Ontology Pattern-Based Data Integration

A dissertation submitted in partial fulfilment
of the requirements for the degree of
Doctor of Philosophy

By

ADILA ALFA KRISNADHI
S.Kom., Universitas Indonesia, 2002
M.Sc., Technische Universität Dresden, 2007

2015
Wright State University
Dayton, Ohio, USA

WRIGHT STATE UNIVERSITY
SCHOOL OF GRADUATE STUDIES

December 18, 2015

I HEREBY RECOMMEND THAT THE DISSERTATION REPARED UNDER MY SUPERVISION BY Adila Alfa Krisnadhi ENTITLED Ontology Pattern-Based Data Integration BE ACCEPTED IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE DEGREE OF Doctor of Philosophy.

Pascal Hitzler, Ph.D.
Dissertation Director

Michael L. Raymer, Ph.D.
Director, Computer Science and Engineering PhD Program

Robert E.W. Fyffe, Ph.D.
Vice President of University Research and Dean of Graduate School

Committee on
Final Examination

Pascal Hitzler, Ph.D

Krzysztof Janowicz, Ph.D

Krishnaprasad Thirunarayan, Ph.D

Michelle A. Cheatham, Ph.D.

ABSTRACT

Krisnadhi, Adila Alfa. Ph.D. Department of Computer Science and Engineering, Wright State University, 2015. Ontology Pattern-Based Data Integration.

Data integration is concerned with providing a unified access to data residing at multiple sources. Such a unified access is realized by having a global schema and a set of mappings between the global schema and the local schemas of each data source, which specify how user queries at the global schema can be translated into queries at the local schemas. Data sources are typically developed and maintained independently, and thus, highly heterogeneous. This causes difficulties in integration because of the lack of interoperability in the aspect of architecture, data format, as well as syntax and semantics of the data.

This dissertation represents a study on how small, self-contained ontologies, called ontology design patterns, can be employed to provide semantic interoperability in a cross-repository data integration system. The idea of this so-called ontology pattern-based data integration is that a collection of ontology design patterns can act as the global schema that still contains sufficient semantics, but is also flexible and simple enough to be used by linked data providers. On the one side, this differs from existing ontology-based solutions, which are based on large, monolithic ontologies that provide very rich semantics, but enforce too restrictive ontological choices, hence are shunned by many data providers. On the other side, this also differs from the purely linked data based solutions, which do offer simplicity and flexibility in data publishing, but too little in terms of semantic interoperability.

We demonstrate the feasibility of this idea through the actual development of a large scale data integration project involving seven ocean science data repositories

from five institutions in the U.S. In addition, we make two contributions as part of this dissertation work, which also play crucial roles in the aforementioned data integration project. First, we develop a collection of more than a dozen ontology design patterns that capture the key notions in the ocean science occurring in the participating data repositories. These patterns contain axiomatization of the key notions and were developed with an intensive involvement from the domain experts. Modeling of the patterns was done in a systematic workflow to ensure modularity, reusability, and flexibility of the whole pattern collection. Second, we propose the so-called pattern views that allow data providers to publish their data in very simple intermediate schema and show that they can greatly assist data providers to publish their data without requiring a thorough understanding of the axiomatization of the patterns.

Contents

List of Figures

List of Tables

Acknowledgements

Alhamdulillah. All praises are due to Allah, the Lord of the Universe, the Most Compassionate, and the Most Merciful. He showered me with His countless blessings due to which I was able to complete this dissertation work.

Throughout my doctoral study, I have been very fortunate to receive support from many great people. First and foremost, I would like to extend my heartfelt gratitude to Dr. Pascal Hitzler who has been both a great advisor and mentor in study, research, and life. Finishing this dissertation work would have been impossible without his numerous suggestions, constructive critiques, and recommendations. Sessions with him were almost always about throwing ideas at each other, which I would fondly cherish all my life.

I would like to express my special appreciation to members of my dissertation committee, Dr. Krzysztof Janowicz, Dr. Khrisnaprasad Thirunarayan, and Dr. Michelle Cheatham, for their valuable comments and suggestions to improve this dissertation work. A special mention goes to Dr. Krzysztof Janowicz for the invaluable discussions on Skype as well as during several occasion we were able to meet offline. A significant amount of ideas in this dissertation were inspired by those discussions.

My family has always been one of the key pillar of support in my life. My wife, Tuntas Margi Hartini, has been there for me both in my happiest and my most difficult moments. I could never count all the love, the prayers, the smiles, the kindness, and all other big and little things she did and keeps on doing for me, not just during the years of my doctoral study, but also the years before that, and hopefully, throughout the rest of my life. I am immensely grateful to my mother, Dyah Pamularsih, and

my father, Imam Soegito, also to my two brothers, Andika and Aditya, for their unconditional love, prayers, and support. I am also thankful for the support from the rest of extended family.

I would like to thank my colleagues at the Data Semantics Lab and the CSE Department of Wright State University for their support. I am also grateful with the support and encouragement from my Indonesian friends in Dayton and Columbus area. I also appreciate the support of the Faculty of Computer Science, Universitas Indonesia, with a special mention to Dr. Wisnu Jatmiko for his invaluable advice.

I owe a great deal to colleagues at the GeoLink project because this dissertation work relies critically on the application context and practical experience provided by the project. The GeoLink patterns would not have been realized without the intense collaborative effort expended by all of the project partners at the Woods Hole Oceanographic Institution, the Lamont-Doherty Earth Obsevatory at Columbia University, Marymount University, University of California Santa Barbara, and Consortium for Ocean Leadership. In particular, I would like to mention Adam Shepherd, Tom Narock, and Robert Arko for squeezing some spare time in their busy schedule to help me with the evaluation of the ideas of this dissertation.

My doctoral study has been funded by generous funding from several sources: Wright State University through the tuition waiver and various travel support; the US State Department and the Indonesian government through the Fulbright Indonesian Presidential Scholarship 2010-2013; and the US National Science Foundation (NSF) through the three projects, namely the NSF Project 1017225 "III: Small: TROn – Tractable Reasoning with Ontologies", the NSF Award 1354778 "EAGER: Collaborative Research: EarthCube Building Blocks, Leveraging Semantics and Linked Data for Geoscience Data Sharing and Discovery", and the NSF Award 1440202 "EarthCube Building Blocks: Collaborative Proposal: GeoLinkLeveraging Semantics, and Linked Data for Data Sharing and Discovery in the Geosciences".

1 Introduction

1.1 Semantic Web and Data Integration

In their paper published almost fifteen years ago [12], Berners-Lee et al. argued that most of the Web content is intended for human consumption and unsuitable for machines to manipulate meaningfully. That is, there is no way for a machine to reliably understand that a particular web address is actually the homepage of a particular person, or a given hyperlink points to that person's curriculum vitae. Their vision was to extend the Web into a form they call the Semantic Web where any piece of information available in it is equipped with a well-defined description of their meaning or *semantics*, and thus, can be understood by machines without the need of a human involvement in interpreting the information. This would pave the way for machines to carry out more sophisticated tasks without needing a very high level of artificial intelligence.

Since Semantic Web is an extension of the Web, it retains its decentralized nature. Every information source can publish whatever information however the want. No single, overarching, global, centralized authority would be capable of managing all possible information sources on the Web. Nor would it be possible to force every information source to conform to such a centralized authority. Unsurprisingly, this gives rise to an extreme heterogeneity, even when every information source provides formal description explaining the semantics of the information it publishes. On the other hand, satisfying information needs of the users, be it a human or a machine, often requires combining information from multiple sources. In such a situation,

rather than expressing the information need directly at each information source, which may require the information need to be translated into a form that is suitable to the schema and constraints and the information source, users would of course benefit from an extra layer of abstraction that contains a unified view over the data and hides the translation from them.

The problem of providing users with a unified view over data residing in multiple data sources as illustrated above is not something that only became known when Semantic Web was coined. Traditionally, this problem is called *data integration* and it was originally studied in relational database [89]. Applications of data integration first emerged in the context of enterprise information systems [30, 62], which often have to deal with data scattered across multiple databases – a situation that can easily occur in a large enterprise. In recent years, however, data integration needs have appeared in a wide range of application domains beyond the context of enterprise information systems, for example, in life sciences [53], geosciences [104], life cycle analysis [38, 60], government sector [75], e-commerce [86], and many others.

In a data integration system, the unified view to the users is provided by a *global schema* defined in a particular vocabulary using which users express queries about the data. Meanwhile, each data source involved in the integration is defined according to its own *local schema*, which is hidden from the users and may employ a different vocabulary from the global schema vocabulary. The integration is carried out through a set of mappings between the global and local schema, which allow the system to translate the query from a user into queries over each data source. These data sources may of course be located in separate geographical locations and belong to different organizations, and furthermore, might have been designed, developed, and maintained independently according to locally defined constraints from established business processes or organizational policies.

Data integration is a very hard problem with both technical and non-technical

challenges [42, Section 1.2]. The first kind of challenges have to do with the lack of syntactic and architectural interoperability between the systems involved in a data integration effort. This lack of interoperability may be caused by differences in data formats, software and hardware constraints, access protocol, and query processing capabilities. Even when all the systems run on the same hardware platform and comply with the same data access standard, problems may still happen due to differences in local, vendor-specific software configurations.

The second kind of challenge concerns the structural and semantic heterogeneity in the data. More precisely, since each data source is independently developed and maintained, it is almost unavoidable that the schema developed for a particular data source would be different from schemas from the other data sources. The differences may lie in the use of distinct vocabularies, different levels of granularity in modeling the data, or conflicting conceptualization.

Meanwhile, the third kind of challenge is non-technical, and usually boils down to inability or unwillingness of the data owners to participate in a data integration effort. There are many reasons for this problem. For example, data owners may be unable to participate because their data sources simply do not possess relevant data required by the integration effort. Data owners may also be unwilling to participate because of the fear of unanticipated costs, the aversion toward data sharing, the reluctance to commit resources to make the integration happen, the worry that participating in the integration may entail major changes in the established management practices and policies, and the skepticism about the scalability of the integration itself [122]. Note that non-technical nature of these challenges, they are often no less hard than the technical challenges. Furthermore, they can also be caused by the inherent technical difficulties of realizing the data integration.

Those three challenges are exacerbated if we consider large scale (or even, Web-scale) integration involving a big and very diverse community of stakeholders in which

dozens of data sources have been established, a multitude of institutions are involved and no single stakeholder holds the sole authority over the whole community. Development of a data integration solution for such a community cannot be done in top-down manner. Rather, a more natural way is to start small, i.e., by working with a small part of the community, and then gradually include more stakeholders over time. The desired solution not only needs to be effective, i.e., it works according to its intended purpose for data integration, but also flexible and easily extendible. Flexible here means that data sources still retain a high degree of independence both in technical and non-technical aspects. For example, this implies that data sources do not have to make drastic changes in their local schema design, nor do they have to significantly alter their established business processes just to participate in the data integration. Extendible means that joining (and also, leaving) the data integration does not necessitate costly changes to the whole framework.

Fortunately, Semantic Web offers some ingredient technologies, which we can use to push us toward overcoming the aforementioned challenges if one were to develop a data integration solution. In fact, although Berners-Lee et al. did not mention data integration explicitly in their paper [12], they alluded to it through the emphasis on *linking* data and the use of *ontologies*. The former corresponds to the Resource Description Framework (RDF) which became World Wide Web Consortium (W3C) standards [39, 76, 88], and today, it forms a core part of a suite of technology standards called Linked Data [13]. This suite not only contains standard syntax formats for data publishing, but also adopts the simplicity of HTTP[1]-based architecture and triple-based graph data model. Development of various software solutions, tools, and APIs around this technology suite in the last decade has allowed virtually everyone to publish data according to the Linked Data architecture relatively easily. All of these simplify the issue regarding the lack of syntactic and architectural interoperablity.

[1]the HyperText Transfer Protocol

Meanwhile, the more challenging problem is concerned with semantic heterogeneity. Even prior to the emergence of Semantic Web, ontology has been considered as a potential solution in addressing semantic heterogeneity problem. In data integration, ontologies play a role of global (and possibly, local) schema by formally defining vocabulary terms used by the application at hand. Since ontologies are designed with the purpose of knowledge sharing and improving semantic interoperability, it has been considered superior than relational database schema in dealing with semantic heterogeneity. This motivated extensive research in ontology-based data integration [99, 125, 133]. In the beginning, there was a somewhat tacit understanding that generic, extensively axiomatized, monolithic, foundational ontologies would serve the need of an ontology-based system in the Semantic Web better [101, Chapter 3]. However, later research trends in Semantic Web appeared to move away from the focus on such monolithic ontologies toward Linked Data and very lightweight, sometimes semiformal, vocabularies [51, 72]. Simplicity of Linked Data is very appealing for many data publishers and they do not even need to understand much of the semantics employed by ontology languages. This spurred an explosive growth on the number of linked datasets forming the so-called Web of Data [114].

The aforementioned shift was also fueled by the difficulties in employing foundational, monolithic ontologies. Indeed, Oberle already indicated that such ontologies are hard to design both conceptually and computationally, and furthermore, they typically make very specific ontological choices [101]. Unless one is very familiar with the ontologies, such choices may not be easy to understand because they may be buried within the complexity of the axiomatization, and even if they can be understood, they may impose too strict ontological commitments, which are not necessarily be accepted by the parties who are supposed to use the ontologies. In the context of cross-repository data integration involving multiple stakeholders, this can lead to an unsurmountable barrier for the success of the integration effort. On the other hand,

as argued by Jain et al., solely relying on Linked Data is not enough because too little schema information also causes problems in understanding and integrating different linked datasets [72].

The above difficulties concerning foundational ontologies motivated the idea of *ontology design patterns* (ODPs), which are reusable solutions to some frequently occurring ontological modeling problem that emerges in different domains [15, 49, 109]. ODPs have emerged as a practical solution for engineering complex, foundational ontologies. The main idea is that one can expend less amount of effort in engineering an ontology if one can reuse small building blocks that recurring frequently when modeling the notions in the ontology. Furthermore, cognitive barrier in understanding the content of an ontology can be reduced if small, modular, easily understood parts can be identified. Interestingly, the use of ODPs as a basis for a flexible integration framework is hitherto still relatively unexplored. This thesis represents our investigation in this direction, building upon earlier results in ontology-based data integration, and drawing from practical experience in the NSF[2]-funded GeoLink project[3] (and its predecessor, the OceanLink project), a cross-repository data integration project for ocean science domain, which is part of NSF's EarthCube program[4].

1.2 Research Questions

Our discussion in the previous sections point to the benefits of both ontologies and Linked Data for data integration. However, it is relatively unexplored whether we can build a data integration system that employs both, while avoiding their pitfalls. This motivates the following cardinal research question.

[2]the U.S. National Science Foundation
[3]http://www.geolink.org
[4]http://www.earthcube.org

Cardinal Research Question. Can we build a linked data integration framework that combines the benefits of both ontologies and Linked Data, while avoiding their pitfalls?

By the benefits of ontologies, we refer to strong, machine readable semantics that provides conceptual clarity and common modeling reference. The benefits of Linked Data refers to the simplicity and flexibility in publishing the data, already in an open and web-friendly way. The pitfalls of ontologies are their undesirable characteristics such as a high barrier for understanding their content, abstractions that are too far from the actual data, overly strict ontological commitments, so rigid that extendibility is impaired, and also, costly engineering effort. Finally, the pitfalls of Linked Data are too little shareable schema information that makes understanding the dataset difficult and causes true integration to be difficult to realize.

The intuition of the cardinal question above is thus a balancing act between two complementary aspects. To arrive at an answer, we can intuitively start from either side. The real bottleneck, however, is the ontology side of the question because we do not have a complete understanding on how to realize a framework that employs ontologies satisfying our desiderata above.

Main Research Question 1 What kind of ontologies are suitable as global schema for cross-repository data integration? Here, most suitable ontologies need to provide conceptual clarity through precise, machine readable semantics, but should not possess undesirable characteristics as mentioned in the cardinal research question. If such kind of ontologies can be identified, how do we best engineer it?

As suggested in the previous section, a hint toward a possible answer to the above question was provided by ontology design patterns (ODPs). ODPs were conceived as building blocks for engineering more complex ontologies, hence they have been mostly used as such. Nevertheless, the versatility of ODPs means that their potential

transcends this original intent. In our case, we see ODPs as the core part of ontology-based linked data integration. One could certainly argue that this is just another case of ontology engineering, but in the literature, modeling of ODPs was rarely specifically intended as the *main* integrating component. Oberle et al. [102] employed ODPs indirectly for integration, that is, as parts of a foundational ontology with which the actual integration was conceived. Their application examples also did not include data integration, but rather, web service integration. Our proposed research is thus to explore the use of ODPs as the main integrating component leading to the following hypothesis.

Hypothesis 1. Ontology design patterns are suitable to be used as the global schema (i.e., the main integrating component) of a data integration involving heterogeneous data sources and they bring the benefits of the traditional ontology-based approach while avoid its pitfalls.

We shall proceed by conducting a construction of ODPs for a data integration purpose based on a number of principles that we illustrate in Chapter 4. This construction is done in the context of the OceanLink and GeoLink projects, aimed at achieving what we hypothesized above. The result was a set of ODPs employed for an integration of several ocean science data repositories, which participate in the aforementioned projects.

It is generally very difficult to find sensible quantitative measures on the quality of ontologies. Hence as suggested by Oberle [101] who were faced with a similar challenge, a qualitative assessment will be conducted. As part of this assessment process, data providers were asked to publish their data following Linked Data principles and use the ODPs as the basic vocabulary. Setting up a Linked Data repository is the simpler part of this task, and thus, in this thesis, we consider this of secondary im-

portance. The problematic part was establishing the connection between the data and the ODPs. We then study the feedback from the data providers

Feedback from the data providers indicated that application of ODPs still suffer some problems. The most cited one is that abstractions within the ODPs turn out to be still too far from the actual data, and this hinders the effort from the data providers to populate the ODPs with real data. This is formulated as the following research question.

Main Research Question 2. How do we address the gap between the abstraction within ODPs and the local conceptualization on the data providers' side, so that data publishing can be done by retaining the simplicity and flexibility of Linked Data, retaining the simplicity, while keeping the benefits of semantic interoperability from ODPs?

To address this gap, we introduce the notion of *pattern view*, which is inspired from the notion of a view in databases. The idea is to create views to bridge the gap of abstraction between the ODPs and the data. This leads to the following secondary hypothesis.

Hypothesis 2. The pattern view can bridge the gap between the abstraction within ODPs and the local conceptualization of data providers such that the semantic interoperability of the ODPs as well as simplicity and flexibility of Linked Data are retained.

To test this hypothesis, we will first formulate a more rigorous definition of pattern view, study the connection of pattern view and ODPs, and as above, will again perform qualitative assessment. Moreover, by consolidating our results from verifying the two hypotheses above, we can formulate a principled approach in the use of ODPs for data integration.

1.3 Dissertation Overview

Parts of the work needed to realize our proposed research illustrated in the previous section have been conducted and published in a number of publications. This section summarizes our contribution from the research for this dissertation.

1. We will demonstrate the feasibility of establishing a flexible, cross-repository data integration by employing Linked Data and ODPs. Previous exercises that laid a groundwork for the principles followed by this approach were presented in Krisnadhi et al. [80, 82, 83, 84, 85] and Vardeman et al. [126, 127].

2. We will present a set of novel ODPs targeted for the ocean science domain that is suitable for data integration in that domain. This contribution has been presented in Krisnadhi et al. [80, 82, 83, 84].

3. We will provide a novel definition of pattern views and an analysis on this notion in the context of Linked Data integration with ODPs. An initial effort toward this contribution has been presented in Krisnadhi et al. [81, 85] and Rodríguez-Doncel et al. [112].

The remainder of this thesis thus proceeds as follows. In Chapter 2, we provide a background on Semantic Web technologies, as well as ontology languages. For the latter, we provide syntax and semantics of RDFS, OWL, and Datalog/Rule languages, which we use for modeling the ontology patterns. We then proceed to Chapter 3 where we present an overview of data integration problem, discuss how ontology and linked data can play a role in realizing a cross-repository data integration, as well as reveal some of the problems introduced when using them for data integration. This discussion motivates Chapter 4 where we present our contribution of using ontology design patterns (ODPs) for data integration with an application on ocean science data repositories, including selected modeling details of the patterns. We also elaborate therein why Hypothesis 1 is partially affirmed. In Chapter 5, we examine some of the

issues encountered during our attempt to realize an ODP-based data integration in the preceding chapter, i.e., issues that we need to solve to fully affirm Hypothesis 1, which is captured in Hypothesis 2. Next, we describe pattern view as an idea to solve these issues, thus representing half of Hypothesis 2. Chapter 6 presents a qualitative evaluation on the use of pattern view as part of ODP-based data integration framework. This complete the other half of Hypothesis 2, and by fulfilling Hypothesis 2, we fully verified Hypothesis 1. We then present our conclusion in Chapter 7. In addition, this thesis also contains Appendix A in which we include details of all patterns we modeled for the data integration.

2 Semantic Web and Ontology Languages

2.1 The Web

When Tim Berners-Lee proposed the (World Wide) Web in 1989, he intended it to be a kind of gigantic information system, which can act as a medium in which people and machines could communicate and share *any* kind of information [10]. There were six criteria he suggested that the Web should be based on. The first was that in the Web, anyone should be able to link any object to any other object. This naturally would go beyond any kind of database systems since such systems would impose restrictions on the kind of associations one can make between particular objects. Secondly, it should be possible to create links between any two independent parts of the Web incrementally without resorting to expensive operations such as database merging. Third, using the Web should not be constrained on particular operating systems. Fourth, any information available on the Web should never be limited to particular platforms. Furthermore, future platforms should not be excluded in this regard. Fifth, how users model data on the Web mentally should not be restricted to a particular pattern. Finally, adding and modifying information on the Web should be easy to do by the user who is directly knowledgeable with it. These design criteria led to an architecture that was very generic, simple, flexible, and usable across the Internet. The architecture essentially consists of Uniform Resource Identifier (URI) and Hypertext Transfer Protocol (HTTP) as its basic components.

URI, URL, URN, and IRI

Any object that lives on the Web is called a *resource*. The design criteria of the web architecture places a lot of emphasis on links between resources. The first and main web resource format is hypertext, i.e., text that contains links to other web resources. Obviously, a link would need a way to say which object does it point to. This is achieved through a *Uniform Resource Identifier* (*URI*)[1]. A URI is simply a string that is used as an identifier of a resource on the Web. The syntax conforms to the following grammar as specified in RFC 3986 [11]:

```
URI = scheme ':' hier-part [ '?' query ] [ '#' fragment ]
```

For example, http://dase.cs.wright.edu/people/Adila-Krisnadhi is a URI identifying my webpage within the Data Semantics Lab's website.

A URI can serve dual purposes. First, it can serve as a *name* of a resource in which case, it would be interpreted as a *Uniform Resource Name* (*URN*). The second interpretation is that a URI also indicates the *location* of a resource. In this case, a URI is interpreted as a *Uniform Resource Locator* (*URL*). Since a URI may indicate a location, it can be *dereferenced* or *resolved*, though the Web Architecture does not require every URI to be dereferenceable. That is, a resource can contain a link to another resource without providing a guarantee that the linked resource exists. In fact, each URI is treated as an opaque string and nothing about the object it references can be inferred solely using that string. Also, there is no guarantee of uniqueness with regards to an object's URI. That is, it is possible that multiple URIs to refer to the same resource. Finally, it was later realized that URI, as defined in the standard, cannot be written using characters outside of US-ASCII character set, so it was generalized into *Internationalized Resource Identifier* (*IRI*) whose only difference

[1]The letter 'U' in URI, URN, and URL used to stand for 'Universal', though nowadays both 'Uniform' and 'Universal' are used almost interchangeably.

with URI is the use of the UCS[2], hence allowing one to use, e.g., Chinese characters to create an identifier [43]. The term 'URI' though remains ubiquitous, and except in very special circumstances, when people talk about URI, they really mean IRI, i.e., the generalized form. Throughout this thesis, we also take this position and use the term 'URI' to mean IRI.

