIGMOD Record*, 30(4):97–102, 2001.

[70] Sebastian Michel, Ali Salehi, Liqian Luo, Nicholas Dawes, Karl Aberer, Guillermo Barrenetxea, Mathias Bavay, Aman Kansal, K. Ashwin Kumar, Suman Nath, Marc Parlange, Stewart Tansley, Catharine van Ingen, Feng Zhao, and Yongluan Zhou. Environmental Monitoring 2.0. In *25th International Conference on Data Engineering (ICDE) (Demo paper)*, 2009.

[71] Viorel Milea, Flavius Frasincar, Uzay Kaymak, and Tommaso di Noia. An OWL-Based Approach Towards Representing Time in Web Information Systems. In *Proceedings of the 4th International Workshop of Web Information Systems Modeling Workshop (WISM 2007)*, pages 791–802. Tapir Academic Press, 2007.

[72] Alina Dia Miron, Jerome Gensel, and Marlene Villanova-Oliver. Semantic Analysis for the Geospatial Web - Application to OWL-DL Ontologies. In *Proceedings of the 8th Symposium on Web and Wireless Geographical Information System (W2GIS)*, volume 5373, pages 37–49. Springer, 2008.

[73] S. Nath, J. Liu, and F Zhao. Challenges in Building a Portal for Sensors World-Wide. In *First Workshop on World-Sensor-Web: Mobile Device Centric Sensory Networks and Applications (WSW2006)*, Boulder CO, October 2006.

[74] M. P. Papazoglou. Service-oriented Computing: Concepts, Characteristics and Directions. In *Proceedings of the 4th International Conference on Web Information Systems Engineering (WISE 2003)*, pages 3–12, 2003.

[75] Jorge Pérez, Marcelo Arenas, and Claudio Gutierrez. Semantics and Complexity of SPARQL. In *Proceedings of the 5th International Semantic Web Conference (ISWC 2006)*, pages 30–43, 2006.

[76] Jorge Pérez, Marcelo Arenas, and Claudio Gutierrez. Semantics of SPARQL. Technical Report, Universidad de Chile TR/DCC-2006-17, 2006. Retrieved March 1, 2008, from `http://ing.utalca.cl/~jperez/papers/sparql_semantics.pdf`.

[77] Matthew Perry. *A Framework to Support Spatial, Temporal and Thematic Analytics over Semantic Web Data*. PhD thesis, Wright State University, 2008.

[78] Matthew Perry, Farshad Hakimpour, and Amit P. 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, 2006.

[79] Matthew Perry, Amit P. Sheth, Farshad Hakimpour, and Prateek Jain. Supporting Complex Thematic, Spatial and Temporal Queries over Semantic Web Data. In *Proceedings of the Second International Conference on GeoSpatial Semantics (GeoS)*, pages 228–246, 2007.

[80] Agnes Voisard Philippe Rigaux, Michel Scholl. *Spatial Databases*. Morgan Kaufmann Publishers, 2001.

[81] PR. Pietzuch and J. Bacon. Peer-to-peer Overlay Broker Networks in an Eventbased Middleware. In *Second International Workshop on Distributed Event-Based Systems (DEBS'03)*, 2003.

[82] Andrea Pugliese, Octavian Udrea, and V. S. Subrahmanian. Scaling RDF with Time. In *WWW '08: Proceeding of the 17th international conference on World Wide Web*, pages 605–614, 2008.

[83] C. S. Jensen R. T. Snodgrass, M. H. Bohlen and A. Steiner. Transitioning Temporal Support in TSQL2 to SQL3. In O. Etzion, S. Jajodia, and S. Sripada, editors, *Temporal Databases: Research and Practice*, pages 150–194. Springer, 1998.

[84] Peter Z. Revesz. *Introduction to Constraint Databases*. Springer, 2002.

[85] P. Rigaux, M. Scholl, and A. Voisard. *Introduction to Spatial Databases: Applications to GIS*. Morgan Kaufmann, Reading, MA, 2000.

[86] Philippe Rigaux, Michel Scholl, Luc Segoufin, and Stephane Grumbach. Building a constraint-based spatial database system: model, languages, and implementation. *Information Systems*, 28(6):563–595, 2003.

[87] Flavio Rizzolo and Alejandro A. Vaisman. Temporal XML: Modeling, Indexing, and Query Processing. *VLDB J.*, 17(5):1179–1212, 2008.

[88] Francesca Rossi, Peter van Beek, and Toby Walsh. *Handbook of Constraint Programming*. Elsevier, 2006.

[89] Andre Santanche, Suman Nath, Jie Liu, Bodhi Priyantha, and Feng Zhao. Senseweb: Browsing the physical world in real time. In *Demo Abstract*, Nashville, TN, April 2006.

[90] Arno Scharl and Klaus Tochtermann, editors. *The Geospatial Web: How Geobrowsers, Social Software and the Web 2.0 are Shaping the Network Society*. Springer, London, 2007.

[91] A. Sheth, C. Henson, and S. S. Sahoo. Semantic Sensor Web. *Internet Computing, IEEE*, 12(4):78–83, 2008.

[92] A. Sheth and M. Perry. Traveling the Semantic Web through Space, Time, and Theme. *IEEE Internet Computing*, 12(2):81–86, 2008.

[93] R. Singh, A. Turner, M. Maron, and A. Doyle. GeoRSS: Geographically Encoded Objects for RSS Feeds. Retrieved May 18, 2008, from `http://georss.org/gml`, 2008.

[94] R. Snodgrass. The Temporal Query Language TQuel. *ACM Transcactions on Database Systems*, 12(2):247–298, June 1987.

[95] R. Snodgrass and I. Ahn. A Taxonomy of Time in Databases. In *Proceedings of ACM SIGMOD International Conference on Management of Data*, pages 236–246, 1985.

[96] A. Szalay, J. Gray, G. Fekete, P. Kunszt, P. Kukol, and A. Thakar. Indexing the Sphere with the Hierarchical Triangular Mesh. Microsoft Research Technical Report MSR-TR-2005-123, September 2005.

[97] David Toman. SQL/TP: A Temporal Extension of SQL. In *Constraint Databases*, pages 391–399. Springer Verlag, 2000.

[98] Goce Trajcevski, Ouri Wolfson, Klaus Hinrichs, and Sam Chamberlain. Managing Uncertainty in Moving Objects Databases. *ACM Transactions of Database Systems*, 29(3):463–507, 2004.

[99] T. Uslander. Specification of the Sensor Service Architecture V1. Deliverable D2.3.2, SANY Sensor Anywhere - IST FP6 Integrated Project.