HTTP

The HyperText Transfer Protocol (HTTP) [45] is a stateless data transfer protocol that allows a client (e.g., a web browser) to request a hypertext resource to a server and the server to respond to that request appropriately. The requested resource is identified by its URI, and HTTP specifies how it should be resolved. HTTP is particularly efficient in handling requests for hypertext resources, hence allowing for speedy traversal of links.

Web of Documents

The Web owes its success to the simplicity of architecture. Consuming information on the Web is easy: a user can simply run a web browser with a URI as an input, and a web resource is opened (assuming the URI is resolvable, of course). In a typical situation, the web resource is a hypertext document, marked-up using a standard mark-up language such as HTML[3], which will be readily displayed by the browser. Navigating the links is now as simple as clicking the link and the browser will proceed by requesting the referenced resource to some appropriate web server and again, display it as soon as it receives a successful response. At the other end, publishing a hypertext document on the Web is also fairly easy and cheap. A basic HTML page is quite simple to create and plethora of tools are available to assist anyone creates a

[2]Universal Coded Character Set, colloquially known as Unicode. The latest version of the standard is ISO/IEC 10646 [70]

[3]HyperText Markup Language, http://www.w3.org/html/

hypertext document. One then only needs to put the document at a web server and makes it accessible with an appropriate URI. The ease of publishing and consuming web resources were in fact demonstrated by Tim Berners-Lee himself: when he put out the proposal for the World Wide Web, he immediately followed it with writing the first Web browser, Web server, and a Web page [37]. Together with openness of the technical standards underlying the Web, all of these spur countless innovations and explosive growth of the Web.

Nevertheless, a significant amount of information presentation on the Web is geared toward human consumption. Major Web standards such as HTML, CSS[4], and JavaScript[5] emphasize aspects concerning readability for human, human user experience, and responsiveness to human interaction. Human-centric perspective implies that information are mostly organized into hypertext, or more generally, hypermedia documents, and the linking between them led to so-called *Web of Documents*. The problem with Web of Documents is that although a document does contain some degree of structure – some may even be quite well-structured, the data that may actually describe what the document is all about are most likely intermixed with text intended for presentation purposes. This makes it very difficult for a machine to "understand" the content of the document.

2.2 Semantic Web

The idea of Semantic Web is *Web of Data*, i.e., linking data, instead of linking documents. Semantic Web is not intended to replace or rebuild the Web, but rather, extend it with additional layers, depicted in Fig. 2.1, all of which are aimed at incorporating *machine-processable semantics* into it, hence facilitate data sharing and

[4]Cascading Style Sheets, http://www.w3.org/Style/CSS/

[5]The standardized version is called ECMAScript, see http://www.ecma-international.org/publications/standards/Ecma-262.htm

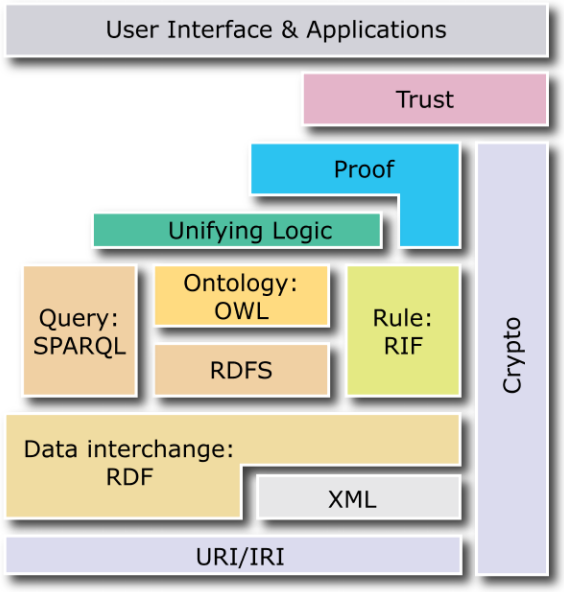


Figure 2.1: Semantic Web Layer Cake. (Original image from http://www.w3.org/2001/sw/layerCake.png)

reuse by people and most importantly by machines too. The first layer is URI/IRI, which is already part of the Web. Here, URIs really function as a name and it is not required to be dereferenceable. Also, as stated it in the preceding section, it is possible to have multiple URIs to reference the same resource. We briefly describe XML, RDF, and SPARQL in this section. RDFS, OWL, and RIF will be described in Section 2.3.

2.2.1 Data Interchange Layer

XML

One of the first things one needs to have to enable data sharing is to agree upon a format. By the time Semantic Web was conceived, the Extensible Markup Language (XML) has been the *de facto* standard format for *data exchange* on the *web*. It was already widely used, compatible with HTML, and already on the way toward a W3C

Recommendation[6]. XML thus became a natural choice of a standard data exchange format for Semantic Web, and despite newer data formats were proposed later on, it remains a part of Semantic Web suite of standards to this day.

Technically, XML facilitates an encoding of data in a nested, tree-like structure. It is platform independent and extensible with the so-called XML Schema, which allows one to specify a restriction on the structure, and also, defines standard datatypes, such as numbers, strings, and dates. Note, however, that although XML Schema expresses a structure for data, it does not express the semantics of the data, i.e., how the data should be interpreted.

RDF

The Resource Description Framework (RDF) is a data model for representing metadata on the Web [39]. It is a graph-based model and logically, can also be seen as an assertional language with a very lightweight semantics. RDF's main feature is the ability to express a link between any two resources on the Web.

Every RDF statement is called an (*RDF*) *triple*, which is a tuple $\langle s, p, o \rangle$ where s is either a URI or a *blank node*, p is a URI, and O is either a URI, a blank node or a *literal*. A triple containing no blank node is called *ground*. An (*RDF*) *graph* is a finite set of RDF triples. An RDF graph may be associated with a URI or a blank node. If so, such a graph is called a *named graph*, while the associated URI or blank node is called the *graph name*. An (*RDF*) *dataset* is a nonempty set of RDF graphs where exactly one graph is unnamed and may be empty, and the remaining graphs are named graphs whose name is unique within the dataset. The unnamed graph in an RDF dataset is called the *default graph*.

Literals represent values such as numbers, strings, and dates. In RDF, a literal is always typed, and many of standard types for literals come from XML [105]. For

[6]Due to various reasons, there are actually *two* W3C Recommendations for XML, namely XML 1.0 [20] and XML 1.1 [21], and the latter does *not* obsolete the former.

example, `"mystring"`, `"7.5^^xsd:decimal"`, and `"true^^xsd:boolean"`[7]. The first one is an example of a simple literal, which is simply an abbreviation of the string literal `"mystring^^xsd:string"`, while the latter two are the decimal number 7.5 and the Boolean value true. Blank nodes, on the other hand, are understood as existential variables in logic: a triple with blank node(s) indicates a relationship involving some unnamed resource(s).

Note that, despite being called a graph, an RDF graph is not strictly a graph in the usual definition in graph theory because there is no strict separation between the set of nodes and the set of edges. That is, within one RDF graph, a URI may appear both as a "node" (subject or object) and an "edge" (predicate). In fact, it is possible that one URI appears as the subject, predicate, and object of a triple simultaneously.

RDF has a model theoretical semantics, which defines the notion of *simple entailment* and *RDF entailment* [61]. However, they amount almost nothing. Simple interpretation interprets every triple as an instance of a binary relation over resources. Simple entailment is essentially graph containment relation: a graph entails all its subgraphs and all graphs obtained from those subgraphs by replacing URIs or literals with blank nodes. RDF entailment fixes semantics of literals, and assigns a restricted semantics to `rdf:type`: any property p is related to `rdf:Property` through the `rdf:type` binary relation. That is, if $\langle s, p, o \rangle$ is in the graph, then the graph would entail $\langle p, \texttt{rdf:type}, \texttt{rdf:Property} \rangle$.

An RDF graph or dataset is usually encoded in an RDF document, which is written in some RDF concrete syntax, or *serialization*. One example of RDF concrete syntax is XML, which was actually the only RDF serialization format that made it into a W3C Recommendation status for RDF 1.0 [76]. The later RDF 1.1 standards [39], besides XML [48] – Fig. 2.2, includes additional concrete syntaxes: N-Triples

[7] The string '`xsd:`' is called a *namespace prefix*, which is an abbreviation of a *URI namespace*. In this particular case, `xsd:` stands for `http://www.w3.org/2001/XMLSchema#`, and thus the URI `xsd:decimal` is an abbreviated form of `http://www.w3.org/2001/XMLSchema#decimal`. Syntax for namespace and namespace prefix is defined in a W3C Recommendation [19].

```
<?xml version="1.0" encoding="utf-8" ?>
<rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
         xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
         xmlns:dbo="http://dbpedia.org/ontology/">

  <rdf:Description rdf:about="http://dbpedia.org/ontology/capital">
    <rdfs:subPropertyOf rdf:resource="http://dbpedia.org/ontology/administrativeHeadCity"/>
  </rdf:Description>

  <rdf:Description rdf:about="http://dbpedia.org/ontology/country">
    <rdfs:subPropertyOf
       rdf:resource="http://www.ontologydesignpatterns.org/ont/dul/DUL.owl#hasLocation"/>
  </rdf:Description>

  <rdf:Description rdf:about="http://dbpedia.org/resource/Ohio">
    <dbo:capital rdf:resource="http://dbpedia.org/reosurce/Columbsus,_Ohio"/>
    <dbo:country rdf:resource="http://dbpedia.org/resource/United_States"/>
  </rdf:Description>
</rdf:RDF>
```

Figure 2.2: RDF triples about the state of Ohio in XML.

```
<http://dbpedia.org/resource/Ohio> <http://dbpedia.org/ontology/capital>
                                    <http://dbpedia.org/resource/Columbus,_Ohio> .
<http://dbpedia.org/resource/Ohio> <http://dbpedia.org/ontology/country>
                                    <http://dbpedia.org/resource/United_States> .
<http://dbpedia.org/ontology/capital> <http://www.w3.org/2000/01/rdf-schema#subPropertyOf>
                                      <http://dbpedia.org/ontology/administrativeHeadCity> .
<http://dbpedia.org/ontology/country> <http://www.w3.org/2000/01/rdf-schema#subPropertyOf>
                                  <http://www.ontologydesignpatterns.org/ont/dul/DUL.owl#hasLocation> .
```

Figure 2.3: RDF triples about the state of Ohio in N-Triple format.

[27] – Fig. 2.3, N-Quads [26], Turtle [111] – Fig. 2.4, TriG [28], JSON-LD [119], and RDFa [63].

2.2.2 Query Language: SPARQL

SPARQL is a query language for querying and manipulating RDF data [117]. It is analogous to what SQL is for relational databases. It supports federated queries, and though it does not provide semantics to the data *per se*, it supports simple graph entailment, and through *entailment regimes*, also supports other entailments, e.g., RDF, RDFS, and OWL entailments. Every SPARQL (non-federated) query is run on an RDF dataset, which can be understood as a set of the form $\{G_0, \langle u_1, G_1\rangle, \ldots, \langle u_n, G_n\rangle\}$ where G_0 is the default graph of the dataset (which always exists, though can be

```
@prefix : <http://dbpedia.org/resource/> .
@prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#> .
@prefix dbo: <http://dbpedia.org/ontology/> .
@prefix dul: <http://www.ontologydesignpatterns.org/ont/dul/DUL.owl#>

<Ohio> dbo:capital <Columbus,_Ohio> ;
       dbo:country <United_States> .
dbo:capital rdfs:subPropertyOf dbo:administrativeHeadCity .
dbo:country rdfs:subPropertyOf dul:hasLocation .
```

Figure 2.4: RDF triples about the state of Ohio in Turtle format.

empty), and the rest are zero or more named graphs $\langle u_i, G_i \rangle$ where u_i is the graph name of G_i in the dataset.

For querying, SPARQL provides the following query operations: SELECT, CONSTRUCT, ASK, DESCRIBE, while for modifying RDF datasets, SPARQL provides the following update operations: LOAD, CLEAR, DROP, ADD, MOVE, COPY, CREATE, INSERT, and DELETE[8]. A basic component of a SPARQL query is graph pattern matching, which is essentially a form of conjunctive query.

2.3 Ontology Languages

There are different definitions of ontology depending on the context of the discussion. In knowledge acquisition and representation, the following definition from Gruber [54] is very often-cited: "An ontology is an explicit specification of conceptualization". This definition predates the idea of Semantic Web with which the notion of ontology is now popularly associated. In fact, ontology has been a subject of study much earlier than that, e.g., in philosophy.

Gruber's definition became prominent when it was realized that ontology plays an important role in achieving interoperability among knowledge-based systems [97]. In

[8]SPARQL update is a feature of SPARQL 1.1, but not of SPARQL 1.0

this context, ontology's role is enabling knowledge sharing and reuse. That is, ontology became the means by which different agents or systems can communicate. This is possible because every knowledge-based system, explicitly or not, subscribe to their own *conceptualization* about the world, i.e., some abstraction about the world that one wants to represent. Such a conceptualization contains entities and relationships between them, and ontology captures the part of conceptualization shared among the systems. As a result, ontology would contain terms and their definition, which are agreed upon by the parties who use the ontology.

Naturally, for an ontology to be usable by machines, one needs to employ an unambiguous, machine-processable, formal knowledge representation language to spell out the specification. No less importantly, the terms, or *vocabulary*, and their definitions have to be *physically shareable*: an ontology has to be *portable*. This can only be done if the formal language used to write or serialize an ontology has some syntactic object that can be physically referred to, copied, and worked on. A further benefit of such formal languages, especially those based on formal logic, is that the vocabulary definition in an ontology can be checked for logical inconsistency. For some of these languages, there are even algorithms for automated reasoning that have been implemented in widely used tools. W3C Semantic Web standards such as RDF, RDFS, OWL, and RIF are examples for such languages.

In the following, we shall describe some of these languages. Note that RDF as a data model actually already facilitates knowledge representation. However, it has very little semantics and not much inference can be done with it. RDFS, OWL, and RIF build more semantics on top of RDF, and thus, are more powerful than RDF.

2.3.1 RDF Schema (RDFS)

RDF Schema (RDFS) [22] extends RDF with some lightweight knowledge modeling features. Notably, it introduces *class* as a new semantic construct. It also defines

```
rdf:type           rdfs:domain rdfs:Resource ;   rdfs:range rdfs:Class .
rdfs:domain        rdfs:domain rdf:Property ;    rdfs:range rdf:Property .
rdfs:range         rdfs:domain rdf:Property ;    rdfs:range rdf:Property .
rdfs:subPropertyOf rdfs:domain rdf:Property ;    rdfs:range rdf:Property .
rdfs:subClassOf    rdfs:domain rdfs:Class ;      rdfs:range rdfs:Class .
```

Figure 2.5: RDFS axiomatic triples for `rdfs:domain`, `rdfs:range`, `rdfs:subPropertyOf`, and `rdfs:subClassOf`, expressed in Turtle format.

a set of URIs as new vocabulary terms on top of existing RDF vocabulary, defines their semantics, as well as, extends the semantics of some of the RDF vocabulary. The new vocabulary terms represent special classes, special properties, and URIs for purposes such as modeling containers, reification, and extra-logical annotations. In addition to semantic conditions of RDF entailment, RDFS adds several other semantic conditions. In particular, every class, say c, is interpreted as the set of all resources identified by x such that $\langle x, \texttt{rdf:type}, c\rangle$ is in the graph. RDFS semantics also defines the notions of subclasses, subproperties, as well as domain and range of properties. The semantics conditions are easier to explain through entailment rules, and there are 13 of them defined in the RDFS semantics [61].

RDFS semantics asserts that all RDFS axiomatic triples are (implicitly) true in every RDF graph. The ones for the domain, range, subproperty, and subclass properties are given in Fig. 2.5. For example, the first two there state that the domain of `rdf:type` is `rdfs:Resource` and its range is `rdfs:Class`. The remaining (infinitely many) axiomatic triples omitted here are for properties used for RDF reification, containers (e.g., list), and annotations. So, let G' be an RDF graph and G be the graph obtained by adding all RDFS axiomatic triples to G'. The following are most of the RDFS entailment rules; the entailment rules for literals, datatypes, and `rdfs:containerMembershipProperty` are omitted.

(rdfs2) If $\langle s, p, o\rangle, \langle p, \texttt{rdfs:domain}, c\rangle \in G$, then G' entails $\langle s, \texttt{rdf:type}, c\rangle$.

(rdfs3) If $\langle s, p, o\rangle, \langle p, \texttt{rdfs:range}, c\rangle \in G$, then G' entails $\langle o, \texttt{rdf:type}, c\rangle$.

(rdfs4) If $\langle s, p, o\rangle \in G$, then G' entails $\langle s, \texttt{rdf:type}, \texttt{rdfs:Resource}\rangle$ and $\langle o, \texttt{rdf:type}, \texttt{rdfs:Resource}\rangle$.

(rdfs5) If $\langle p, \texttt{rdfs:subPropertyOf}, q\rangle, \langle q, \texttt{rdfs:subPropertyOf}, r\rangle \in G$, then G' entails $\langle p, \texttt{rdfs:subPropertyOf}, r\rangle$.

(rdfs6) If $\langle p, \texttt{rdf:type}, \texttt{rdf:Property}\rangle \in G$, then G' entails $\langle p, \texttt{rdfs:subPropertyOf}, p\rangle$.

(rdfs7) If $\langle p, \texttt{rdfs:subPropertyOf}, q\rangle, \langle s, p, o\rangle \in G$, then G' entails $\langle s, q, o\rangle$.

(rdfs8) If $\langle c, \texttt{rdf:type}, \texttt{rdfs:Class}\rangle \in G$,
then G' entails $\langle c, \texttt{rdfs:subClassOf}, \texttt{rdfs:Resource}\rangle$.

(rdfs9) If $\langle c, \texttt{rdfs:subClassOf}, d\rangle, \langle s, \texttt{rdf:type}, c\rangle \in G$, then G' entails $\langle s, \texttt{rdf:type}, d\rangle$.

(rdfs10) If $\langle c, \texttt{rdf:type}, \texttt{rdfs:Class}\rangle \in G$, then G' entails $\langle c, \texttt{rdfs:subClassOf}, c\rangle$.

(rdfs11) If $\langle c, \texttt{rdfs:subClassOf}, d\rangle, \langle d, \texttt{rdfs:subClassOf}, e\rangle \in G$, then G' entails $\langle c, \texttt{rdfs:subClassOf}, e\rangle$.

From the above rules, we understood that for example, both subclass and subproperty relations are reflexive and transitive (rdfs5, rdfs6, rdfs9, rdfs10). So, with RDFS, one can express class hierarchy and property hierarchy, also reason about class membership of individuals along the hierarchy because it propagates along the hierarchy. One can also infer class membership of an individual from the domain and range of properties.

RDFS also supports some form of metamodeling, e.g., deriving that $\langle \texttt{rdfs:Class}, \texttt{rdf:type}, \texttt{rdfs:Class}\rangle$ is entailed by any graph by applying rdfs3 rule together with only the axiomatic triple $\langle \texttt{rdf:type}, \texttt{rdfs:range}, \texttt{rdfs:Class}\rangle$. This showed that the resource `rdfs:Class` is also a class. Similarly applying rdfs3 rule, the earlier axiomatic triple, and any triple $\langle s, \texttt{rdf:type}, c\rangle$ would allow us to infer $\langle c, \texttt{rdf:type}, \texttt{rdfs:Class}\rangle$, i.e., `rdfs:Class` is the set of all classes.

One thing to note here, which was also demonstrated in the previous paragraph, is that RDFS semantics does not partition the resources into disjoint categories of individuals, classes, and properties. That is, a resource can belong simultaneously in any two or even all of those categories. In fact, the fact that the resource `rdfs:Class` is a member of the set of all classes, which is the resource `rdfs:Class` itself, constitute a well-known Russel's paradox. Fortunately, in practice, one would not typically be interested in this kind of inference.

In terms of serialization, there is no difference between RDFS and RDF. RDFS only adds a set of new URIs whose meaning is as determined by RDFS semantics. Any RDFS document is an RDF document and thus any RDF serialization can be used for sharing RDFS ontology.

2.3.2 The Web Ontology Language (OWL)

Although RDFS does provide us with a way to specify some semantics on our data, it is still very limited. For example, one cannot use RDFS to infer, given two RDF triples stating that Dayton is a city in Ohio and Ohio is a state in the US, that Dayton is a city in the US. To support such an inference, a more expressive language is needed.

Semantic Web provides the Web Ontology Language (OWL) as a language that one can use to write and share ontologies. It defines constructs with a formal semantics for expressing logical assertions, over which, automated reasoning to derive entailments can be done. OWL was first standardized in 2004 [91]. The second version of the language, dubbed OWL 2, which is more expressive than the first, came out later in 2009 [103]. We provide the OWL 2 language definition below and simply refer to it as OWL. Note that our definition below does not provide the definition of *every* logical construct specified in OWL standards [96] because the omitted constructs can be expressed by some combination of the constructs presented here and

in the select few that cannot, they are never used in any of the modeling done in this thesis.

From a theoretical perspective, OWL is roughly a syntactic variant of $\mathcal{SROIQ}(D)$, which belongs to *Description Logics* (*DLs*), a rather prominent family of decidable fragments of first-order logic [6]. We shall use the more concise DL notation to introduce the language below.

A logical statement in an OWL ontology is called *axiom*. An axiom is syntactically formed of atomic symbols, literals and a number of logical constructs. Every atomic symbol is a URI that, unless stated otherwise, are not reserved by OWL. Since we are using DL notation, logical constructs will typically be written using special symbols, though the OWL standards use reserved keywords or URIs. For readability reason, we omit the full namespace of the URIs whenever it is not ambiguous.

An *atomic symbol* is a URI that is either a *class*, an *object property*, a *data property*, a *named individual*, or a *datatype*. A *literal* is either a *plain literal*, which is a simple string value, e.g., "abc", or a *typed literal*, which is a string value that represents some lexical form of some concrete value whose type is the indicated datatype, e.g., "abc"^^datatypeURI where datatypeURI is a datatype URI. OWL also recognize the so-called *anonymous individuals*, which are identified by a local node ID, and not by URI, i.e., analogous to RDF blank nodes. OWL allows anonymous individuals to be used anywhere named individuals can be used. Throughout this thesis, however, we shall only use named individuals.

Note that the same URI occurring in different ontologies always represents the same entity (the same individual if the URI is an individual URI, the same class if it is a class URI, etc.). From atomic symbols and literals, we can additionally define *class expressions*, and *object property expressions*, and *data ranges*.

An *object property expression* is an expression of the form R where R is an object property, or S^-, called the inverse of S, where S is some object property expression.

OWL reserves two special object properties, `owl:topObjectProperty`, denoted with U, and `owl:bottomObjectProperty`, denoted with N, and both have fixed semantics defined below. Likewise, we reserve two special data properties, `owl:topDataProperty` and `owl:bottomDataProperty`, both with fixed semantics defined below.

A *class expression* is an expression of one of the form below. The corresponding OWL abstract syntax [96] is given in Table 2.1.

- A where A is a class, including two predefined classes, `owl:Thing`, denoted by $\top$, and `owl:Nothing`, denoted by $\bot$;
- class complement $\neg C$ where C is a class expression;
- intersection $C_1 \sqcap \cdots \sqcap C_m$ and union $C_1 \sqcup \cdots \sqcup C_m$ with $m \geq 2$ and all $C_1, \ldots, C_m$ themselves class expressions;
- $\{a\}$ where a is a named individual;
- self-restriction $\exists R.\mathsf{Self}$ where R is an object property expression;
- existential restriction $\exists R.C$, universal restriction $\forall R.C$, and number restrictions $(\geqslant n\ R.C)$, $(\leqslant n\ R.C)$, $(=n\ R.C)$ where n is a natural number, R is an object property expression and C is a class expression;
- existential restriction $\exists P.D$, universal restriction $\forall P.D$, and number restrictions $(\geqslant n\ P.D)$, $(\leqslant n\ P.D)$, $(=n\ P.D)$ where n is a natural number, P is a data property and D is a data range.

A *data range* is recursively defined as a k-ary datatype, or an intersection of data range, or a union of data range, or a complement of data range, or an enumeration of literals, or a datatype restriction. Throughout this thesis, however, we will only use unary datatypes, which are sufficiently represented as URIs. We refer the reader to OWL standards [95, 96] for more details on the syntax and semantics of data ranges.

Having defined class expressions and property expressions above, we now define OWL axioms. An *axiom* is either

- a *subclass axiom* (i.e., *class inclusion*) of the form $C \sqsubseteq D$; or

DL Notation	OWL Abstract Syntax
$\neg C$	`ObjectComplementOf`(C)
$C_1 \sqcap C_2 \sqcap \cdots \sqcap C_m$	`ObjectIntersectionOf`$(C_1\ C_2\ \ldots\ C_m)$
$C_1 \sqcup C_2 \sqcup \cdots \sqcup C_m$	`ObjectUnionOfOf`$(C_1\ C_2\ \ldots\ C_m)$
$\{a\}$	`ObjectOneOf`(a)
$\exists R.\mathsf{Self}$	`ObjectHasSelf`(R)
$\exists R.C$	`ObjectSomeValuesFrom`$(R\ C)$
$\forall R.C$	`ObjectAllValuesFrom`$(R\ C)$
$(\geqslant n\ R.C)$	`ObjectMinCardinality`$(n\ R\ C)$
$(\leqslant n\ R.C)$	`ObjectMaxCardinality`$(n\ R\ C)$
$(=n\ R.C)$	`ObjectExactCardinality`$(n\ R\ C)$
$\exists P.D$	`DataSomeValuesFrom`$(P\ D)$
$\forall P.D$	`DataAllValuesFrom`$(P\ D)$
$(\geqslant n\ P.D)$	`DataMinCardinality`$(n\ P\ D)$
$(\leqslant n\ P.D)$	`DataMaxCardinality`$(n\ P\ D)$
$(=n\ P.D)$	`DataExactCardinality`$(n\ P\ D)$

Table 2.1: OWL Notation for Class Expression

- a *class equivalence* of the form $C \equiv D$; or
- a *class assertion* of the form $C(a)$; or
- a *property assertion* of the form $R(a, b)$; or
- an expression of the form $R_1 \circ \cdots \circ R_m \sqsubseteq R$ where $m \geq 1$ — if $m = 1$, we call it *subproperty axiom*, otherwise it is called *property chain axiom*; or
- an *individual equality assertion* $a_1 = a_2 = \cdots = a_n$ with $n \geq 2$; or
- an *individual inequality assertion* $\mathsf{alldifferent}(a_1, \ldots, a_n)$ with $n \geq 2$; or
- a *class disjointness assertion* $\mathsf{alldisjoint}(C_1, \ldots, C_n)$ with $n \geq 2$.

where C, $C_1, \ldots, C_n$, and D are any kind of class exressions; $a, a_1, \ldots, a_n$ are individuals; b is either a named individual or a literal; R is an object property expression if b is a named individual, otherwise (i.e., when b is a literal), R is a data property; and $R_1, \ldots, R_m$ and S are object property expressions. Note that there are other kinds of

DL Notation	OWL Abstract Syntax
$C \sqsubseteq D$	$\texttt{SubClassOf}(C\ D)$
$C \equiv D$	$\texttt{EquivalentClasses}(C\ D)$
$C(a)$	$\texttt{ClassAssertion}(C\ a)$
$R(a,b)$	$\texttt{ObjectPropertyAssertion}(R\ a\ b)$ if b is an individual
	$\texttt{DataPropertyAssertion}(R\ a\ b)$ if b is a literal
$R_1 \sqsubseteq R$	$\texttt{SubObjectPropertyOf}(R_1\ R)$
$R_1 \circ \cdots \circ R_m \sqsubseteq R$	$\texttt{SubObjectPropertyExpression}($
	$\quad\texttt{ObjectPropertyChain}(R_1\ \ldots\ R_m)\ R)$
$a_1 = \cdots = a_n$	$\texttt{SameIndividual}(a_1\ \ldots\ a_n)$
$\mathsf{alldifferent}(a_1, \ldots, a_n)$	$\texttt{DifferentIndividuals}(a_1\ \ldots\ a_n)$
$\mathsf{alldisjoint}(C_1, \ldots, C_n)$	$\texttt{DisjointClasses}(C_1\ \ldots\ C_n)$

Table 2.2: OWL Axioms

axioms in OWL that we omit here because either they can be expressed using the axioms we defined above or we do not use them throughout this thesis.

Semantics of axioms and expressions is defined in terms of *interpretations* and we mostly follow OWL 2 Direct Semantics [95] with some simplifications. An interpretation is a tuple $\mathcal{I} = (\Delta_\mathcal{I}, \Delta_\mathcal{D}, \cdot^\mathcal{I}, \cdot^\mathcal{D})$ where

- $\Delta_\mathcal{I}$ and $\Delta_\mathcal{D}$ are pairwise disjoint nonempty sets called *object domain* and *data domain*, respectively;
- $\cdot^\mathcal{I}$ is an *object interpetation* function such that:
  - it maps every named individual a to an element $a^\mathcal{I} \in \Delta_\mathcal{I}$, every class expression C to a set $C^\mathcal{I} \subseteq \Delta_\mathcal{I}$, and every object property expression R to a binary relation $R^\mathcal{I} \subseteq \Delta_\mathcal{I} \times \Delta_\mathcal{I}$;
  - $\top^\mathcal{I} = \Delta_\mathcal{I}$; $\bot^\mathcal{I} = \emptyset$; $U^\mathcal{I} = \Delta_\mathcal{I} \times \Delta_\mathcal{I}$; $N^\mathcal{I} = \emptyset$;
  - $(C_1 \sqcap \cdots \sqcap C_m)^\mathcal{I} = C_1^\mathcal{I} \cap \cdots \cap C_m^\mathcal{I}$; $(C_1 \sqcup \cdots \sqcup C_m)^\mathcal{I} = C_1^\mathcal{I} \cup \cdots \cup C_m^\mathcal{I}$;
  - $\{a\}^\mathcal{I} = \{a^\mathcal{I}\}$;
  - $(\exists R.\mathsf{Self})^\mathcal{I} = \{x \in \Delta_\mathcal{I} \mid \langle x, x\rangle \in R^\mathcal{I}\}$

- $(\exists R.C)^{\mathcal{I}} = \{x \in \Delta_{\mathcal{I}} \mid \exists y\colon \langle x, y\rangle \in R^{\mathcal{I}} \wedge y \in C^{\mathcal{I}}\}$ if R is an object property expression;
- $(\forall R.C)^{\mathcal{I}} = \{x \in \Delta_{\mathcal{I}} \mid \forall y\colon \langle x, y\rangle \in R^{\mathcal{I}} \rightarrow y \in C^{\mathcal{I}}\}$ if R is an object property expression;
- $(\geqslant n\ R.C)^{\mathcal{I}} = \{x \in \Delta_{\mathcal{I}} \mid |\{y \mid \langle x, y\rangle \in R^{\mathcal{I}} \wedge y \in C^{\mathcal{I}}\}| \geqslant n\}$
- $(\leqslant n\ R.C)^{\mathcal{I}} = \{x \in \Delta_{\mathcal{I}} \mid |\{y \mid \langle x, y\rangle \in R^{\mathcal{I}} \wedge y \in C^{\mathcal{I}}\}| \leqslant n\}$
- $(=n\ R.C)^{\mathcal{I}} = \{x \in \Delta_{\mathcal{I}} \mid |\{y \mid \langle x, y\rangle \in R^{\mathcal{I}} \wedge y \in C^{\mathcal{I}}\}| = n\}$

- $\cdot^{\mathcal{D}}$ is a *data interpretation* function such that:
  - it maps every literal l to an element $v^{\mathcal{D}} \in \Delta_{\mathcal{D}}$ such that v is the data value of l, every datatype T to a subset $T^{\mathcal{D}} \subseteq \Delta_{\mathcal{D}}$, and every data property R to a binary relation $R^{\mathcal{D}} \subseteq \Delta_{\mathcal{I}} \times \Delta_{\mathcal{D}}$;
  - $(\texttt{rdfs:Literal})^{\mathcal{D}} = \Delta_{\mathcal{D}}$;
  - $(\texttt{owl:topDataProperty})^{\mathcal{D}} = \Delta_{\mathcal{I}} \times \Delta_{\mathcal{D}}$; $(\texttt{owl:bottomDataProperty})^{\mathcal{D}} = \emptyset$;
  - $(\exists R.T)^{\mathcal{D}} = \{x \in \Delta_{\mathcal{I}} \mid \exists y\colon \langle x, y\rangle \in R^{\mathcal{D}} \wedge y \in T^{\mathcal{D}}\}$ if R is a data property;
  - $(\forall R.T)^{\mathcal{D}} = \{x \in \Delta_{\mathcal{I}} \mid \forall y\colon \langle x, y\rangle \in R^{\mathcal{D}} \rightarrow y \in T^{\mathcal{D}}\}$ if R is a data property;

Note that each datatype represents set of data values such as strings or integers. Examples of datatypes supported in OWL include numbers (e.g., `owl:real`, `xsd:decimal`, etc.), strings (e.g., `rdf:PlainLiteral`, `xsd:string`, etc.), time instants (e.g., `xsd:dateTime`, etc.), etc. Furthermore, each literal represents an actual data value within a particular datatype as indicated by the syntactic appearance of the literal itself.

An intepretation $\mathcal{I}$ *satisfies* (is a *model* of) an OWL axiom:

- $C \sqsubseteq D$ if $C^{\mathcal{I}} \subseteq D^{\mathcal{I}}$;
- $C \equiv D$ if $C^{\mathcal{I}} = D^{\mathcal{I}}$;
- $C(a)$ if $a^{\mathcal{I}} \in C^{\mathcal{I}}$;
- $R(a, b)$ if $\langle a^{\mathcal{I}}, b^{\mathcal{I}}\rangle \in R^{\mathcal{I}}$ for object property expression R;
- $P(a, b)$ if $\langle a^{\mathcal{I}}, b^{\mathcal{D}}\rangle \in P^{\mathcal{D}}$ for data property P;

- $R_1 \sqsubseteq R$ if $R_1^{\mathcal{I}} \subseteq R^{\mathcal{I}}$;
- $R_1 \circ \cdots \circ R_m \sqsubseteq R$ if the relation composition $R_1^{\mathcal{I}} \circ \cdots \circ R_m^{\mathcal{I}} \subseteq R^{\mathcal{I}}$
- $a_1 = \cdots = a_n$ if $a_1^{\mathcal{I}} = \cdots = a_n^{\mathcal{I}}$;
- $\mathsf{alldifferent}(a_1, \ldots, a_n)$ if $a_i \neq a_j$ for all $1 \leq i < j \leq n$;
- $\mathsf{alldisjoint}(C_1, \ldots, C_n)$ if $C_i \sqcap C_j \sqsubseteq \bot$ for all $1 \leq i < j \leq n$.

An OWL ontology is uniquely identified by an *ontology URI* and possibly, *version URI*. These URIs allow the ontology documents to be accessible on the Web. We define the *name* of an ontology $\mathcal{O}$ to be its version URI if both its ontology URI and its version URI are defined, and its ontology URI, otherwise.

An *OWL ontology* is a set of OWL axioms. In addition, an OWL ontology $\mathcal{O}$ may import other ontologies by declaring the name of the other ontologies in the import statements in $\mathcal{O}$. To this end, we say that $\mathcal{O}$ *directly imports* an ontology $\mathcal{O}'$ if the name of $\mathcal{O}'$ appears in an import statement in $\mathcal{O}$. Then, we say that $\mathcal{O}$ *imports* $\mathcal{O}'$ if either $\mathcal{O}$ directly imports $\mathcal{O}'$, or if there exists an ontology $\mathcal{O}''$ such that $\mathcal{O}$ directly imports $\mathcal{O}''$ and $\mathcal{O}''$ imports $\mathcal{O}'$. The *import closure* $\mathit{Imp}(\mathcal{O})$ contains $\mathcal{O}$ and all ontologies $\mathcal{O}'$ that $\mathcal{O}$ imports.

The *axiom closure* of $\mathcal{O}$ is then the smallest set of OWL axiom α such that $\alpha \in \mathcal{O}'$ for some $\mathcal{O}' \in \mathit{Imp}(\mathcal{O})$. An interpretation $\mathcal{I}$ *satisfies* (is a *model* of) an OWL ontology $\mathcal{O}$ if it satisfies all axioms in the axiom closure of $\mathcal{O}$. An axiom α (not necessarily in $\mathcal{O}$) is *entailed* by $\mathcal{O}$, denoted $\mathcal{O} \models \alpha$, if every model of $\mathcal{O}$ satisfies α. Entailment of axioms is straightforwardly extended to sets of axioms.

Remark on domain and range restrictions of properties. Since properties are semantically a binary relation over the object domain, when a property is defined within a particular pattern, we often need to assert classes that cover the domain and range of that property. OWL actually also allows axioms of the form (in OWL

functional syntax) where R is an object property, C a class, P a data property, and D a data range:

- `ObjectPropertyDomain`(R C), semantically equivalent to $\exists R.\top \sqsubseteq C$;
- `DataPropertyDomain`(P C), semantically equivalent to $\exists P.\texttt{rdfs:Literal} \sqsubseteq C$;
- `ObjectPropertyRange`(R C), semantically equivalent to $\top \sqsubseteq \forall R.C$;
- `DataPropertyRange`(P D), semantically equivalent to $\top \sqsubseteq \forall P.D$.

The first two axioms are called *unguarded domain restrictions*, while the latter two are called *unguarded range restrictions*. The term unguarded comes from the way the restrictions are written in a rule form (see Section 2.3.3). For example, the first axiom above is equivalent to the rule $R(x, y) \rightarrow C(x)$. Here, $R(x, y)$ is not "guarded" because there is no other condition in the body of the rule.

For the purpose of reusability of patterns, the unguarded domain/range restrictions force a very strong ontological commitment: if a data instance p is such that $R(p, q)$ is asserted or inferred for some q, then unguarded domain restriction would force p to belong to A regardless whether q belongs to B or not – the problem for unguarded range restriction is similar. As an alternative, we will frequently use a *guarded domain restriction* which, in the context of the earlier example, will be of the form $R(x, y) \wedge B(y) \rightarrow A(x)$ (or $\exists R.B \sqsubseteq A$ in DL notation), and a *guarded range restriction* which will be of the form $R(x, y) \wedge A(x) \rightarrow B(y)$ ($\exists R^{-}.A \sqsubseteq B$, or equivalently, $A \sqsubseteq \forall R.B$ in DL notation). It is easy to see that if R is used in some other patterns, two individuals connected through R need not be forced to belong A and B, respectively.

2.3.3 Datalog

If OWL is based on description logics, the other comparable standard provided by Semantic Web to write a knowledge base is the Rule Interchange Format (RIF) [74].

RIF is divided into several dialects. RIF-Core is a dialect of RIF that corresponds to Datalog, i.e., the language of definite Horn rules without function symbols interpreted under standard first-order semantics [16]. RIF Basic Logic Dialect (RIF-BLD) corresponds to the language of definite Horn rules with equality interpreted under standard first-order semantics [17]. RIF Production Rule Dialect (RIF-PRD) corresponds to production rules, i.e., rules with action [41]. Furthermore, RIF provides a general framework for developing other rule dialects via RIF Framework for Logic Dialects (RIF-FLD) [18].

The only rule dialect that will be of some use in modeling for our needs is Datalog and its generalization, called Datalog +/- or tuple-generating dependency [1]. So we present its definition next. For translating Datalog to the RIF syntax, the reader can consult the RIF-Core specification [16].

An *atom* is an expression of the form either $C(x)$, $R(x, y)$, or $x = y$ where C is a class name, R is a property name, and x, y are either named individuals or variables. A *rule* is a statement of the form $B_1 \wedge \cdots \wedge B_m \rightarrow H$ where $m \geq 0$, and H and all B_i's are atoms. The conjunction of B_i's is called the *body*, while H is called the *head* of the rule. If $m = 0$, we say that the rule is a *fact*. A rule/fact is *ground* if no variable appears in it. We assume that the rule is *safe*, i.e., each variable occurring in the head occurs somewhere in the body. We sometimes use an expression of the form $B_1 \wedge \cdots \wedge B_m \rightarrow H_1 \wedge \cdots \wedge H_n$ as an abbreviation of a set of n rules: $B_1 \wedge \cdots \wedge B_m \rightarrow H_1$, $\ldots$, $B_1 \wedge \cdots \wedge B_m \rightarrow H_n$. Semantics of a rule is a first-order implication in which all variables in the rule are universally quantified over individuals in the object domain and equality is interpreted as the standard equality relation over the object domain (i.e., $x = y$ iff x and y are the same element of the object domain).

A tuple-generating dependency (tgd) is a first-order logic formula of the form

$$\forall \vec{x} \Big((P_1(\vec{x}_1) \wedge \cdots \wedge P_n(\vec{x}_n)) \rightarrow \exists \vec{y} (T_1(\vec{y}_1) \wedge \cdots \wedge T_k(\vec{y}_k)) \Big)$$

where $\vec{y}$ contains variables in $\vec{y}_1, \ldots, \vec{y}_k$ that are existentially quantified, and $\vec{x}$ contains the rest of the variables (and thus are universally quantified). An equality-generating dependency (egd) is a first-order logic formula of the form

$$\forall \vec{x} \Big((P_1(\vec{x}_1) \wedge \cdots \wedge P_n(\vec{x}_n)) \rightarrow x_i^1 = x_j^1 \wedge \cdots \wedge x_i^k = x_j^k \Big)$$

where all $x_i^1, x_j^1, \ldots, x_i^k, x_j^k$ are variables in $\vec{x}$. Both tgd and egd are interpreted as first-order implication. Also, when writing them, the universal quantifier of a tgd and egd is often omitted, and in particular, tgd would appear similar to a Datalog rule, but with a rule head that contains existentially quantified variables.

3 Data Integration

In this chapter, we provide a more detailed overview of data integration in the relational database, ontology-based data integration, and linked data integration. We end the chapter with a discussion regarding the strengths and weaknesses of using ontology for data integration.

3.1 Data Integration in Databases

Data integration is a well-studied problem in relational databases, initially presented by Landers et al. with their Multi-Base System [87] almost 35 years ago [42]. The objective of any data integration is to allow users to obtain data from different data sources through only a single query interface. In general, those different data sources are typically developed and operated independently, and thus heterogeneous in nature. A user would pose a query to a *global schema*, which acts as the (single) query interface, and then the system would use data specified according to *local schemas* to return an answer, if any.

3.1.1 Formal Definition of Data Integration

Formally [2, 42, 89], a data integration system D consists of a global schema G, a local schema S, and a mapping M between G and S, which is a set of assertions defined later. For this definition, we assume a relational model. So, the global schema G consists of relation symbols $G_1, \ldots, G_\ell$ with fixed arities. Similarly, S consists of relation symbols $S_1, \ldots, S_m$, and furthermore, the local data sources are predefined. In a more general

setting, the global (resp. local) schema G (resp. S) is expressed in a language $\mathcal{L}_G$ (resp. $\mathcal{L}_S$) over a signature $\mathcal{A}_G$ (resp. $\mathcal{A}_S$), and depending on the language, the global and local schemas may also include a set of constraints expressed over the relation symbols in their signatures. This definition does not preclude multiple data sources because each source itself consists of relations and we can simply consider them together as one local schema.

A (*conjunctive*) *query* over a schema T is an expression of the form

$$q(x_1, \ldots, x_n) \text{ :- } A_1(\vec{u}_1), \ldots, A_k(\vec{u}_k)$$

where $A_1, \ldots, A_k$ are relations in T, $x_1, \ldots, x_n$ are *distinguished* variables, and $\vec{u}_1, \ldots, \vec{u}_k$ are vectors of constants and variables such that each distinguished variable x_i occurs in some $\vec{u}_j$. Non-distinguished variables in $\vec{u}_1, \ldots, \vec{u}_k$ are called *existential* variables. In the query expression above, $q(x_1, \ldots, x_n)$ is called the *head*, while $A_1(\vec{u}_1), \ldots, A_k(\vec{u}_k)$ the *body* of the query.

The mapping M between the global schema G and local schema S consists of assertions, which can be categorized into three classes: Global-as-View (GAV), Local-as-View (LAV), and Global-and-Local-as-View (GLAV). A *GAV mapping* is an expression of the form $G_i(\vec{x}) \supseteq q^S(\vec{x}, \vec{y})$ or $G(\vec{x}) = q^S(\vec{x}, \vec{y})$ where G_i is a relation in the global schema G, $\vec{x}$ is a vector of variables, q^S is a query over relations in the local schema S whose distinguished variables are precisely in $\vec{x}$ and existential variables are in $\vec{y}$. An *LAV mapping* is an expression of the form $S_i(\vec{x}) \subseteq q^G(\vec{x}, \vec{y})$ or $S_i(\vec{x}) = q^G(\vec{x}, \vec{y})$ where S_i is a relation in the local schema S, $\vec{x}$ is a vector of variables, q^G is a query over relations in the global schema G whose distinguished variables are precisely in $\vec{x}$ and existential variables are in $\vec{y}$ A *GLAV mapping* is essentially a combination of GAV and LAV mappings, that is, an expression of the form $q^S(\vec{x}, \vec{y}) \subseteq q^G(\vec{x}, \vec{z})$ or $q^S(\vec{x}, \vec{y}) = q^G(\vec{x}, \vec{z})$. Here, q^S and q^G are query over relations in resp. the local schema S and the global schema G. The vector $\vec{x}$ consists of the distinguished variables of

both q^S and q^G, while the vectors $\vec{y}$ and $\vec{z}$ consists of existential variables in resp. q^S and q^G.

Semantically, a data integration system can be understood as a set of first-order formulas as follows. First, since the data sources are predefined, the relations in the local schema are instantiated. To this instance of the local schema, we associate a set of ground first-order atomic formulas with $\mathcal{A}_S$ as the declared signature. Constraints in the local schema, such as key constraints and functional dependencies, are associated with tuple-generating dependencies (tgds) and equality-generating dependencies (egds) [1], defined over the signature $\mathcal{A}_S$. Meanwhile, the global schema contains no real data, so we only need to associate tgds and egds to the constraints in the global schema, defined over the signature $\mathcal{A}_G$. Finally, each of the mapping assertion is associated with tgds as follows:

- GAV mapping $G_i(\vec{x}) \supseteq q^S(\vec{x}, \vec{y})$ with tgd $\forall\vec{x}(\exists\vec{y}.q^S(\vec{x}, \vec{y}) \rightarrow G_i(\vec{x}))$;
- GAV mapping $G_i(\vec{x}) = q^S(\vec{x}, \vec{y})$ with tgd $\forall\vec{x}(\exists\vec{y}.q^S(\vec{x}, \vec{y}) \leftrightarrow G_i(\vec{x}))$;
- LAV mapping $S_i(\vec{x}) \subseteq q^G(\vec{x}, \vec{y})$ with tgd $\forall\vec{x}(S_i(\vec{x}) \rightarrow \exists\vec{y}.q^G(\vec{x}, \vec{y}))$;
- LAV mapping $S_i(\vec{x}) = q^G(\vec{x}, \vec{y})$ with tgd $\forall\vec{x}(S_i(\vec{x}) \leftrightarrow \exists\vec{y}.q^G(\vec{x}, \vec{y}))$;
- GLAV mapping $q^S(\vec{x}, \vec{y}) \subseteq q^G(\vec{x}, \vec{z})$ with tgd $\forall\vec{x}(\exists\vec{y}.q^S(\vec{x}, \vec{y}) \rightarrow \exists\vec{z}.q^G(\vec{x}, \vec{z}))$;
- GLAV mapping $q^S(\vec{x}, \vec{y}) = q^G(\vec{x}, \vec{z})$ with tgd $\forall\vec{x}(\exists\vec{y}.q^S(\vec{x}, \vec{y}) \leftrightarrow \exists\vec{z}.q^G(\vec{x}, \vec{z}))$.

Clearly, from a data integration system D one can obtain a first-order theory $\mathcal{T}_D$ representing the global schema, local data sources and the mapping.

A query $q(x_1, \ldots, x_n)$ to the data integration system D where $x_1, \ldots, x_n$ are the distinguished variables is actually a query over the global schema G. A tuple $\langle v(x_1), \ldots, v(x_n)\rangle$ for some assignment v of variables to constants occurring in D is an answer to q if $q(v(x_1), \ldots, v(x_n))$ logically follows from the $\mathcal{T}_D$.

The benefit of GAV mapping is of simplicity in query processing: a query to the global schema can simply be unfolded into queries to the local schemas according to

the mapping. On the other hand, adding or removing data sources may imply the need to reexamine and modify the mapping: in GAV mapping, the global schema is defined using views over the local schema, hence if a new source is added, no data from the new source can be queried unless the mapping is modified to include the new source.

Meanwhile, we cannot as easily obtain a query rewriting for an LAV mapping, though adding and removing data sources are simpler: we can simply add an appropriate new mapping to define a source or remove the mapping that correspond to the removed source. Regarding GLAV mapping, by following an approach from Calì et al. [24], one can actually transform a data integration system with GLAV mappings into one with GAV mappings by essentially adding inclusion dependencies to the global schema. So, GLAV mapping has the same benefit as GAV mapping.

Observe that although the formal model of data integration above was defined for a relational model, it can be employed for other models too. Furthermore, the languages used to express the global schema, the local schema, and the mapping can be all different.

3.1.2 Architecture of Data Integration Systems

According to the literature, a majority of data integration systems can be categorized into two architectural variants: *mediator*-based data integration, and *data warehousing*, both of which can be succinctly described by Fig. 3.1. In both variants, the local schema lies on the data sources, while the global schema lies on the mediator or data warehouse layer. Here, both variants differ on the kinds of components operating on top of the data sources upward.

In a mediator-based integration [89, 124, 135], which is also called *virtual* data integration, users interact with a *mediator* component using the global schema. This mediator component's task is only to mediate: it does not store any data for answering

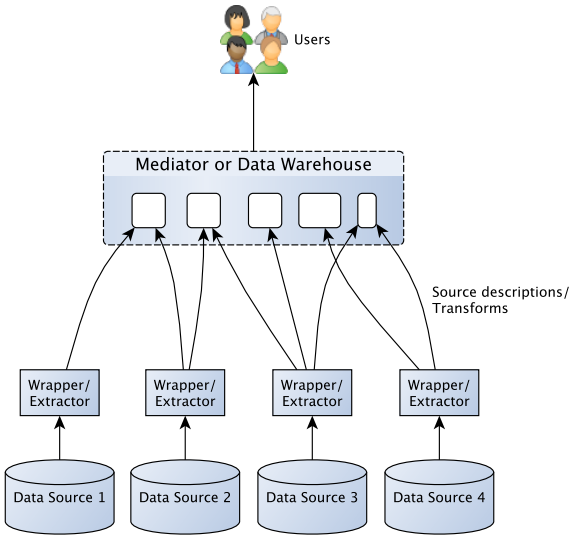


Figure 3.1: Data Integration Architecture (this author's version of Doan et al.'s Fig. 1.4 [42]). The arrows refer to the direction of data flow.

the input queries. Rather, the mediator rewrites the input query into appropriate queries over the local schemas of each data source by utilizing the schema mappings contained in the source descriptions. The *wrappers*' task is then to send the translated queries to the actual data sources, receive the answers from them, and return them back to the mediator, possibly after applying some basic transformation.

In contrast to the mediator-based data integration, the warehousing approach considers the global schema to be materialized [44, 78]. More precisely, instead of a mediator, the topmost layer of the architecture contains a *data warehouse*, which does not only house the global schema as a logical schema that users may use to formulate queries, but more than that, it actually houses the global schema *physically* with an underlying database instance. To answer an input query, the warehouse would not be connected at all to the data sources since all the data it needs to answer the query is presumably already in the warehouse. Periodical updates are done to the warehouse in separate processes by employing Extract-Transform-Load programs, which, based on the schema mapping, extract data from the data sources, perform some, possibly

very complicated, transformations to the data, such as cleaning, aggregation, joins, etc., and then load the data into the warehouse.

In addition to the two major variants above, people have also considered a peer-to-peer data integration approach [25, 59]. In this approach, one would do away with additional components such as mediators or warehouses. Rather, in this approach, input queries can be posed directly to any data sources, each of which is accompanied with descriptions of the other data sources. The integration is achieved by allowing each data source to call/query the other data sources to obtain complete answers to the input queries.

Regardless of the architecture choice for data integration, data integration using relational database schema suffers from the following problems [100]. Firstly, database schema typically do not contain sufficient formal description of the conceptual notions they represent in the database. This is due to the limitation in the expressive power of the language traditionally used for relational model: a schema is given as a set of relations, i.e., tables, each of which further contains a list of columns, the column types and possibly some additional constraints intended to ensure the consistency of the database instances. Although semantics of the conceptual notions may be modeled during the design phase, this is lost when translated into a physical model of the database. Secondly, database schema model specific physical data models, and hence, are not suitable for sharing, reuse and extending, which impairs their flexibility. Thirdly, database schema are typically developed and maintained centrally. This imposes a single world-view, which may not be acceptable to every participant of the integration. Some of these problems have been realized in the literature and motivated the development of ontology-based data integration described in the next section.

3.2 Ontology-based Data Integration

Although research in data integration originated from the database community, it soon became apparent that data integration is a fertile application area for ontology research. Intuitively, database schema can be thought of as a specification of some conceptualization that is focused on the way data are physically stored in a database. As such, this fits Gruber [54]'s definition of ontology. In fact, Gruber made this claim and described a system, called ONTOLINGUA, to back the claim, both in the same aforementioned paper. Since then, the similarities between database schema and ontologies have led to a significant body of work in ontology-based data integration (OBDI), as pointed out by a number of surveys [71, 73, 99, 133, 137].

From an architectural perspective, we find throughout the literature that an ontology-based data integration (OBDI) system resembles Fig. 3.1 whereby (one or more) ontologies hold the role of global schema. Uschold et al. [125] elaborated on some of the similarities and differences between (relational) database schema and ontologies as follows.

(1) First, from a formal language perspective, both database schema and ontologies specify a list of terms that act as a vocabulary used in the application at hand. Furthermore, both are essentially fragments of first-order predicate logic.

(2) Ontologies, however, are designed with the purpose of improving interoperability and knowledge sharing, unlike database schema whose primary purpose is to act as a structure on which a set of instances are organized in a single database, i.e., no consideration toward interoperability with other databases.

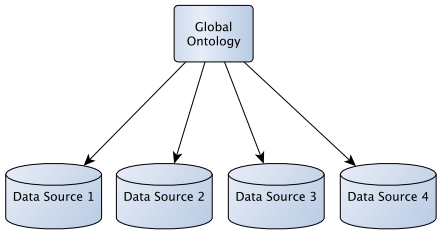
(3) Due to the difference in purpose above, the role of constraints (or axioms in ontologies) differ in both. Constraints in databases are specified over the vocabulary to ensure the integrity of the data. These constraints may express some meaning of the vocabulary terms, but this is of minor importance. On the other hand,

axioms in ontology are specified primarily to constrain the possible interpretations of the vocabulary terms to facilitate (automated) reasoning to infer implicit knowledge.

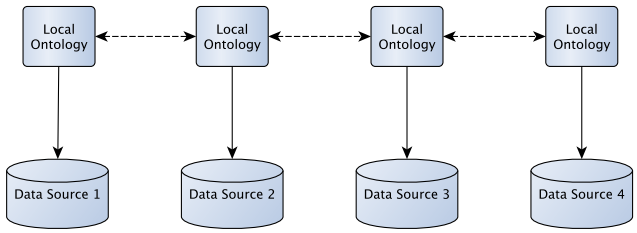
(4) Database engines are highly specialized for query answering, reasoning with (database) views, and ensuring integrity on the data, whereas reasoners for ontologies are mainly used for inferring new knowledge, including taxonomic reasoning, and checking the logical consistency of the ontology even without the presence of instance data.

From the similarities and differences between database schema and ontologies above, we can glean at least two advantages offered by ontologies. First, the meaning of vocabulary terms is more explicit and precisely expressed in ontologies than in database schema. Second, database schema would mostly consist of specific, application-dependent vocabulary terms, which are difficult to reuse in other applications, in contrast to ontologies whose vocabulary terms tend to be much less application-dependent. Both of these are immensely helpful in overcoming the lack of interoperability and bridging semantic heterogeneity. As claimed by Uschold et al. [125],

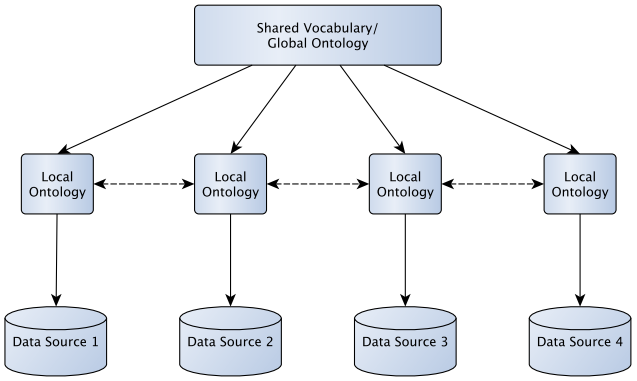
> *"The **less** the following things are true:*
>
> - *there is widespread agreement about the meaning of a term, and the syntax for expressing it, and*
> - *all the software is built by humans who correctly embed the agreed meaning of the term, and*
> - *all the databases and Web pages and applications use the term in the agreed way,*
>
> *then the **more** necessary it is to:*



(a) Single ontology approach



(b) Multiple ontologies approach



(c) Hybrid approach

Figure 3.2: Three ways ontologies are used in OBDI (this author's version of Wache et al.'s Fig. 1[133]). Regular arrows indicate where vocabulary is used. Dotted arrows indicate mapping direction.

- *have an explicit formal declarative semantics of the term that the machine can process to interpret the meaning of that term."*

The distinction between mediator-based integration and data warehousing turns out to be of secondary importance, as OBDI can be deployed as either of them. More interesting to examine is how ontologies are actually used in an OBDI system. Wache et al. [133] identified three ways ontologies can used as the global schema in an OBDI, namely *single ontology approaches*, *multiple ontologies approaches*, and *hybrid approaches*, all of which are illustrated in Fig. 3.2.

In the single ontology approach, one global ontology is employed to provide the set of vocabulary shared by all of the data sources. Here, the global ontology can be a domain ontology, or a combination of modules from a potentially large, monolithic ontology. As examples of this approach, the survey mentioned the SIMS [4] and ON-

TOLINGUA [54] systems. It also mentioned Carnot [35] – which actually employed the foundational ontology Cyc[1] as the global ontology of the system – albeit without explicitly classifying it as a single ontology approach. The single ontology approach is more suitable when all data sources provide almost the same view of the domain. Unfortunately, this ideal situation very rarely happens. Even within the same domain, independently developed data sources almost never have exactly the same local schema, and thus, finding minimal ontological commitments [55] that everybody is willing to agree upon is generally a difficult task, even more so if the global ontology is expected to be large and overarching. Also, adding new data source to the integration is complicated: it may need to adjust its local schema to conform to the global ontology, and this adjustment may be complex and costly.

In the multiple ontology approach, each data source is described using its own ontology acting as the local schema with OBSERVER [92] mentioned as an example. By describing local ontologies separately for each data source, we avoid the need to have a single global ontology that everybody involved in the integration agree upon. Also, as each local ontology is independently developed, adding and removing a data source is easier than in the single ontology approach. However, the lack of common vocabulary makes it difficult to use from the users' perspective. To bridge the semantic heterogeneity, inter-ontology mappings are defined among the local ontologies, though creating and managing such mappings are not always straightforward, because one has to consider different conceptualizations on a domain, e.g., different levels of granularity, which may necessitate complex mappings.

The hybrid approach, in the meantime, combines both of the previous approaches by having one global shared vocabulary or ontology containing only basic terms. These basic terms are used by local ontologies to define more specific notions. This approach allows the addition (and removal) of new data sources almost as easily as

[1]This is a huge, foundational ontology, which has been very well-known from the 80s. Its current incarnation is called OpenCyc. Further details are available http://www.cycfoundation.org

the multiple ontology approach. The use of a shared vocabulary also improves the interoperability among the local ontologies. One potential drawback of this approach is that directly reusing existing domain or upper ontologies for the shared vocabulary may be difficult since such existing ontologies may make ontological commitments that are not accepted by the participant of the data integration.

The strength of the hybrid approach makes it rather appealing as a choice in deploying ontologies within OBDI. This is apparent in the literature as the majority of OBDI application took this approach, particularly from the time following the survey. The survey mentioned COIN [52], MECOTA [132], and BUSTER [130] as examples of the hybrid approach. Beyond those examples, we found that the hybrid approach being employed in various application domains [8, 32, 33, 46, 47, 66, 90, 94, 102, 107, 116, 123, 128, 136], although the single ontology approach still found applications in some domains, e.g., life sciences [93, 106, 134].

In addition to the survey by Wache et al. [133], two surveys by Kalfoglou et al. [73] and Noy [99] are also notable and related to semantic integration, though with rather different emphasis. Both surveys were more focused on the body of work concerning (inter-)ontology mappings, alignment, and other closely related problems. Although these problems are certainly related to OBDI, neither survey focused on different ways ontologies can be employed in an OBDI system. Meanwhile, the survey by Buccella et al. [23] compared 11 systems and methodologies on how they use ontologies for integration and how geographical features within the ontologies are taken into account in the process. This survey also classified the compared the systems and methodologies in the categorization from Wache et al. [133], and most of the proposed systems actually took the hybrid approach. Finally, another survey worth mentioning is from Izza [71]. This survey provided an overview of syntactic and semantic integration in industrial information systems whereby data integration is only one among different kinds of integration considered in enterprise environment – the others being

application integration and business process integration. Interestingly, there was no mention of ontology-based data integration in this survey: both ontologies and data integration were mentioned, but not in the same context. In this survey, semantic integration using ontologies was reviewed in the context of (web) service integration, i.e., the ontologies are used to describe services, rather than data.

Going through the literature, we can make the following observations regarding the role of ontologies as the global schema of a data integration system.

- As confirmed in the systematic study conducted by [102], ontologies are highly important for semantic interoperability in a data integration system. The fact that ontologies were chosen to act as the global schema in many examples above also supports this observation. The use of ontologies brings in strong, machine readable semantics that leads to conceptual clarity and a common modeling basis. These enable different data-source-specific conceptualizations to "speak" between each other, thus realizing semantic interoperability.
- It is evident from the examples of applications cited above that the ontologies employed as the global schema are either an existing domain ontology or foundational ontology. These ontologies are typically very large, containing up to thousands of classes and properties. Moreover, they may contain complex and tightly-coupled logical axioms. One reason for this, among others, is an intuition that the global schema should provide enough coverage in the conceptualization of the domain to cater for heterogeneity from the potential data sources. While this may not be a wrong intuition per se, this brings its own sets of problems.
  - To use an ontology requires adequate understanding on the semantics expressed in it. Larger and more complex ontologies thus are more difficult to understand. Moreover, large domain and foundational ontologies tend to contain a lot of abstract notions that ordinary data providers are not

familiar with. Learning those notions necessitates a steep learning curve, which can be too steep for those data providers. Also, those abstract notions may be too far from the real data and thus would never be populated with data.

  - Large and complex ontologies assert strong ontological commitments through complicated and tightly-coupled axiomatization, which may be too restrictive to be accepted by many parties potentially involved the data integration. Indeed, making such ontologies as a common modeling basis may not be a good idea if this basis turns out to be only acceptable by those parties that initiate the integration, but not by those that potentially join the integration much later.
  - Related to the above point, overly strong ontological commitments also means the ontology becomes too rigid. This impairs its extendibility as an effort to extend it would be difficult, costly, and worse, may break some of the existing functionality served by the system.

- In the cases where there is no domain or foundational ontology deemed suitable for the application, one has to construct a new one from scratch or from a modification of existing domain/foundational ontologies. As documented by Oberle et al. [102], in practice, this still a large and complicated manual effort, particularly if the aim is to obtain an ontology that captures a big domain.

Note that this observation ignores complexities in achieving interoperability on the syntactic level, such as data formats, or querying languages. Fortunately, some of these problems can be solved by employing Linked Data.

3.3 Linked Data Integration

The notion of Linked Data [13] refers to a set of four simple principles coined by Berners-Lee [9] for publishing and interlinking (structured) data on the Web. These principles are:

(1) "Use URIs as names of things."

(2) "Use HTTP URIs so that people can look up those names."

(3) "When someone looks up a URI, provide useful information, using the standards (RDF, SPARQL)."

(4) "Include links to other URIs, so that they can discover more things."

One appealing aspect of Linked Data for data providers is the fact that it uses the RDF data model, which is very simple and particularly suitable to represent heterogeneous and distributed data in a concise manner. RDF [39] is a graph-based data model whose basic unit is a *triple*. Each triple consists of three parts: a subject, a predicate, and an object. One form of triples is when all those three parts are URIs (Uniform Resource Identifiers) – there are other variants, of course, for example, a subject can be a blank node, while an object can be a blank node or a literal. Since URIs are also links that point to the location of the data and the URIs in a triple do not necessarily need to point to the same origin, it is very straightforward to state a piece of data than spans across different data sources. This simple model, together with the use of HTTP as a standardized data access protocol, simplifies data access: we can do away with a lot of complex APIs. One can easily navigate Linked Data similar to navigating web documents, i.e., in a "follow-your-nose" way. If there is an unrecognizable URI in the data currently at hand, assuming the Linked Data principles are followed, then one can simply look up the URI and dereference it to get further information. In addition, RDF is also accompanied with a standard query language, called SPARQL[117], that allows one to query across multiple RDF graphs

to find triples that satisfy certain patterns. In a way, it is similar to SQL for relational databases, though SPARQL is also equipped with capabilities to express federated queries spanning across different data sources.

The simplicity of the RDF model and the Linked Data principles helped spur the growth of structured data sharing on the web, leading to the so-called Web of Data. Schmachtenberg et al. [114] reported that there are currently more than 1000 linked datasets with 271% growth in the number of datasets within the three year period between 2011 and 2014. Moreover, there are standards and systems [3, 5, 40] that allow one to convert data, e.g., from relational databases, to RDF in a relatively straightforward way. Data publishing is also flexible as there is no requirement to subscribe to a fixed schema. Some even advocated to avoid schema altogether [98] and just work with instance data.

If we assume that all the data we are interested in can be made available as RDF, the syntactic interoperability for data integration is more or less no longer a big problem. In recent years, this idea has been gaining traction as evidenced by a number of implementations such as Hermes [121], FedX [115], LIDS [118], Information Workbench [57] and Optique [50]. Rather than a global schema, data integration in Linked Data from an architecture perspective would be more focused on federated query processing. Haase et al. [56] identified three possible architectural approaches for query federation. In the first approach, all data sources dump their data physically into a central repository to which queries are posed. In the second approach, data providers load their data to single repositories, one per provider. A federation layer sits on top of the single repositories of each data provider, and its task is to split an input query into sub-queries sent to the single repositories via a native API, and then combine the results. This federation is hidden from the user, i.e., the user only sees the federator as if it is a single repository. Finally, the third approach is similar to the second approach, but instead of native APIs, the single repositories provide

SPARQL endpoints, and this means there is no loading step in this approach.

The issue with semantics is in contrast rather more complicated. As illustrated by Hitzler et al. [64], when the Linked Data project [13] was conceived, Semantic Web research seemed too focused on studying ontologies for their own sake, neglecting the purpose of ontologies to improve semantic interoperability for data exchange. While there is some truth behind this countermovement, ignoring the values of shared ontologies would bring its own problems.

One of the main problems caused by too little schema information in Linked Data was the difficulty in understanding the content of a dataset and formulating a query against it [72]. In practice, if someone wants to query a dataset, (s)he needs to be familiar with the graph structure of the dataset to be able to formulate the correct query. Without any schema information, this can only be achieved by browsing the content of the dataset first. Thus, if the information needs call for querying over multiple datasets, which essentially amounts to a data integration, the situation would quickly become very troublesome. From a data integration perspective, it is thus apparent that availability of schema information is a key requirement that needs to be satisfied. This begs for bringing ontologies back into the picture, and recent work also indicated that traditional OBDI converged to this marriage between Linked Data technologies and ontologies [29, 90, 129].

4 Ontology Design Patterns for Data Integration

In this chapter, we will describe our effort addressing Hypothesis 1 from Section 1.2. Specifically, we shall describe the results we achieve so far in modeling ontology design patterns for the purpose of data integration, which constitute an important contribution of this doctoral research. We first begin with an overview of ontology design patterns. Afterwards, we describe the application context of this data integration, namely in ocean sciences, which drive the whole research work. We then describe our modeling approach and follow it with modeling details of the ODPs. We do not intend to put all of these modeling details here, and for more details, we refer the reader to Appendix A, which is based on a separate technical report Krisnadhi et al. [82]. Results in this chapter also appeared in Krisnadhi et al. [80, 83, 84].

4.1 Ontology Design Patterns

4.1.1 General Definition

Ontology Design Patterns (ODPs) were originally independently proposed by Gangemi [49] and Blomqvist et al. [15] as solutions to frequently occurring ontological modeling problems emerging in different domains. An ODP that models a conceptualization of some domain notion can act as a building block for more complex ontologies, for example, in the foundational DOLCE[1] and the Semantic Sensor

[1] http://www.loa.istc.cnr.it/old/DOLCE.html

Network (SSN) ontology [36].[2] Alternatively, an ODP can also capture best practices, such as naming conventions, documentation, vocabulary mappings, etc.

As part of the European-funded NeOn project,[3] Presutti et al. [110] developed different kinds of ontology design patterns for various purposes related to modeling ontologies. As noted in the project report just cited, the idea of ontology design pattern was inspired from software design pattern, and the term "design pattern" itself was in fact coined in the study of architecture to refer to "an archetypal solution to a design problem in a certain context". Presutti et al. provided the following definition of ontology design pattern, which is actually grounded on the DOLCE Ultra Light ontology [102].

Definition 4.1 (Ontology design pattern according to Presutti et al. [110])
An *ontology design pattern* is a modeling solution to solve a recurrent ontology design problem. It expresses a *design pattern schema*, which can only be satisfied by *design solutions*. Design solutions themselves provide the setting for *ontology elements* that play some *element roles* from the schema.

According to Presutti et al., a design pattern schema is a description that represents a conceptualization. It may contain some schematic structure, as well as roles, tasks, best practices, and parameters necessary to solve the design issue. Meanwhile, a design solution for a design pattern schema is a situation, i.e., essentially a factual implementation of the schema, which contains formal expressions involving ontology elements playing a role specified according to the schema. The above definition was grounded to the DOLCE Ultra Light ontology [102] where the notions of description, and situation are defined. Note that the above definition did not refer to a particular knowledge representation language. In the context of OWL, ontology elements would refer to classes, properties, etc., as defined in OWL semantics.

[2] http://www.w3.org/2005/Incubator/ssn/ssnx/ssn
[3] http://www.neon-project.org/

Hoekstra [67] noted that Definition 4.1 casts ODP as a reification of the relation between a design schema (i.e., conceptualization) and its solutions, and he argued that this makes the reason why a solution satisfies a schema rather unclear. Regarding this problem, Presutti et al. alluded to the idea of knowledge pattern [34], which Hoekstra then suggested to make it as an explicit part of ODP definition. Clark et al. defined a knowledge pattern to be a self-contained logical theory (the source) that can be used as a template to obtain another logical theory (the target).

Definition 4.2 (Clark et al. [34])
A *specification* is a tuple $\langle Sig, Axs \rangle$ where Sig is a signature consisting of non-logical symbols and logical operators, and Axs is a set of axioms over Sig. Given two specifications $\langle Sig_1, Axs_1 \rangle$ and $\langle Sig_2, Axs_2 \rangle$, a *signature morphism* is a consistent mapping M between Sig_1 and Sig_2. We say that M is a *specification morphism* between the given two specifications if for every axiom $a \in Axs_1$, $Axs_2 \models M(a)$, i.e., every axiom $a \in Axs_1$, after being translated according to M, follows from Axs_2.

Viewing an ODP as a knowledge pattern, Hoekstra suggested the following definition.

Definition 4.3 (Ontology design pattern according to Hoekstra [67])
An *ontology design pattern* (ODP) is an ontology $\mathcal{P}$ that is said to be *implemented as an ODP* in an ontology $\mathcal{O}$. Here, $\mathcal{P}$ is implemented as an ODP in $\mathcal{O}$ iff there exists a mapping M specified by a *signature morphism* between the signatures $\mathsf{Sig}(\mathcal{P})$ and $\mathsf{Sig}(\mathcal{O})$ such that for all axioms $\alpha \in \mathcal{P}$, it holds that $\mathcal{O} \models M(\alpha)$ where $M(\alpha)$ is the set of axioms produced by applying M to α and its signature.

The above definition intuitively captures why an ontology would be called a pattern. Hoekstra assumed that the mapping M above is total, although it may be neither injective nor surjective. This implies that an implementation of an ODP may

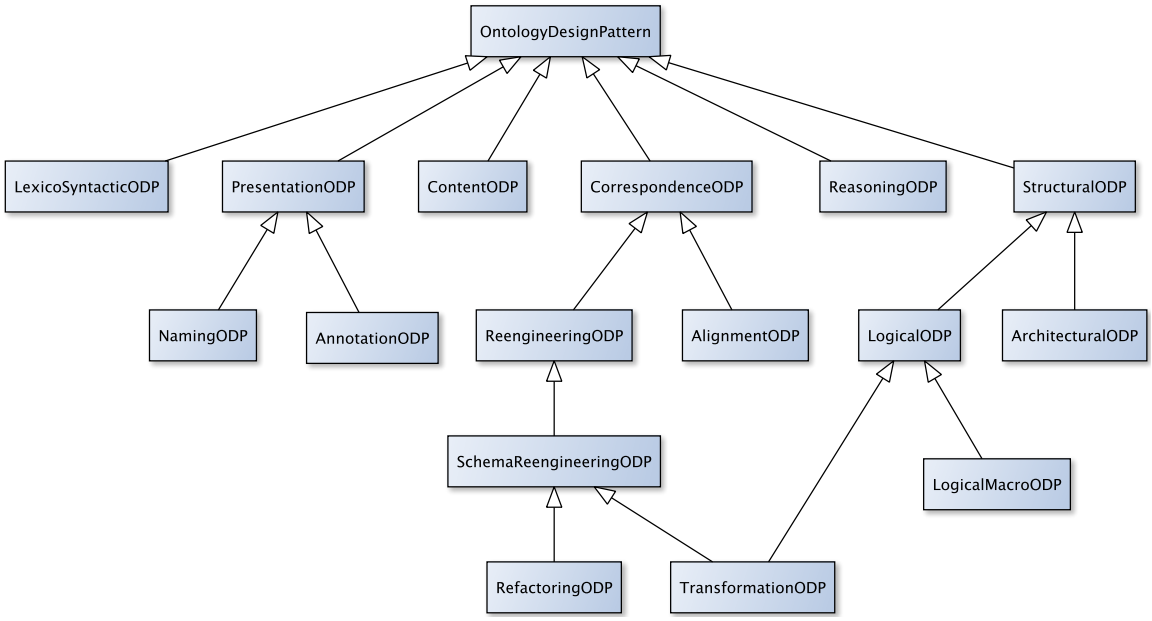


Figure 4.1: Different kinds of ontology design patterns

have more content than the ODP itself.

Presutti et al. [110] discussed different kinds of ODPs as depicted in Fig. 4.1. Lexico-syntactic ODPs refer to the linguistic schemas that what kind of ordering is valid for particular types of words in a natural language sentence. Presentation ODPs are concerned with good practices to achieve readability and usability of ontologies from users' point of view. Correspondence ODPs represent solutions regarding transforming conceptual models, possibly non-ontological ones, into a new ontology, as well as, representing different types of semantic relations between multiple ontologies. Reasoning ODPs, depending on the implementation of a given reasoning engine, describes types of reasoning that need to be done to obtain certain inferences.

Structural ODPs consist of Architectural ODPs and Logical ODPs. Architectural ODPs describe how an ontology should appear as a whole. For example, an Architectural ODP would describe how a taxonomy or a modular architecture look like. Logical ODPs, on the other hand, deal with problems of expressing certain logical constructs, which otherwise cannot be expressed by a given ontology language be-

cause it lacks a direct support for such constructs. For example, OWL semantics does not directly express n-ary relation, $n > 2$, in first-order logic. So, a Logical ODP can be specified to describe an ontology whose semantics approximate first-order n-ary relation.

4.1.2 Content Patterns

One type of ODPs we have not mentioned above is Content ODPs (CPs). CPs differ from other types of ODPs in that CPs are domain dependent. Ontology design problems that CPs are intended to solve concern the definition of vocabulary in the domain of interest. CPs are actually small ontologies, and they can be considered modules because one can create a larger, domain ontology through composition of CPs. Presutti et al. [110] defined the notion of CPs behind this intuition as stated in the following definitions.

Definition 4.4 (Content Pattern according to Presutti et al. [110])
A networked ontology is a member of a network of ontologies. A *network of ontologies* is a collection of ontologies related together via a variety of different relationships such as mapping, modularization, version, and dependency relationships. A *Content Ontology Design Pattern* (*CP*) is a distinguished networked ontology, residing in its own namespace, and it covers a specific set of *requirements*.

According to Presutti et al., a requirement is a *competency question*, which expresses a typical query that users may pose to the ontology. One straightforward way to express a requirement is by expressing it as a natural language question. On the other hand, since a CP is intended to model a particular notion in the domain of interest, it is also possible that a requirement is expressed as a description of characteristics that that notion *must* or *may* possess. Besides covering a specific set of requirements, Presutti et al. specified that

- CPs must be encoded in some formal representation language, e.g., OWL, so that they can be reused as ontology building blocks and some form of inference must be possible to made from the CPs.
- CPs must be small and autonomous.
- CPs must be cognitively relevant. That is, it should be easily recognized by domain experts as a representation of a core notion of the domain. Furthermore, it should be possible to provide a compact and intuitive visualization of the CPs.
- CPs should capture best practices of modeling in the domain of interest, which are obtained from the knowledge possessed by the domain experts.

An alternative definition of CPs were also offered by Hoekstra [67]. He formalized the remark by Presutti et al. [110] that not only CPs should be *invariant under signature morphism*, i.e., changes in the predicates in the signature do not change the pattern, but also preserve *downward taxonomic ordering*. That is, mapping of symbols (and axioms) in the pattern should subsume the symbols (and axioms) they map to. Using Definition 4.3, Hoekstra proposed the following definitions.

Definition 4.5 (Content Pattern according to Hoekstra [67])
The *import closure* of an ontology $\mathcal{O}$ is the smallest set $Imp(\mathcal{O})$ such that $\mathcal{O} \in Imp(\mathcal{O})$ and if $\mathcal{O}$ imports $\mathcal{O}'$, $Imp(\mathcal{O}') \subseteq Imp(\mathcal{O})$. The *axiom closure* of an ontology $\mathcal{O}$, denoted $\mathcal{O}_\cup$, is the smallest set containing all axioms from all ontologies in the import closure of $\mathcal{O}$. A *Content Ontology Design Pattern* (CP) is an ontology $\mathcal{P}$ that is said to be *implemented as a CP* in an ontology $\mathcal{O}$. Here, $\mathcal{P}$ is implemented as a CP in $\mathcal{O}$ iff $\mathcal{P}$ is implemented as an ODP in $\mathcal{O}$ and if $\mathcal{P} \subseteq \mathcal{O}_\cup$, then for every axiom $\alpha \in \mathcal{P}$, the signature morphism M also satisfies that $M(\alpha)$ is subsumed by α in $\mathcal{O}$.

4.2 Collection of Content Patterns for Global Schema

For data integration needs, content patterns[4] are particularly useful for providing a unified perspective over the data while still permitting a degree of semantic independence between the data repositories. Concretely, each content pattern focuses only on one generic notion, realized as a self-contained, highly modular ontology that contains some axiomatization (preferably using a standard like OWL) that defines the formal semantics and relationships between the vocabulary items used in it. It represents what constitutes the given notion and what important and widely reusable aspects about it the domain experts have agreed upon, and captures an appropriate graph structure for that notion. The axiomatization is carefully formulated such that no overly strong (i.e., application specific) ontological commitment[5] is made by the pattern. In comparison to a monolithic, upper ontology, a content pattern can thus be seen as a snippet that defines only one particular notion without excessive intricacies an upper ontology may entail. Relationships to other patterns that define different, but related, notions can still be provided, but not specified in detail. Such characteristics make content patterns more suitable for dealing with semantic heterogeneity when integrating knowledge than monolithic foundational ontologies.

The collection of patterns in principle then forms the global schema for data integration. As modeling is done one notion at a time with careful axiomatization so that axioms do not make too strong ontological commitments, the resulting schema is highly modular. This brings a number of benefits as follows.

- Extending the schema is easier. If some important notion is not yet modeled, we simply add another pattern without risking breaking down the axiomatization on the existing patterns as the axiomatization impose only minimal ontologi-

[4] Unless stated otherwise, we use the term "pattern" to refer to content patterns only.

[5] We do not offer a formal, mathematical definition for the idea of "ontological commitment" here – a formal definition was offered, e.g., in Oberle [101]. Intuitively, it corresponds to the intended meaning of the axioms and a stronger commitment means a narrower meaning.

cal commitments. Minimal ontological commitments also means that existing patterns are less likely to change.

- Data providers can choose and decide by themselves which parts of the global schema they want to actually populate with data, depending on the availability of the data, technical infrastructure constraints, local institutional policies, or established business process. This also means that data providers are not forced to subscribe to the *all* ontological commitments made by the whole global schema. Furthermore, each data provider (and future potential ones) can go through a gradual process to get up-and-running in the integration on their own pace independently.
- As a pattern constitutes a module, we have an obvious division of module, and this eases reuse of the patterns beyond the data integration project.

Beyond data integration, we argued that ODPs can also provide added values to Linked Data publishing and dataset reuse in general [112]. In fact, without us realizing it, some types of logical/structural patterns are used in abundance on the Web of Data For example, the idea of using *roles* in the sense of schema.org[6] can easily be understood as a type of design pattern to provide a uniform representation for a group of related relationships. More to the core of RDF and Linked Data, the common representation of n-ary predicates by way of several binary predicates is a typical example of a so-called *logical* (structural) ODP.[7] The RDF list construct could also easily be understood as a type of design pattern, even though it actually lacks a formal semantics according to the RDF specification. The benefits of reusing established logical (structural) patterns such as those just mentioned are rather obvious: They make it much easier for the experienced linked data user to recognize the intended

[6] https://schema.org/Role

[7] http://ontologydesignpatterns.org/wiki/Submissions:N-Ary_Relation_Pattern_%28OWL_2%29

meaning of a particular graph structure, and thus makes it easier for this user to query the dataset or to reuse it programmatically.

The aforementioned benefit for Linked Data can also be understood in the context of content patterns. We could even take the perspective that this is already done on the Web of Data. E.g., the abundant use of foaf:person or foaf:name could be understood as constituting a kind of pattern, and we assume that it is not really necessary to point out the advantages of this piece of vocabulary reuse, e.g., for the detection of person-related content in a formerly unknown linked dataset. The problem is, of course, foaf:person and foaf:name are rather minimalistic (if not to say, simplistic) types of pattern, which are as such not able to reflect finer-grained issues related to persons and names, such as the question about the proper way to sort, alphabetically by surname, a list of names including *Frank van Harmelen*: whether it's to be sorted with the *v*'s or the *H*'s is effectively dependent on the traditions of the country of origin (in this case, according to Dutch tradition, the appropriate grouping would be with the *H*'s). SKOS could also be understood as a type of simple pattern. Of course it lacks formal semantics, but like the RDF list construct (which also lacks formal semantics), the reuse of familiar structures is useful for understanding and querying datasets, and for reusing them.

More complex relationships, of course, require more complex patterns, e.g., when one creates a linked dataset about certain types of organizations and people affiliated with these organizations via different types of roles within the organizations. Five-star linked data can be created without giving the exact graph structure much thought. However in order to aid understandability and reuse of the datasets, it would be much more helpful to reuse, at least partially, some well-constructed graph structures. In fact, if the reused graph structure is carefully designed with avoidance of overly strong ontological commitments, then it will be widely reusable for different linked datasets. Furthermore, if the graph structure comes with an underlying logical

axiomatization which disambiguates meaning, then consistent reuse is made simpler. We shall continue this line of argument in Chapter 5. In the meantime, we focus on the set of patterns we have been developing for data integration in oceanography whose application context is described in Section 4.3.

4.3 Application Context: Oceanography Data Integration

The work is done in the context of the GeoLink project[8], and its predecessor, the OceanLink project, two building blocks of the EarthCube program sponsored by the US National Science Foundation. EarthCube[9] is an initiative that brings together the US geoscience research community through a number of funded building blocks, research coordination networks, and special interest groups to establish a knowledge infrastructure crucial for enabling cross-discipline scientific endeavors. Obviously, cross-repository data integration is an important piece of this initiative. The GeoLink project itself aims to leverage advances in semantic technologies for developing a data integration and discovery framework involving seven major data repositories, mainly in the area of ocean science. Those repositories are BCO-DMO[10], DataONE[11], IEDA[12], IODP[13], LTER[14], MBLWHOI Library[15], and R2R[16]. The data integration problem faced by this project is both technically and socially challenging, not just because of the lack of alignment between data from different repositories, but also due to fundamental differences in the way data and knowledge are modeled. GeoLink tackles this problem by the use of Linked Data[13] and Ontology Design Patterns [49].

[8] http://www.geolink.org
[9] http://www.earthcube.org
[10] Biological and Chemical Oceanography Data Management Office - http://www.bco-dmo.org
[11] Data Observation Network for Earth - https://www.dataone.org/
[12] Interdisciplinary Earth Data Alliance - http://www.iedadata.org/
[13] International Ocean Discovery Program - http://www.iodp.org/
[14] Long Term Ecological Research Network - http://www.lternet.edu/
[15] Marine Biological Laboratory/Woods Hole Oceanographic Institution Library - http://www.mblwhoilibrary.org/
[16] Rolling Deck to Repository - http://www.rvdata.us/

Linked Data enables repositories to describe and publish their data using standard syntax featuring links to other data, possibly in different repositories. Meanwhile, ODPs allows a horizontal integration featuring semantic alignment between repositories with possibly independent semantic models. The approach taken by this project is hoped, not only to be applicable for the participating repositories, but is also extendible beyond this project to include the broader geoscience community.

4.4 Collaborative Modeling Approach

4.4.1 The Modeling Workflow

There is currently only one known systematic method of designing an ontology design pattern, namely the eXtreme design [108]. Though this method embodies good principles in designing ontology patterns including user involvement, collaborative modeling with iterative refinement, expressing user requirements through competency questions and contextual statements, and task-oriented design focusing on the user requirements, there are aspects from this method that we had to adjust to fit the constraints of our project.

First, we shorten the duration spent on listing all competency questions, and rather mixing it with directly listing on what information should a pattern provide and what constraints should it absolutely satisfy. This is due to the time constraints that it was not easy to gather the whole ontology development team for face-to-face meeting and any meeting time is precious. The eXtreme design method suggested the use of supporting tools such as wiki, but this was also not readily available.

Second, testing of the created patterns whether user requirements are served through SPARQL queries was mostly done through manual and on-paper checking because data providers did not have a readily accessible Linked Data infrastructure to support the testing. Third, we do not do pair design because of the lack of manpower

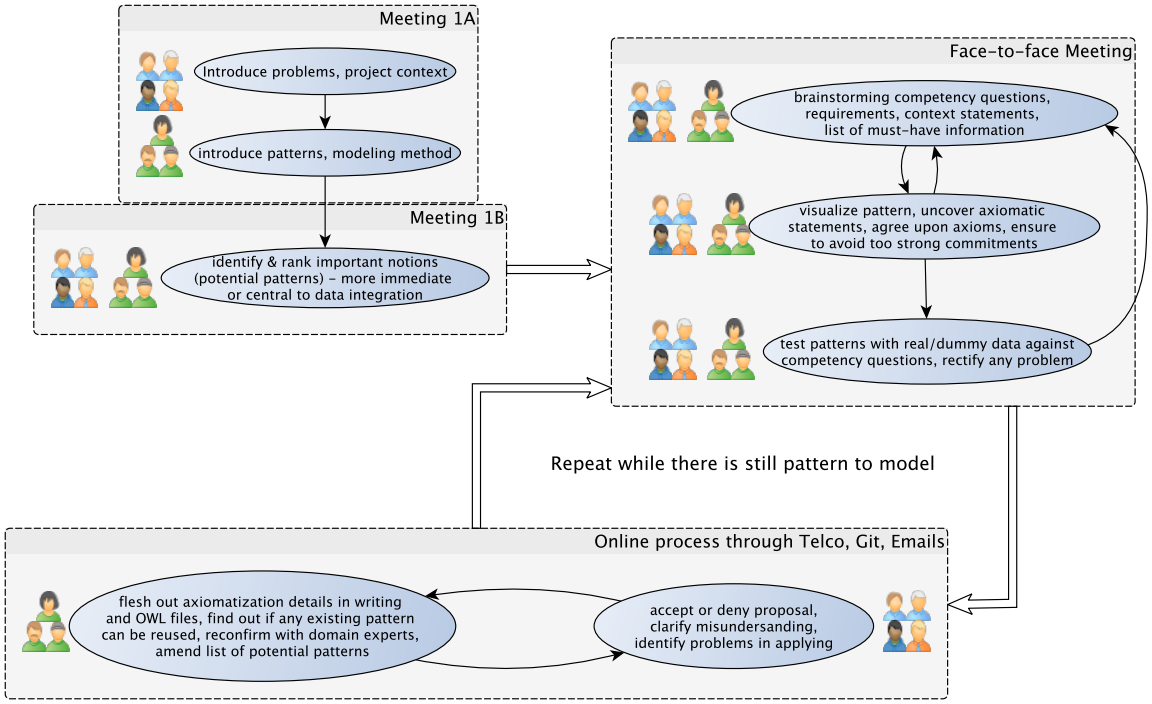


Figure 4.2: Modeling Approach

in this project. Instead, iterative refinement is intensified through frequent teleconferences, which became quite useful since we make sure the core and most important commitments of the patterns nailed down during the face-to-face meeting.

So, generally, we adopted the following steps in our modeling approach as depicted in Fig. 4.2.

1. The team conducts an initial face-to-face meeting to get into the project context. In this meeting, data providers share some high-level description on motivations for establishing integration, as well as how an integrated system should serve their needs. In the same meeting, knowledge engineers present a general overview of the pattern development process and their benefits for the integration.

2. Both data providers and knowledge engineers immediately try to identify a list of important notions (i.e., the list of potential patterns) of the domain occurring in the integration scenarios. This list needs not be exhaustive since this can

be amended later, but more importantly, the team identifies which notions are more immediate, important, or central to the integration, hence should be modeled first before the rest.

3. In the next several face-to-face meetings, the team goes through the notions one by one and starts modeling each of them. For each of the notion, the modeling technically involves:

    - brainstorming the competency questions, context statements, and the list of information that a pattern modeling the given notion should provide;
    - visualizing and documenting all important piece of information necessary for modeling through a combination of whiteboard and the use of diagram/graph editing and collaborative note taking tools; the visual notation is provided later below;
    - knowledge engineers guide the team to create an explicit list of ontological commitments that the participant would want to make (e.g., what a pattern must have, provide, etc.) based on the information the team currently obtain by brainstorming; each ontological commitment is expressed in natural language sentences preferably in a form that is easier to translate into formal logical axioms, e.g., "a person must have a name, but may have zero or more aliases", or "a cruise must have exactly one chief scientist who may be affiliated with a certain university at the time of the cruise"; here, the knowledge engineers provide a warning if necessary regarding the consequence of making a particular ontological commitment to prevent making it too strong;
    - the team may be aware of some existing ontology or pattern that model the notion currently being considered, and a quick decision is made on whether the team should do a "vanilla" reuse or use it as the inspiration;

- if the considered notion is deemed straightforward, then the team can decide to immediately move on to modeling other notions; also, the team may identify some notions that turn out to be rather important to model, and thus amend the list of potential patterns;
- the team together manually test the resulting patterns against the user stories, competency questions, and context statements, by providing an example on how the pattern would be populated; here, the team should identify potential problematic situations and rectify the pattern accordingly.

4. In between the face-to-face meetings, knowledge engineers focus on fleshing out the axiomatization details, and write the resulting patterns down in a project document as well as in OWL documents. Problems encountered in axiomatization are documented and communicated to the whole team in the next meeting (face-to-face or online).

With the steps above, we were able to model more than a dozen patterns as documented in Appendix A and in Krisnadhi et al. [82]. Selected samples are described in the next section.

4.4.2 Graphical Notation

We now describe the graphical notation that we use to describe the patterns. Note that the graphical notation described here is *informal* in nature, and its sole purpose is for communicating the intention of the pattern to human readers. Nevertheless, the presence of a visual aid like this is a key component in the modeling process. Formal representation of the pattern is given through logical axioms using notation described in Section 2.3.2 and 2.3.3.

Each pattern is visualized as a graph structure. The nodes in such a graph represent classes, datatypes or data range expressions, and individuals. Some classes in a

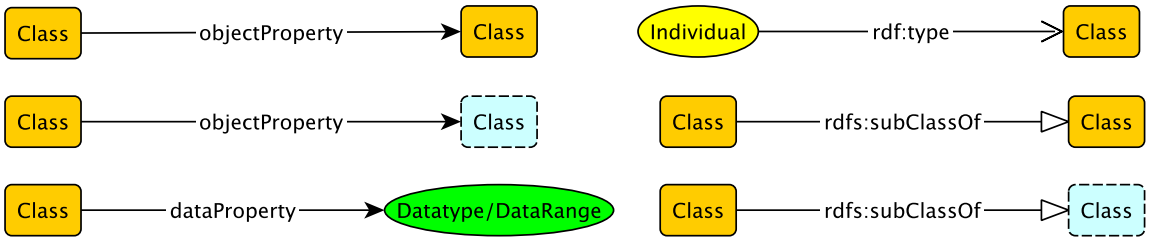


Figure 4.3: Graphical Notation for a pattern

pattern are also intended as a "hook" to another class in a different pattern, separate from this pattern. These classes are depicted as blue nodes with dotted lines and should be read as "this is a notion that is relevant to the currently modeled notion, but more details of this notion are defined in another pattern". These classes *are* defined in the current pattern, i.e., become part of its signature, though use of but additional details are modeled separately. The types of node-edge-node connection in such a graph are depicted in Fig. 4.3. An edge is either an object property, a data property, a typing relation, and a subclassing relation. The first two are in the sense of OWL. For object and data properties, the direction of the edge is from the class that is a domain of the property toward a class or datatype/data range that is a range of this property.

4.5 Selected Modeling Details

We present a number of patterns we were able to obtain in the modeling effort for the OceanLink and GeoLink projects. As described in the previous section, prior to modeling, the team identified a number of notions that are present in data sources and can be considered a facet in data discovery. The initial list contained Person, Cruise, Organization, Vessel, Funding Award, Program (in the sense of NSF programs), Dataset, Publication/Document, Physical Sample, and Instrument. From this list, the team quickly realized that we need a pattern to model roles of a person

in a particular situation. We slightly generalized this into Agent Role, which we include in the list. Also, when modeling Cruise, the team realized the need to separate objects from their information objects on which one can attach labeling information, information about web page, etc. The last two amendments were also essentially good practices suggested by Blomqvist et al. [14] from their experience in pattern development. The list later thus evolved to also include Personal Information Item, Information Object, Place, Measurement, etc. In this section, we do not intend to present all details of the patterns, but rather select a few representative examples to illustrate the modeling approach.

4.5.1 Person

When modeling Person, a rather obvious solution is to simply reuse the FOAF model. The team, however, decided that the FOAF model to be inappropriate to be used as a whole. So, we simply took an inspiration from it, namely by considering Person to be a kind of Agent. Initially, we have the following list of modeling requirements and context statements:

(i) a person may have a name;

(ii) a person may have an organization as his/her affiliation;

(iii) a person's name may appear in different formats, e.g., first name followed by last name, separate fields for first name and last name, the use of prefixes and suffixes, etc.;

(iv) other unknown types of personal information may be included by other data providers, e.g., nationality.

Here, competency questions were not specified explicitly since the above requirements can be obtained quite quickly without them.

The graph visualizing the Person pattern is given Fig. 4.4. Here, a modeling decision was made to represent all kinds of personal information as a separate notion,

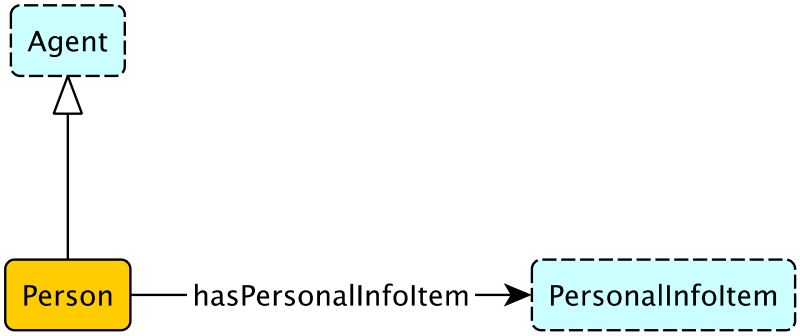


Figure 4.4: Person pattern

which necessitates a separate pattern modeled by the project later. We also decided that an Agent pattern is needed as a person can have a role in many contexts, and sometimes these roles are also performed by some organization. Hence, inspired by FOAF, we model a person as a kind of agent.

Our consideration above led to the following axiomatization. First, every person is an agent.

$$\mathsf{Person} \sqsubseteq \mathsf{Agent} \tag{4.1}$$

Here, Agent is understood to be modeled in a separate pattern. Affiliation relationship is a special kind of role that a person may hold in an organization. By elaborating this further, we quickly understood that such a role can appear in many contexts and situations, and may also be held by an organization. So, we decided to create a separate abstraction called Agent Role, and model it as its own pattern. Regarding person's name and other types of personal information, again we see that these notions may be not as simple, so we abstract them away as a personal info item, leading to a new requirement:

(v) a person may have personal information item, e.g., names, nationalities, etc.

PersonalInfoItem pattern is then modeled separately, and its specialization includes PersonName, which we omit from this section (see Appendix A.5 and A.6, or Krisnadhi et al. [82] for further details).

The rest of the pattern consists only of asserting domain and range restrictions as well as class disjointness. As explained in Section 2.3.2, we use guarded version of these restrictions.

$$\exists \mathsf{hasPersonalInfoItem}.\mathsf{PersonalInfoItem} \sqsubseteq \mathsf{Person} \tag{4.2}$$

$$\mathsf{Person} \sqsubseteq \forall \mathsf{hasPersonalInfoItem}.\mathsf{PersonalInfoItem} \tag{4.3}$$

Finally, class disjointness is asserted.

$$\mathsf{alldisjoint}(\mathsf{Person}, \mathsf{PersonalInfoItem}) \tag{4.4}$$

4.5.2 Agent Role

The next example is Agent Role pattern. This pattern initially arose from a modeling decision made when we considered the Person pattern in the context of modeling different ways a person can be associated with an organization. It became apparent later on, however, that this is also a very useful pattern. As pointed out by Blomqvist et al. [14], an example of good modeling practice is the separation between persons and their roles. For example, we should not say that a student is a person, but rather, being a student is a role performed by some person. Additional research on ontologydesignpatterns.org repository as well as schema.org vocabulary also indicate that notions very similar to this exist. So, we took and combined inspiration from those examples and modeled the Agent Role pattern according to the following requirements:

- an agent role embodies a particular instance of role that an agent performs (here, agent can be a person, an organization, etc.), thus we agreed that an agent role is performed by exactly one agent;
- furthermore, the instance of role that an agent perform above is only meaningful in a particular context, thus we assert that every agent role is a role in exactly one context; this is a generic context, i.e., can be anything;

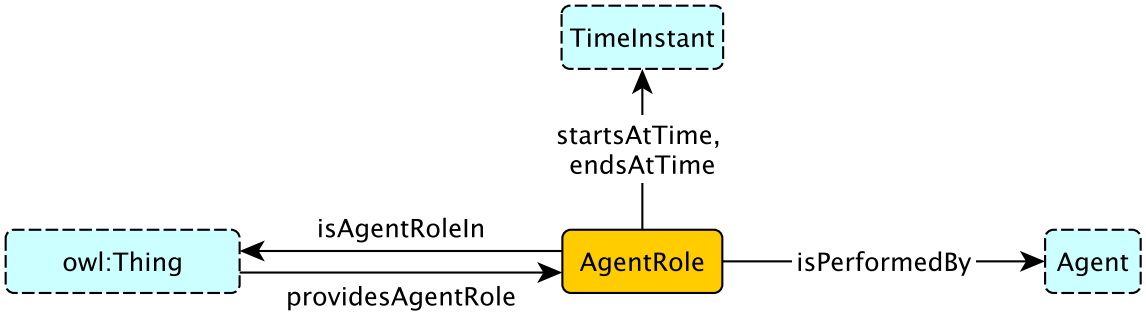


Figure 4.5: The Agent Role pattern

- from the perspective of the aforementioned context, we want to say that such a context may provide an agent role; note, however, that more than one instance of agent roles may be relevant here;
- a role typically has a time limit, i.e., an agent may not perform a particular role forever, hence, we say that an agent role starts at some time point and ends at some time point.

Fig. 4.5, describes the Agent Role pattern that captures the above requirements. The axiomatization is as follows: An AgentRole is performed by exactly one Agent, has exactly one starting time and one ending time, and is an agent role in exactly one thing. Note that the Agent Role pattern has a hook with Agent pattern and OWL Time ontology [65], the latter through the class TimeInstant, which corresponds to the class Instant in OWL Time ontology.

$$\begin{aligned} \mathsf{AgentRole} \sqsubseteq (=1\ \mathsf{isPerformedBy.Agent}) \sqcap (=1\ \mathsf{isAgentRoleIn}.\top) \\ \sqcap (=1\ \mathsf{startsAtTime.TimeInstant}) \\ \sqcap (=1\ \mathsf{endsAtTime.TimeInstant}) \end{aligned} \tag{4.5}$$

$$\mathsf{providesAgentRole} \equiv \mathsf{isAgentRoleIn}^{-} \tag{4.6}$$

We next assert the domain and range restrictions of the properties in this pattern. Specifically for the isAgentRoleIn property, since it ranges over all individuals,

range restriction is not needed and its domain restriction is unguarded. For the providesAgentRole property, it is the other way around: domain restriction is not needed while its range restriction is unguarded. For the other object properties, domain and range restrictions are guarded.

$$\exists \mathsf{isPerformedBy}.\mathsf{Agent} \sqsubseteq \mathsf{AgentRole} \tag{4.7}$$

$$\mathsf{AgentRole} \sqsubseteq \forall \mathsf{isPerformedBy}.\mathsf{Agent} \tag{4.8}$$

$$\exists \mathsf{startsAtTime}.\mathsf{TimeInstant} \sqsubseteq \mathsf{AgentRole} \tag{4.9}$$

$$\mathsf{AgentRole} \sqsubseteq \forall \mathsf{startsAtTime}.\mathsf{TimeInstant} \tag{4.10}$$

$$\exists \mathsf{endsAtTime}.\mathsf{TimeInstant} \sqsubseteq \mathsf{AgentRole} \tag{4.11}$$

$$\mathsf{AgentRole} \sqsubseteq \forall \mathsf{endsAtTime}.\mathsf{TimeInstant} \tag{4.12}$$

$$\exists \mathsf{isAgentRoleIn}.\top \sqsubseteq \mathsf{AgentRole} \tag{4.13}$$

$$\top \sqsubseteq \forall \mathsf{providesAgentRole}.\mathsf{AgentRole} \tag{4.14}$$

Finally, we assert the following class disjointness axioms.

$$\mathsf{alldisjoint}(\mathsf{AgentRole}, \mathsf{Agent}, \mathsf{TimeInstant}) \tag{4.15}$$

4.5.3 Cruise

Cruise Overview

The Cruise pattern is an example of more complex patterns in Krisnadhi et al. [82]. The cruise notion is a very important in oceanography and a lot of things, whose data may be important, have some connection to the cruise notion. Intuitively, the notion of oceanographic cruise is rather too specific since one can obviously also think of sight-seeing cruises, pleasure cruises, or even science cruises which are not used for ocean science purposes. In this context, to develop a pattern that is highly reusable, the generic notion of cruise would be a better candidate than ocean science cruise.

However, for the purpose of the project, that would be a too abstract generalization that may become too complicated. So, instead, we really focus on the notion of oceanographic cruise and do not venture beyond that. This pattern has also appeared in Krisnadhi et al. [83].

For ocean science data repositories, a cruise can be seen as an abstract record that can act as a glue between otherwise separate pieces of information that ocean science data repositories may store. Those pieces of information are derived from generic use cases, which we can describe through a number of competency questions that represent queries to the pattern.

One kind of competency question concerns the spatiotemporal information contained within the cruise route or trajectory. For example,

(1) "Find all cruises passing through Gulf of Maine in August 2013."

(2) "Show the trajectories of cruises in operation in September 2013."

Another kind of competency question involves querying the vessel on which a cruise is operated.

(3) "List all cruise vessels that departed from Woods Hole in 2012."

Also relevant to a cruise are competency questions for finding the people who serve in some capacity during the cruise's operation. For example,

(4) "Find the chief scientists of any cruise that collected samples of carbon-isotope data in Lake Superior."

Activities on a cruise may result in datasets or other digital objects stored in repositories, about which some users may issue questions such as:

(5) "What datasets were produced by the cruise AE0901?"

Finally, some party may also be interested in some administrative information about a cruise, exemplified by the following competency questions:

(6) "Which cruises are funded by the NSF award DBI-0424599?"

(7) "List all cruises under the Ocean Flux Program."

(8) "What is the address of the webpage for the cruise AE0901?"

The above questions illustrate different pieces of information that are related to the notion of Cruise. From Question 1, 2, and 3, we know that trajectory and vessel are two important components of a cruise. A closer observation would lead us to an understanding that the trajectory and vessel of a cruise are indispensable: there is no cruise without a vessel and a trajectory. From Question 4, we understand that a cruise involves people who hold particular roles in its operation. To answer Question 5, information about an ocean science cruise clearly has to be related to the data and documents the cruise generated during its operation. Furthermore, due to Question 6, 7, and 8, it also needs to be related to the information about the funding award and program which support the activities embodied by the cruise, as well as other pieces of information such as the webpage of a cruise. In principle, all of these pieces of information are described by their own separate patterns which may possess more detailed information that need not be formulated explicitly in the cruise pattern.

The use cases above gave us an insight that the notion of Cruise can essentially be viewed from three different angles: (1) as the route or trajectory a vessel traverses, hence providing the *spatiotemporal boundary* of a cruise; (2) as the collection of activities performed by *actors*, which can be humans or other kinds of agents; and (3) as a placeholder for various pieces of explanatory information that fit neither the trajectory nor the constituting activities, e.g., funding award, cruise type, etc. Points (1) and (2) motivate us to understand a cruise as a type of *event* since events are things that happen at some place and time whereby actors participate by performing some activities or roles. Moreover, by point (3), a cruise is not just a simple event; it is an event adorned with other explanatory information. Specifically, we conceptualize a cruise as *an adorned event undertaken by a vessel traversing through a particular trajectory*. This motivates a design choice where we formalize the Cruise

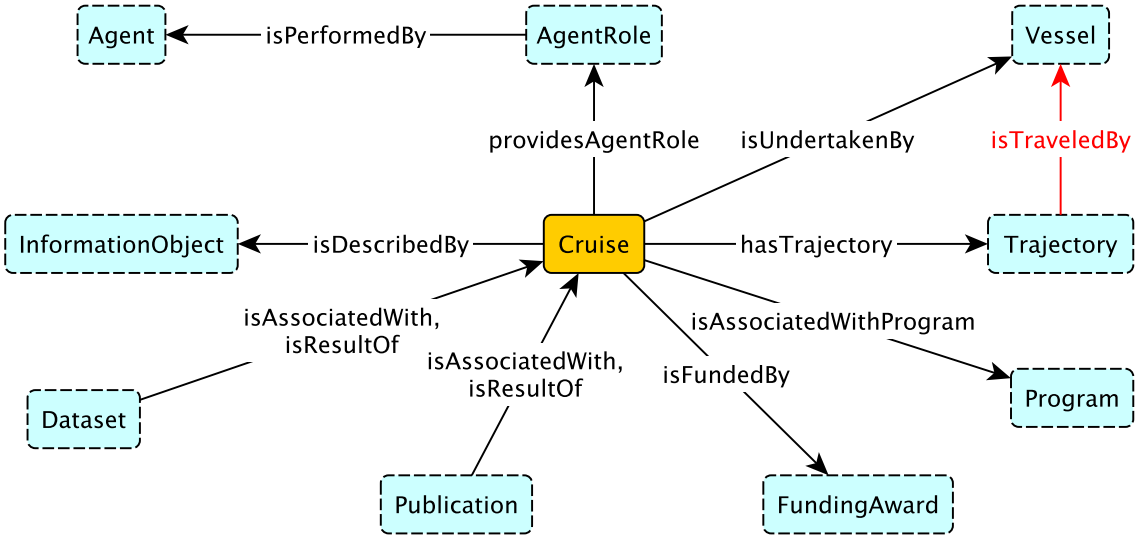


Figure 4.6: Overview of the Cruise pattern; the red arrow indicates a property implied by a property chain.

pattern through reusing, adjusting, combining, and extending several already-existing patterns, including the Semantic Trajectory [68], Simple Event Model [58], and the Information Object pattern derived from DOLCE [102].

Core of Cruise Pattern

Fig. 4.6 depicts a high level overview of the Cruise pattern, which omits some details explained and visualized in the remainder of this section. Notice that the relationship between the classes Cruise, Trajectory, and Vessel involves an internal class of the Trajectory subpattern. Since a cruise is a kind of event, we specify that Cruise is a subclass of the more generic class Event. Adornments of a cruise are either those that concern the cruise itself, such as funding awards, datasets, programs, and publications, or those that concern the abstraction of a cruise as an entity in some information system, e.g., webpages, textual descriptions, etc. We model the latter through the (re-)use of the Information Object pattern described in Appendix A.8, which is derived from DOLCE's Information Object pattern [102]. Meanwhile, the former is modeled through direct relationships with the concerned entities, modeled in separate

patterns. During the preparation of this thesis, we have modeled the Funding Award and Program patterns, described in Appendix A.10 and A.11, while the Dataset and Publication patterns are still not yet finalized. The direct relationships to the Program, Funding Awarad, Dataset, and Publication patterns are defined through the use of isAssociatedWith, isResultOf, isFundedBy, and isAssociatedWithProgram properties according to Fig. 4.6. The direction of the relationships refers to how the actual data would actually be represented by most data providers. More precisely, the direction of isAssociatedWith goes from Dataset to Cruise, because data providers in the GeoLink project would prefer their actual data regarding this relationship would be published as triples of the form: `<dataset> <isAssociatedWith> <cruise>`. The same consideration is also behind the opposite direction of isAssociatedWithProgram property from Cruise to Program. The naming of isAssociatedWithProgram is dictated by the Program pattern in Appendix A.11, while for Dataset and Publication, we still use the generic naming of isAssociatedWith because the corresponding patterns are not finalized.

We assert that a cruise is an event. We then assert that a cruise has exactly one trajectory, is undertaken by exactly one vessel, and is described by exactly one InformationObject. We also assert that if a cruise is undertaken by a vessel, then the trajectory of the cruise has to be traveled by the vessel.

$$\mathsf{Cruise} \sqsubseteq \mathsf{Event} \tag{4.16}$$

$$\mathsf{Cruise} \sqsubseteq (=1\ \mathsf{hasTrajectory.Trajectory}) \sqcap (=1\ \mathsf{isUndertakenBy.Vessel}) \tag{4.17}$$

$$\mathsf{Cruise} \sqsubseteq (=1\ \mathsf{isDescribedBy.InformationObject}) \tag{4.18}$$

$$\mathsf{hasTrajectory}^{-} \circ \mathsf{isUndertakenBy} \sqsubseteq \mathsf{isTraveledBy} \tag{4.19}$$

Note that since isTraveledBy is implied by a property chain, OWL 2 specification forbids us to express a cardinality restriction using this property. Consequently, we cannot axiomatize that the trajectory of a cruise can only be traveled by one vessel.

Next, we state the guarded domain and range restrictions for the aforementioned properties.

$$\exists \mathsf{hasTrajectory}.\mathsf{Trajectory} \sqsubseteq \mathsf{Cruise} \tag{4.20}$$

$$\mathsf{Cruise} \sqsubseteq \forall \mathsf{hasTrajectory}.\mathsf{Trajectory} \tag{4.21}$$

$$\exists \mathsf{isUndertakenBy}.\mathsf{Vessel} \sqsubseteq \mathsf{Cruise} \tag{4.22}$$

$$\mathsf{Cruise} \sqsubseteq \forall \mathsf{isUndertakenBy}.\mathsf{Vessel} \tag{4.23}$$

$$\exists \mathsf{isDescribedBy}.\mathsf{InformationObject} \sqsubseteq \mathsf{Cruise} \tag{4.24}$$

$$\mathsf{Cruise} \sqsubseteq \forall \mathsf{isDescribedBy}.\mathsf{InformationObject} \tag{4.25}$$

$$\exists \mathsf{isTraveledBy}.\mathsf{Vessel} \sqsubseteq \mathsf{Trajectory} \tag{4.26}$$

$$\mathsf{Trajectory} \sqsubseteq \forall \mathsf{isTraveledBy}.\mathsf{Vessel} \tag{4.27}$$

$$\exists \mathsf{isAssociatedWith}.\mathsf{Cruise} \sqsubseteq \mathsf{Publication} \sqcup \mathsf{Dataset} \tag{4.28}$$

$$\mathsf{Publication} \sqcup \mathsf{Dataset} \sqsubseteq \forall \mathsf{isAssociatedWith}.\mathsf{Cruise} \tag{4.29}$$

$$\exists \mathsf{isResultOf}.\mathsf{Cruise} \sqsubseteq \mathsf{Publication} \sqcup \mathsf{Dataset} \tag{4.30}$$

$$\mathsf{Publication} \sqcup \mathsf{Dataset} \sqsubseteq \forall \mathsf{isResultOf}.\mathsf{Cruise} \tag{4.31}$$

$$\exists \mathsf{isFundedBy}.\mathsf{FundingAward} \sqsubseteq \mathsf{Cruise} \tag{4.32}$$

$$\mathsf{Cruise} \sqsubseteq \forall \mathsf{isFundedBy}.\mathsf{FundingAward} \tag{4.33}$$

$$\exists \mathsf{isAssociatedWithProgram}.\mathsf{Program} \sqsubseteq \mathsf{Cruise} \tag{4.34}$$

$$\mathsf{Cruise} \sqsubseteq \forall \mathsf{isAssociatedWithProgram}.\mathsf{Program} \tag{4.35}$$

Cruise as Event and Cruise Trajectory

Fig. 4.7 depicts cruise as events, details of cruise trajectory, and the fact that cruise may provide agent roles. A cruise trajectory represents a route that the cruise takes in the duration of its activities. We reuse the multi-granular Semantic Trajectory pattern, which already provides basic vocabulary and OWL axiomatization [68] to model cruise trajectory. The multi-granularity of that pattern accommodates the

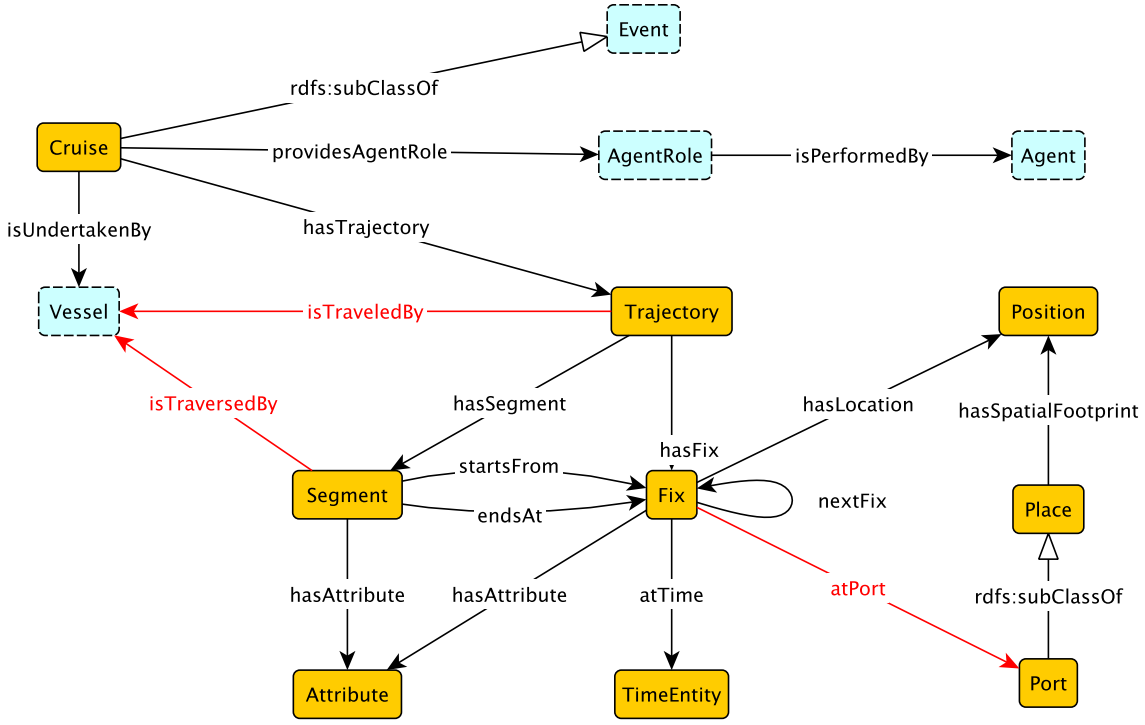


Figure 4.7: The Cruise as Events: Trajectory and Agent Roles

modeling of the (geospatial) path traveled by the vessel of the cruise and with some extension to the vocabulary, it can also be used to model certain meaningful notions, such as the port sequence, port stops, arrival and departure times, etc.

According to the Semantic Trajectory pattern, a trajectory is given by a collection of *fixes*, representing time-stamped locations. Non-spatiotemporal information specific to a fix can be included by assigning it some attributes, for example, to indicate that the fix is the arrival to some port stop. The ordering of the fixes is based on their temporal information. Between two consecutive fixes, we can define a *segment*, which is traversed by some moving object. There is no requirement forcing that all segments in the trajectory can only be traversed by the same moving object. Furthermore, the Semantic Trajectory pattern can also include information about the source of the spatiotemporal information of a fix, such as GPS sensors.

The Semantic Trajectory pattern is a very generic pattern. To fit our needs, we

make a number of adjustments.

(i) We introduce the Port class as subclass of Place, which can then be used to annotate those special fixes.

(ii) Since a cruise is undertaken by no more than one vessel, we remodel the isTraversedBy property to be entailed by a property chain. This is to ensure that if a trajectory is traveled by a vessel, all of the trajectory's segments are also traversed by the same vessel.

(iii) We omit the part of Semantic Trajectory pattern that allows us to include information about the source, e.g., GPS sensors, that establish fixes because they appear to be irrelevant our current project, at least at the current stage. It is of course possible that this can be included in future version of this pattern, and in fact, it is straightforward to do so.

(iv) The Semantic Trajectory pattern models the ordering of fixes by assuming that two fixes and a segment are predefined, and then entailing the actual ordering as nextFix relation from them. In our case, the data typically already contains ordering of fixes, i.e., the nextFix relation is explicit in the data, and thus, segments are auto-instantiated from it. This is motivated by real scenarios whereby indeed only properties of fixes will often be known, in particular their locations, and temporal extension, whether they are at ports, whether they are arrival or departure fixes from ports, and in which sequence the fixes occurred. The traversing vessel will usually also be known. However, trajectory segments to which the trajectory pattern attaches information about the vessel are usually not explicitly represented in the data.

All the above consideration are depicted in Fig. 4.7 and imply that we cannot reuse the whole OWL axiomatization from Hu et al. [68] for our needs here. We thus present in this section our version of the OWL axiomatization for modeling trajectory

which, for the sake of completeness, will also repeat the necessary parts of the OWL axiomatization from that paper.

Note from Fig. 4.7 that the trajectory of a cruise clearly contains spatiotemporal information that is relevant for understanding a cruise as an event. Intuitively, a cruise as an event occurs at places given by the whole route it takes according to its trajectory. In particular, the ports where a cruise stops is a place at which a cruise as event occurs. Similar consideration can also be made for the temporal information. These spatiotemporal information are, however, buried deep within the trajectory.

We now explain Fig. 4.7 in the following. Here, a cruise has exactly one trajectory and is undertaken by exactly one vessel. The trajectory of each cruise has at least two distinct fixes, one represents the starting fix, and the other represents the ending fix. Each of those fixes that is not the ending fix connects to exactly one other fix via the nextFix property, while the ending fix itself connects to no other fix in the trajectory. This provides us with the ordering of fixes. Each fix has some location and time information and may have some attributes. Particular attributes include port_stop_arrival and port_stop_departure. The former indicates that the corresponding fix represents the arrival to a port stop in the trajectory, while the latter represents the departure from a port stop. The location of a fix is a spatial footprint of some place of interest. In particular, if the location of a fix corresponds to some port, we also directly connect the fix with the port through the atPort property. The trajectory also has at least one segment. Each of those segments is auto-generated from two consecutive fixes and is traversed by the vessel by which the cruise is undertaken. A segment may also have some attributes, if necessary.

We begin the axiomatization for cruise trajectory by defining its basic components: fixes and segments. A fix has a location and a time stamp, and always belongs to one particular trajectory. Also, a fix cannot be followed by more than one other fix,

and cannot follow itself. This gives a linear structure in the ordering of the fixes.

$$\mathsf{Fix} \sqsubseteq \exists \mathsf{hasLocation.Position} \sqcap \exists \mathsf{atTime.TimeEntity} \sqcap (=1\ \mathsf{hasFix}^{-}.\mathsf{Trajectory}) \tag{4.36}$$

$$\mathsf{Fix} \sqsubseteq (\leqslant 1\ \mathsf{nextFix.Fix}) \sqcap \neg \exists \mathsf{nextFix.Self} \tag{4.37}$$

We next define starting and ending fixes as special kinds of fixes.

$$\mathsf{StartingFix} \equiv \mathsf{Fix} \sqcap \neg \exists \mathsf{nextFix}^{-}.\top \tag{4.38}$$

$$\mathsf{EndingFix} \equiv \mathsf{Fix} \sqcap \neg \exists \mathsf{nextFix}.\top \tag{4.39}$$

$$\mathsf{StartingFix} \sqcap \mathsf{EndingFix} \sqsubseteq \bot \tag{4.40}$$

A trajectory is linked to at least two consecutive fixes where the first fix is the starting fix. Also, if a fix belongs to a trajectory, then its successor fix also belongs to the same trajectory.

$$\mathsf{Trajectory} \sqsubseteq \exists \mathsf{hasFix.}(\mathsf{StartingFix} \sqcap \exists \mathsf{nextFix.Fix}) \tag{4.41}$$

$$\mathsf{hasFix} \circ \mathsf{nextFix} \sqsubseteq \mathsf{hasFix} \tag{4.42}$$

A segment starts from exactly one fix, and for every fix with a successor fix, there is a segment that starts from it. If a fix belongs to a trajectory and there is a segment that starts from this fix, then the segment belongs to the trajectory. Furthermore, if a segment starts from a fix, then it ends at the successor of the fix.

$$\mathsf{Segment} \sqsubseteq (=1\ \mathsf{startsFrom.Fix}) \tag{4.43}$$

$$\exists \mathsf{nextFix.Fix} \sqsubseteq (=1\ \mathsf{startsFrom}^{-}.\mathsf{Segment}) \tag{4.44}$$

$$\mathsf{hasFix} \circ \mathsf{startsFrom}^{-} \sqsubseteq \mathsf{hasSegment} \tag{4.45}$$

$$\mathsf{startsFrom} \circ \mathsf{nextFix} \sqsubseteq \mathsf{endsAt} \tag{4.46}$$

The above axiomatization ensures that a trajectory is linked to all of its fixes and segments. Note that the above axioms do not model a trajectory to have a finite sequence

of fixes of unknown length, which cannot actually be modeled in OWL 2. In our case, however, data providers will only provide cruise trajectory as a finite collection of fixes with a known ordering, which can be written as a set of ABox axioms of the form $\mathsf{Fix}(f_1), \ldots, \mathsf{Fix}(f_n), \mathsf{nextFix}(f_1, f_2), \ldots, \mathsf{nextFix}(f_{n-1}, f_n), \mathsf{StartingFix}(f_1), \mathsf{EndingFix}(f_n)$. Since a fix cannot have more than one successor fix, we implicitly obtain a finite, linear ordering given by the transitive closure of nextFix.

We next define atPort as a shortcut via property chain involving hasLocation and hasSpatialFootprint, which can be written in Datalog as: $\mathsf{hasLocation}(x, y), \mathsf{hasSpatialFootprint}(z, y), \mathsf{Port}(z) \rightarrow \mathsf{atPort}(x, z)$. The following two axioms express the rule where rollifiedPort is a fresh property name defined solely for the class Port, which is defined as a subclass of Place.

$$\mathsf{hasLocation} \circ \mathsf{hasSpatialFootprint}^{-} \circ \mathsf{rollifiedPort} \sqsubseteq \mathsf{atPort} \tag{4.47}$$

$$\exists \mathsf{rollifiedPort.Self} \equiv \mathsf{Port} \tag{4.48}$$

$$\mathsf{Port} \sqsubseteq \mathsf{Place} \tag{4.49}$$

If a trajectory is traveled by a vessel, then every segment is traversed by that vessel.

$$\mathsf{hasSegment}^{-} \circ \mathsf{isTraveledBy} \sqsubseteq \mathsf{isTraversedBy} \tag{4.50}$$

We assert the following guarded domain and range restrictions.

$$\exists \mathsf{hasFix.Fix} \sqsubseteq \mathsf{Trajectory} \tag{4.51}$$

$$\mathsf{Trajectory} \sqsubseteq \forall \mathsf{hasFix.Fix} \tag{4.52}$$

$$\exists \mathsf{nextFix.Fix} \sqsubseteq \mathsf{Fix} \tag{4.53}$$

$$\mathsf{Fix} \sqsubseteq \forall \mathsf{nextFix.Fix} \tag{4.54}$$

$$\exists \mathsf{hasLocation.Position} \sqsubseteq \mathsf{Fix} \tag{4.55}$$

$$\mathsf{Fix} \sqsubseteq \forall \mathsf{hasLocation.Position} \tag{4.56}$$

$$\exists \mathsf{atPort.Port} \sqsubseteq \mathsf{Fix} \tag{4.57}$$

$$\mathsf{Fix} \sqsubseteq \forall \mathsf{atPort.Port} \tag{4.58}$$

$$\exists \mathsf{atTime.TimeEntity} \sqsubseteq \mathsf{Fix} \tag{4.59}$$

$$\mathsf{Fix} \sqsubseteq \forall \mathsf{atTime.TimeEntity} \tag{4.60}$$

$$\exists \mathsf{hasSpatialFootprint.Position} \sqsubseteq \mathsf{Place} \tag{4.61}$$

$$\mathsf{Place} \sqsubseteq \forall \mathsf{hasSpatialFootprint.Position} \tag{4.62}$$

$$\exists \mathsf{hasSegment.Segment} \sqsubseteq \mathsf{Trajectory} \tag{4.63}$$

$$\mathsf{Trajectory} \sqsubseteq \forall \mathsf{hasSegment.Segment} \tag{4.64}$$

$$\exists \mathsf{startsFrom.Fix} \sqsubseteq \mathsf{Segment} \tag{4.65}$$

$$\mathsf{Segment} \sqsubseteq \forall \mathsf{startsFrom.Fix} \tag{4.66}$$

$$\exists \mathsf{endsAt.Fix} \sqsubseteq \mathsf{Segment} \tag{4.67}$$

$$\mathsf{Segment} \sqsubseteq \forall \mathsf{endsAt.Fix} \tag{4.68}$$

$$\exists \mathsf{isTraversedBy.Vessel} \sqsubseteq \mathsf{Segment} \tag{4.69}$$

$$\mathsf{Segment} \sqsubseteq \forall \mathsf{isTraversedBy.Vessel} \tag{4.70}$$

$$\exists \mathsf{hasAttribute.Attribute} \sqsubseteq \mathsf{Segment} \sqcup \mathsf{Fix} \tag{4.71}$$

$$\mathsf{Fix} \sqsubseteq \forall \mathsf{hasAttribute.Attribute} \tag{4.72}$$

$$\mathsf{Segment} \sqsubseteq \forall \mathsf{hasAttribute.Attribute} \tag{4.73}$$

We next model the actors of a cruise, which is achieved by aligning with Agent Role pattern. In this context, a cruise may provide a number of special agent-roles performed by some agent. For now, we simply state the domain and range restrictions, as well as assert that every agent-role has to be performed by exactly one agent.

$$\exists \mathsf{providesAgentRole.AgentRole} \sqsubseteq \mathsf{Cruise} \tag{4.74}$$

$$\mathsf{Cruise} \sqsubseteq \forall \mathsf{providesAgentRole.AgentRole} \tag{4.75}$$

$$\exists \mathsf{isPerformedBy.Agent} \sqsubseteq \mathsf{AgentRole} \tag{4.76}$$

$$\mathsf{AgentRole} \sqsubseteq \forall \mathsf{isPerformedBy.Agent} \tag{4.77}$$

$$\mathsf{AgentRole} \sqsubseteq (=1\ \mathsf{isPerformedBy.Agent}) \tag{4.78}$$

Various types of agent-roles a cruise may provide are included in a class hierarchy (not visualized) rooted at the AgentRole class. The hierarchy is axiomatized below.

$$\mathsf{CaptainRole} \sqcup \mathsf{OperatorRole} \sqcup \mathsf{SchedulerRole} \sqsubseteq \mathsf{AgentRole}$$

$$\mathsf{ObserverRole} \sqcup \mathsf{InspectorRole} \sqsubseteq \mathsf{AgentRole}$$

$$\mathsf{ForeignObserverRole} \sqcup \mathsf{OtherObserverRole} \sqsubseteq \mathsf{ObserverRole}$$

$$\mathsf{EngineerRole} \sqcup \mathsf{ScientistRole} \sqcup \mathsf{TechnicianRole} \sqsubseteq \mathsf{AgentRole}$$

$$\mathsf{ChiefEngineerRole} \sqsubseteq \mathsf{EngineerRole}$$

$$\mathsf{ChiefScientistRole} \sqcup \mathsf{CoChiefScientistRole} \sqcup \mathsf{PostdocScientistRole} \sqsubseteq \mathsf{ScientistRole}$$

$$\mathsf{LeadTechnicianRole} \sqcup \mathsf{MarineTechnicianRole} \sqsubseteq \mathsf{TechnicianRole}$$

$$\mathsf{StudentRole} \sqcup \mathsf{EducatorRole} \sqsubseteq \mathsf{AgentRole}$$

$$\mathsf{GraduateStudentRole} \sqcup \mathsf{UndergraduateStudentRole} \sqcup \mathsf{K12StudentRole} \sqsubseteq \mathsf{StudentRole}$$

$$\mathsf{HigherEdEducatorRole} \sqcup \mathsf{K12EducatorRole} \sqsubseteq \mathsf{EducatorRole}$$

Finally, we assert the following class disjointness axioms:

$$\begin{aligned}\mathsf{alldisjoint}(&\mathsf{Cruise}, \mathsf{InformationObject}, \mathsf{AgentRole}, \mathsf{Agent}, \mathsf{FundingAward}, \mathsf{Program},\\ &\mathsf{Trajectory}, \mathsf{Vessel}, \mathsf{Fix}, \mathsf{Segment}, \mathsf{Attribute}, \mathsf{TimeEntity}, \mathsf{Place}, \mathsf{Position},\\ &\mathsf{Dataset}, \mathsf{Publication})\end{aligned} \tag{4.79}$$

4.6 Discussion

At this stage, by following similar steps as described in previous sections, we were able to obtain 17 patterns for the GeoLink project. They are Agent, Agent Role, Event, Information Object, Identifier, Person, Personal Info Item, Person Name, Organization, Funding Award, Program, Place, Cruise, Platform, Vessel, Physical

Sample, and Property Value. These are results of collaborative modeling, though I acted as the lead modeler for all but the last two patterns. So except for the last two, the remaining fifteen are described in Appendix A. Some of these patterns are less developed than the others, though this does not turn out to be a big problem since data integration is done gradually and these less developed patterns are not of high priority for integration. OWL implementations of these patterns are available online.[17]

We can now revisit Hypothesis 1 from Section 1.2. Since the patterns are made available in a format according to Semantic Web standards, the next task for data providers is to populate the pattern with data, that is, to publish linked datasets that employ vocabulary defined in the patterns. Once these linked datasets are published, one can either use a data warehousing approach or a mediator approach.

For a data warehousing approach, all those linked datasets are collected as one big RDF dataset at one location, accessible via some standard access method such as a SPARQL endpoint. Users can then query the data using vocabulary defined in the patterns, i.e., the patterns act as the global schema. Since all dumped linked datasets are annotated using the same set of vocabulary terms, querying over the aforementioned SPARQL endpoint essentially realizes a data integration scenario.

For a mediator approach, somebody needs to implement a mediator component that accepts a query expressed using vocabulary defined by the patterns. The mediator can actually just forward the query to all participating repositories without going through any mapping since we assume that linked datasets from those repositories are annotated using the vocabulary defined by the patterns. The mediator then collect the answers and return them to the user. Again, here the patterns act as the global schema.

[17] http://schema.geolink.org/

Also, because we assume all data repositories expose their data as Linked Data, most of heterogeneity issues involving data format, syntax, access method, etc., are less of a concern. Meanwhile, the semantic heterogeneity issue is solved by modeling the patterns that provide semantic interoperablity. Furthermore, patterns contain strong, machine readable semantics. So, we essentially have all the component we need to satisfy Hypothesis 1. Unfortunately, we can only say that Hypothesis 1 is partially verified because there are still some problems if data providers are really supposed to use patterns. This is illustrated in the following discussion.

First, the data providers indicated that the pattern collection covers a sufficiently wide range of notions that they deem important or necessary to publish their data. On the flip side, there are still several key notions that we are not yet able to develop, e.g., Publication, Dataset, and Measurement, and this would be one of the tasks of the project in the coming year, given that the GeoLink project is still ongoing [120]. Experience during the modeling convinced all parties that extending the pattern collection to cover additional notions would not be problematic. Modeling mistakes such as having no distinction between persons and their roles are avoided.

However, the data providers also indicated from their feedback [120] that there are problems in the usability of the patterns, particularly for publishing data. The root of the usability problems come from the fact that some of the patterns contain a rather complicated structure, mostly due to reification, e.g., the use of Agent Role, or Information Object. Though in modeling sessions we have made it clear why such a reification is important: for flexibility of future expansion of the integration framework, or to accommodate aspects such as more spatio-temporal information or provenance information, some data providers find it hard to *populate* the pattern. There are a number of possible reasons for this.

1. Data providers sometimes do not begin populating the patterns right away due to technical or other reasons. Thus, by the time, they start populating the

patterns, they already forget the explanation that knowledge engineers provide during the meeting. Online meetings and written notes do not help much since data providers involved in the project know almost next to nothing on ontology modeling.

2. Some data providers struggle in populating the patterns because they see a number of classes or patterns, which do not have an obvious counterpart in their data source or local schema. For instance, the hasChiefScientist relationship between a Cruise and a Person was reified through the use of Agent Role pattern. This is intended to accommodate cases of n-ary relationship, e.g., when a person is a chief scientist on a cruise, while he was still afiliated to one institution, though currently that person is affiliated to a different institution. If a data source models this relationship only using a very simple binary relation, then the data source's owner would be confused as no notion of Agent Role in his data source that aligns immediately with Agent Role pattern.

3. The reification above led to the question of *URI proliferation*. The data provider in the above example usually unwilling to generate and maintain URIs just for the sake of populating Agent Role pattern. Also, (s)he does not want to generate blank nodes for such a complicated structure on his/her side. Thus, it is essentially up to the federator to maintain such a query. However, realizing this is not obvious technically if we do not want to be forced to use a centralized storage for the RDF graph representing data for the whole integration.

All of the above problems can be understood as the presence of gap between the conceptualization by the patterns and the local conceptualization at each data provider's side. Obviously, different data providers would have different complexity in their local conceptualization. Our idea is to bridge the gap using an intermediate schema that is "simpler" than the patterns and "closer" to the data, but still relatively easy to align to the patterns. Although similar problems have been widely studied

in relational databases and well-known as the notion of views, this is not the case for Linked Data context. So, in Chapter 5, we shall introduce the notion of *pattern views*, i.e., views for patterns, and we argue that this could be useful not only to solve the aforementioned problems, but more generally, also for data integration in Linked Data context.

5 Pattern Views

5.1 Introduction

As discussed in the previous chapter, a well-designed CP does not necessarily imply acceptance from data providers. One contentious point causing this is the perception that abstraction that we introduce in a CP, well-intentioned may it be, can still be viewed as a step too far for their concern. For example, Linked Data publishers often do not like reification, though this modeling choice is often needed to represent a n-ary (with $n > 2$) relationship in RDF. In particular, this can happen if a data provider takes a position that reification is *not* needed for *his* data, although there are generic use cases that necessitate the reification. One reason causing this stance from such a data provider is that such generic use cases are *not* applicable to *his* data. Even worse, it is possible that this is a make-or-break situation for the data provider. At the outset, one can say that this stance is egoistical and the data provider should have relented for the sake of the rest of the stakeholders. On the other hand, the point of having CPs in the first place is to be flexible on this situation so that both modeling in CP and specific modeling choices of a data provider can coexist.

Our proposed solution is to use another layer of modeling consisting of minimalistic schema that is simpler than the pattern and can be populated with the data more easily by the data providers. This schema is called a *pattern view*. The idea is inspired from the notion of views well-known in database. A view in relational database is a virtual (i.e., not physical) relation that is defined by a query on one or more physically stored relations. Views can be queried like ordinary, physical tables, but they do not

```
:ex1 a chess:ChessGame ;
     chess:hasWhitePlayerName "Bobby Fischer" .
```

Figure 5.1: Dataset A: two simple RDF triples

```
:ex1 a chess:ChessGame ;
  chess:hasWhitePlayer [
     a chess:Agent ;
     chess:hasName "Bobby Fischer" ;
     chess:hasELORating "2780" ;
    skos:closeMatch <http://dbpedia.org/resource/Bobby_Fischer> .
     ] .
```

Figure 5.2: Dataset B: data conforming to the pattern in Fig. 5.3

store the data physically. A pattern view in the meantime is a Linked Data schema like patterns. We can populate it just like we populate patterns. In a data integration context, however, we treat pattern view as schema that is only visible to a particular data provider. To distinguish it from the data provider's local schema, the pattern view should intuitively have at most the same terminological coverage as the pattern.

As a minimalistic example – taken from Rodríguez-Doncel et al. [112] – one might think of representing a chess game as an instance of a certain `chess:ChessGame` class, which is attributed the literal "Bobby Fischer" as the name of the white player, resulting in Dataset A given in Fig. 5.1. In many cases, linked data publishers are satisfied with these triples and seek no further complication. However, one might think of a case where details on the person of Bobby Fischer are needed, e.g., as provided by Dataset B in Fig. 5.2.

We can see that Dataset B corresponds to a slightly more complex graph structure than Dataset A, possessing an additional resource node that Dataset A lacks. Going one step further, Dataset B may be actually be published according to a CP $\mathcal{O}$, given in Fig. 5.3, where it is stated that chess games are played by exactly one agent as white player.

The benefit of the more complex structure for Dataset B is the richer information

chess:Agent rdf:type owl:Class.
chess:ChessGame rdf:type owl:Class.
chess:hasWhitePlayer rdf:type owl:ObjectProperty.

∃chess:hasWhitePlayer.chess:Agent ⊑ chess:Game
chess:Game ⊑ ∀chess:hasWhitePlayer.chess:Agent
chess:Game ⊑ (=1 chess:hasWhitePlayer.chess:Agent)
∃chess:hasName.xsd:string ⊑ chess:Agent
chess:Agent ⊑ ∀chess:hasName.xsd:string

Figure 5.3: Pattern $\mathcal{O}$ used by Dataset B – mixed turtle and DL syntax are used.

chess:ChessGame rdf:type owl:Class.
chess:hasWhitePlayerName rdf:type owl:DataProperty.
chess:hasWhitePlayerName rdfs:domain chess:ChessGame.
chess:hasWhitePlayerName rdfs:range xsd:string.

Figure 5.4: Possible schema $\mathcal{V}$ for Dataset A, that is also a view for O in Fig. 5.3

content and the flexibility for even more information about the agent who was the white player of the chess game, hence increasing the potential for reuse. On the other hand, publishing the dataset under the form of Dataset A may satisfy certain audience expecting simplicity. Despite this, we can still see that both datasets still contain roughly the same information regarding the name of the white player in the chess game. In this context, in addition to saying that the Dataset B is published according to the *ontology design pattern* $\mathcal{O}$, we may also say that the Dataset A is published according to a *view* for $\mathcal{O}$, for example, as specified in Fig. 5.4.

More precisely, both CPs and views for CPs are ontologies that can act as schemas over data – henceforth, we use the term ontology and (linked data) schema interchangeably. An ontology is seen as a CP by virtue of qualitative characteristics, such as being well-engineered, concise, able to cater multiple perspectives and granularity,

highly reusable, modular, and highly focused on modeling only a single key notion in a domain. If a CP contains abstractions that are typically designed to cater to multiple perspectives, a view of the CP should be understood as a simplified form of the CP for a *particular* perspective from some data provider or consumer with the assumption that such a view is so self-explanatory that populating it requires a much less effort by the corresponding data provider than populating the patterns directly. Consequently, a CP can have multiple views, and some bridges are needed between CPs and its views to allow them to work together.

5.2 Consumer View

The motivation in the previous section alluded to benefits of view from the perspective of data providers or *producers*. To define the notion of view formally in this section, however, we shall look at this from the perspective of data consumers. The intuition is that a view contains relationships that can be seen as shortcuts of a number of relationships in the pattern. For example, as depicted in Fig. 5.5, the property `chess:hasWhitePlayerName` defined in Fig. 5.4 could be seen as a shortcut in the pattern defined in Figure 5.3. Thus, consumers can see (part of) the data given by Dataset B (Fig. 5.2) as Dataset A (Fig. 5.1). Such a shortcut in principle can be expressed using some mapping rule, which allows one to obtain a simple graph structure.

From the perspective of data producers, the idea is similar, but the direction of the mapping is the opposite. This is more complicated since we need to generate nodes in the graph of the data. For our formal definition, we start with *consumer view*, which corresponds to the perspective of data consumer above. The idea is that a consumer view corresponds to a flat structure that can be inferred from a pattern.

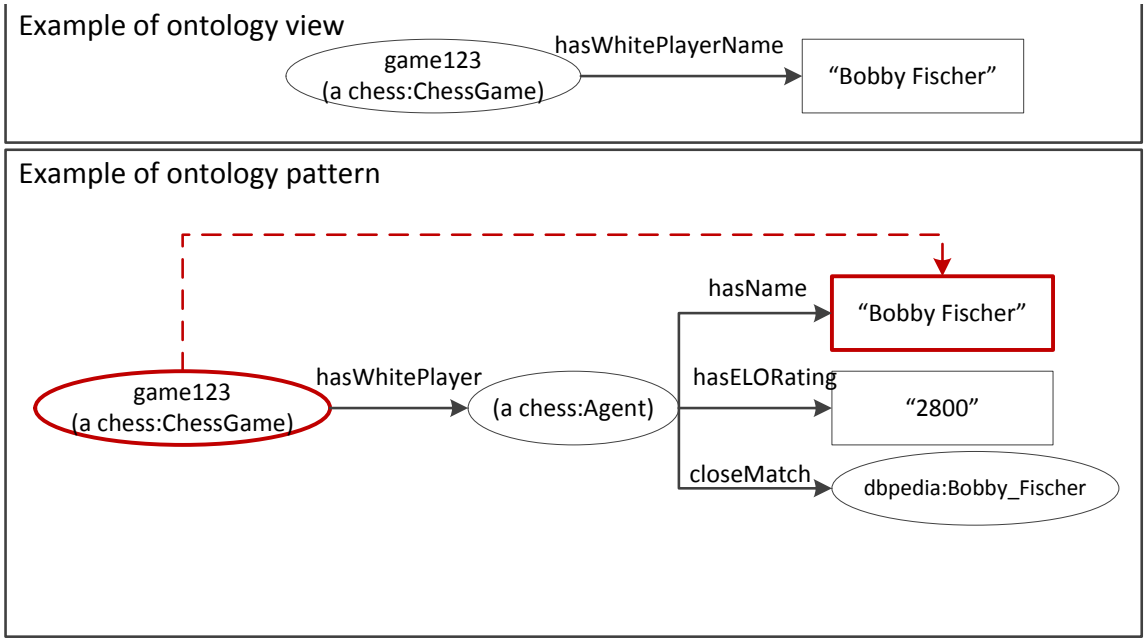


Figure 5.5: The red-colored, dotted line is a shortcut in the pattern, and correspond to the property `hasWhitePlayerName`.

Definition 5.1

An OWL ontology $\mathcal{O}$ is *flat* if both of the following holds:

- it *only* contains axioms of the following forms:
  - (i) $\mathsf{dom}(R) \sqsubseteq A_1 \sqcup \cdots \sqcup A_n$, $n \geq 1$ where R is either an object or data property, and $A_1, \ldots, A_n$ are class names;
  - (ii) $\mathsf{range}(R) \sqsubseteq C_1 \sqcup \cdots \sqcup C_n$, $n \geq 1$ where R is an object property iff $C_1, \ldots, C_n$ are class names, and R is a data property iff $C_1, \ldots, C_n$ are datatypes of arity 1;
  - (iii) $A \sqsubseteq B$ where A and B are class names;
  - (iv) $R \sqsubseteq S$ where R and S are property expressions;
- for each property R in $\mathcal{O}$, there is only at most one axiom of type (i) and at most one axiom of type (ii).

The semantics of (i) is equivalent to the axiom $\exists R.\top \sqsubseteq A_1 \sqcup \cdots \sqcup A_n$, while (ii) is equivalent to $\top \sqsubseteq \forall R.(C_1 \sqcup \cdots \sqcup C_n)$.

Definition 5.2

A *schema graph* G of a flat ontology $\mathcal{O}$ is a set of triples $\langle C, R, D \rangle$ where C is $\top$ or a class name in $\mathcal{O}$, R is either an object property or a data property in $\mathcal{O}$, and D is $\top$, a class name, or a unary datatype in $\mathcal{O}$ such that $\mathcal{D}$ is a class name or $\top$ if R is an object property, and $\mathcal{D}$ is a unary datatype, otherwise. We say that G is consistent with $\mathcal{O}$ if for every triple $\langle C, R, D \rangle$ in G, it holds that:

- there is an axiom $\mathsf{dom}(R) \sqsubseteq A_1 \sqcup \cdots \sqcup A_n$ in $\mathcal{O}$ such that $C = A_i$ for some $1 \leq i \leq n$;
- there is an axiom $\mathsf{range}(R) \sqsubseteq A_1 \sqcup \cdots \sqcup A_n$ in $\mathcal{O}$ such that $D = A_i$ for some $1 \leq i \leq n$.

Thus, a consistent schema graph visualizes a flat ontology schematically, i.e., it describes how the classes are related to each other through the properties. For example, for the flat ontology given in Fig. 5.4, a consistent schema graph would contain only one triple $\langle$`chess:ChessGame`, `chess:hasWhitePlayerName`, `xsd:string`$\rangle$. Obviously, it is possible that a flat ontology has more than one consistent schema graph since the domain and range restrictions in it may be expressed over union of class names. Our intuition is that a schema graph captures the intended conceptualization of the view, to distinguish it from other possible conceptualizations that are consistent with the view. Our definition later then pairs a flat ontology with a consistent schema graph to constitute a view. We define the syntax and semantics of the mapping rules for consumer view in the folowing.

Definition 5.3

Let $\mathcal{P}$ and $\mathcal{V}$ be two ontologies. A *class mapping rule* from $\mathcal{P}$ to $\mathcal{V}$ is a Datalog rule of the form $D(x) \rightarrow C(x)$ where D is a class name in $\mathcal{P}$ and C is a class name in $\mathcal{V}$.

A *property shortcut rule* from $\mathcal{P}$ to $\mathcal{V}$ is a Datalog rule of the form:

$$D_1(x_1) \wedge R_1(x_1, x_2) \wedge D_2(x_2) \wedge R_2(x_2, x_3) \wedge \cdots \wedge R_n(x_n, x_{n+1}) \wedge D_{n+1}(x_{n+1}) \rightarrow C_1(x_1) \wedge R(x_1, x_{n+1}) \wedge C_2(x_{n+1})$$

where $n \geq 1$, R is either an object property expression or a data property in $\mathcal{V}$, C_1 is a class name in $\mathcal{P}$, C_2 is a class name if R is an object property expression, and is a datatype if R is a data property, $R_1, \ldots, R_{n-1}$ are object property expressions in $\mathcal{P}$, R_n is either object property expression or a data property in $\mathcal{V}$, $D_1, \ldots, D_n$ are either $\top$ or class names in $\mathcal{P}$, and it holds that:

- if R is an object property expression, then R_n is an object property expression and D_{n+1} is a class name or $\top$ in $\mathcal{P}$; and we call the rule an *object property shortcut rule*
- if R is a data property, then R_n is a data property and D_{n+1} is a datatype in $\mathcal{P}$; and we call the rule a *data property shortcut rule.*

In every property shortcut rule in the aforementioned form, we call the atomic class D_1 the *shortcut source* of the rule and D_{n+1} the *shortcut target* of the rule. We can omit any of $D_i(x_i)$, $i = 1, \ldots, n$, when writing a property shortcut rule if that D_i is the top concept $\top$. We can also omit $D_{n+1}(x_{n+1})$ if D_{n+1} is $\top$ or a standard unary datatype. Similarly, we can omit $C_1(x_1)$ if it is $\top$ and $C_2(x_{n+1})$ if it is $\top$ or standard unary datatype. If D_{n+1} is actually standard datatype and $C_2(x_{n+1})$ is omitted, we assume that $D_{n+1} = C_2$.

Definition 5.4

Let $\mathcal{P}$ and $\mathcal{V}$ be two ontologies. Also, let $\mathcal{I}$ be an interpretation of $\mathcal{P}$ where $\Delta_o^{\mathcal{I}}$, $\Delta_d^{\mathcal{I}}$, $\cdot^{\mathcal{I}_o}$, and $\cdot^{\mathcal{I}_d}$ are respectively the object domain, the data domain, the object interpretation function, and the data interpretation function of $\mathcal{I}$. Note that the object domain and data domain are disjoint.

The *first-order reading* of an object data interpretation function is obtained by viewing class expressions as unary relations and object property expressions as bi-

nary relations. The first order reading of a data interpretation function is defined analogously: datatypes as unary relations and data property as binary relations.

The semantics of an object property shortcut rule (resp. a class mapping rule) from $\mathcal{P}$ to $\mathcal{V}$ is a standard first-order semantics over $\Delta_o^{\mathcal{I}}$ where the first-order interpretation function is obtained by taking the first order reading of $\cdot^{\mathcal{I}_o}$ and then extending it by assigning a binary relation (resp. unary relation) to the object property expression (resp. class name) occurring in the head of the rule.

The semantics of a data property shortcut rule from $\mathcal{P}$ to $\mathcal{V}$ is a standard first order semantics over the union $\Delta_o^{\mathcal{I}} \cup \Delta_d^{\mathcal{I}}$ where the first-order interpretation function is obtained by taking the first-order reading of the $\cdot^{\mathcal{I}_d}$ and then extending it by assigning a binary relation of type $\Delta_o^{\mathcal{I}} \times \Delta^{\mathcal{I}_d}$ to the data property occurring in the head of the rule.

The notion of model and entailment are then defined as usual.

Intuitively, a consumer view over a pattern should be such that any linked dataset annotated using vocabulary in the pattern (without introducing inconsistency) yields a linked dataset that is consistent with the view and the intended schema graph when the mapping rules are applied.

Definition 5.5

An RDF graph G is *simple* if all of its triples are either of the form $\langle s, \texttt{rdf:type}, C\rangle$ or of the form $\langle s, p, o\rangle$ where none of s, p, o and C are predefined vocabulary terms from RDF, RDFS, and OWL. Let $\mathcal{O}$ be an ontology or content pattern. A simple RDF graph G is $\mathcal{O}$-*annotated* if

- for every triple $\langle s, \texttt{rdf:type}, C\rangle \in G$, C is a class in $\mathcal{O}$; and
- for every triple $\langle s, P, o\rangle$ with $P \neq \texttt{rdf:type}$, P is a data property in $\mathcal{O}$ whenever o is literal and P is an object property in $\mathcal{O}$ otherwise.

In this case, we also say that $\mathcal{O}$ is *populated* by G. Furthermore, the ABox induced by the $\mathcal{O}$-annotated G is the set $\mathcal{A}(G)$ obtained by:

- translating every triple $\langle s, \texttt{rdf:type}, C\rangle$ into a concept assertion $C(s)$; and
- translating every triple $\langle s, P, o\rangle$ with $P \neq \texttt{rdf:type}$ into a property assertion $P(s, o)$.

We say that the $\mathcal{O}$-annotated G is consistent with $\mathcal{O}$ iff $\mathcal{A}(G) \cup \mathcal{O}$ is consistent.

Definition 5.6

Let $\mathcal{V}$ and $\mathcal{P}$ be ontologies, G an RDF graph that is $\mathcal{P}$-annotated, and $\mathcal{R}$ be a set of property shortcut and class mapping rules from $\mathcal{P}$ to $\mathcal{V}$.

A class assertion $C(a)$ is *generated* by $\langle \mathcal{P}, G, \mathcal{R}\rangle$ if there is a class mapping rule $D(x) \rightarrow C(x) \in \mathcal{R}$ and $\mathcal{P} \cup \mathcal{A}(G) \models D(a)$. Property assertions generated by $\langle \mathcal{P}, G, \mathcal{R}\rangle$ is also defined similarly: they are instantiation of the head of some property shortcut rule in $\mathcal{R}$ whose body is entailed by $\mathcal{P} \cup \mathcal{A}(G)$.

We then define the *ABox generated* by $\langle \mathcal{P}, G, \mathcal{R}\rangle$ to be the set of all class and property assertions generated by $\langle \mathcal{P}, G, \mathcal{R}\rangle$.

Definition 5.7

Let $\mathcal{V}$ be a flat ontology, $\mathcal{G}$ be a schema graph consistent with $\mathcal{V}$, and $\mathcal{P}$ be a content pattern. We say that the tuple $\langle \mathcal{V}, \mathcal{G}\rangle$ is a *consumer view* over $\mathcal{P}$ if there exists a nonempty set $\mathcal{R}$ such that

(i) $\mathcal{R}$ only contains property shortcut rules from $\mathcal{P}$ to $\mathcal{V}$ and class mapping rules from $\mathcal{P}$ to $\mathcal{V}$;

(ii) for every $\mathcal{P}$-annotated RDF graph H consistent with $\mathcal{P}$, the ABox $\mathcal{A}$ generated by $\langle \mathcal{P}, H, \mathcal{R}\rangle$ satisfies the following:

- $\mathcal{A}$ is consistent with $\mathcal{V}$

- for every three assertions $C(a), R(a,b), D(b)$ in $\mathcal{A}$, there is a triple $\langle C', R', D' \rangle \in \mathcal{G}$ such that $\mathcal{V} \cup \mathcal{A} \models \{C \sqsubseteq C, R \sqsubseteq R', D \sqsubseteq D'\}$.

Here, the tuple $\langle \mathcal{V}, \mathcal{G}, \mathcal{R} \rangle$ such that $\mathcal{R}$ consists of mapping rules satisfying the above conditions is called a *consumer view implementation* or *pattern contraction*.

So, a consumer view over a pattern is obtained by associating a flat ontology with a set of property shortcut and class mapping rules such that any data generated through the rules are consistent with both the flat ontology and its corresponding schema graph. Moreover, note that we do not require that the union $\mathcal{V} \cup \mathcal{P}$ is consistent, but rather, we only need the ABox assertions generated from $\mathcal{P}$ via $\mathcal{R}$ is consistent with $\mathcal{V}$.

Example 5.8

Consider the pattern $\mathcal{O}$ in Fig. 5.3 and a flat ontology $\mathcal{V}$ given in Fig. 5.4. Let $\mathcal{G}$ be a consistent schema graph for $\mathcal{V}$ defined to be the set $\{\langle \texttt{chess:ChessGame}, \texttt{chess:hasWhitePlayerName}, \texttt{xsd:string} \rangle\}$. We define the following rule r:

$$\texttt{chess:ChessGame}(x) \wedge \texttt{chess:hasWhitePlayer}(x, y) \wedge \texttt{chess:Agent}(y)$$
$$\wedge\, \texttt{chess:hasName}(y, z) \wedge \texttt{xsd:string}(z) \rightarrow \texttt{chess:hasWhitePlayerName}(x, z)$$

Then, $\langle \mathcal{V}, \mathcal{G} \rangle$ is a consumer view over $\mathcal{O}$ due to the set of rule $\{r\}$.

This notion of consumer view can be rather straightforwardly extended to collection of patterns. That is, one can define a consumer view over a pattern collection in a similar vein by allowing the body of the property shortcut rules to include a property that "spans" across different patterns. The idea of consumer view is that it provides a simpler structure for consumers to query against. In practice, a consumer view would only be useful in a data integration system if it is made available as a

consumer view implementation. Also, given a pattern or a collection of ones, one can build different views on top of them depending on the application needs.

All of the above implicitly assume that the patterns are populated. However, as indicated at the end of Chapter 4, one of the difficulties encountered during data integration was the fact that the patterns are *more* complicated than the conceptualization that *some* of the data providers have in mind. Now, since we can create a consumer view to make things simpler for data consumers, can we then make something similar for data producers? The answer is yes, as discussed in the next section.

5.3 Producer View

If a consumer view is designed from the perspective of data consumers, a *producer view* is designed from the perspective of data providers or producers. In the consumer view, shortcuts were defined over the structure of a pattern, hence yielding a simplified structure. In the producer view, the intuition is that data producers also prefer such a simplified structure. So, the question is can data producers populate the pattern indirectly by populating the simplified structure? That way, we could employ such a simplified structure as an intermediate schema between the data providers and the patterns. Technically, this amounts to the question: if one populate the shortcuts, how can one generate data to populate the pattern?

We use pretty much the same definitions with some modifications. First of all, we again base it on the same notion of flat ontology (Definition 5.1), and population of ontology (Definition 5.5). The definition of schema graph (Definition 5.2) for flat ontology is also used. The mapping rules are slightly different though.

Definition 5.9

Let $\mathcal{P}$ and $\mathcal{V}$ be two ontologies. A *property generating rule* from $\mathcal{V}$ to $\mathcal{P}$ is a tuple-generating dependency of the form:

$$C_1(x) \wedge R(x,y) \wedge C_2(y)$$
$$\rightarrow \exists x_1 \ldots \exists x_{n-1}.\Big(D_0(x) \wedge P_1(x,x_1) \wedge D_1(x_1) \wedge \cdots \wedge P_n(x_{n-1},y) \wedge D_n(y)\Big)$$

where $n \geq 1$, R is either an object property expression or a data property in $\mathcal{V}$, C_1 is a class name or $\top$ in $\mathcal{V}$, C_2 is a class name or $\top$ in $\mathcal{V}$ if R is an object property expression, and a datatype in $\mathcal{V}$ if R is a data property, $P_1, \ldots, P_{n-1}$ are object property expressions in $\mathcal{P}$, $D_0, \ldots D_{n-1}$ are class names or $\top$ in $\mathcal{P}$, and it holds that

- if R is an object property expression, then P_n is an object property expression and D_n is a class name or $\top$ in $\mathcal{P}$; and we call the rule an *object property generating rule*
- if R is a data property, then P_n is a data property and D_n is a unary datatype in $\mathcal{P}$; and we call the rule a *data property generating rule.*

Note that if $n = 1$, a property generating rule would have no existentially quantified variables on its head.

The semantics of property generating rules is a standard first-order semantics defined in the manner of Definition 5.4.

The main idea of a producer view is that from the data annotated with the flat ontology, property generating rules generate ABox assertions using fresh, anonymous individuals, and these ABox assertions need to be consistent with the pattern. Again, there is no need for the flat ontology and the pattern to be consistent with each other.

Definition 5.10

A *generalized ABox* contains class assertions and property assertions such that anonymous individuals are allowed to occur in any of the assertions in it (instead of only named individuals).

Let $\mathcal{V}$ be a flat ontology, $\mathcal{G}$ be a schema graph consistent with $\mathcal{V}$, and H be a $\mathcal{V}$-annotated RDF graph consistent with $\mathcal{V}$. We say that H *conforms* to $\mathcal{G}$ if

the ABox $\mathcal{A}(H)$ induced by H satisfies the following: for every three assertions $C(a), R(a,b), D(b) \in \mathcal{A}(H)$, there exists a schema graph triple $\langle C', R', D' \rangle \in \mathcal{G}$ such that $\mathcal{V} \models \{C \sqsubseteq C', R \sqsubseteq R', D \sqsubseteq D'\}$.

So, let $\mathcal{V}$ and $\mathcal{G}$ be as above, and H be a $\mathcal{V}$-annotated RDF graph that is consistent with $\mathcal{V}$ and conforms to $\mathcal{G}$. A class assertion generated by $\langle \mathcal{V}, H, \mathcal{R} \rangle$ is defined as in Definition 5.6. For property generating rule r of the form defined in Definition 5.9, we say that r is applicable whenever there exists individual a and individual/literal b such that $\mathcal{V} \cup \mathcal{A}(H) \models C_1(a), R(a,b), C_2(b)$. For each aforementioned pair a, b such that r is applicable, *generalized ABox assertions generated from* $\mathcal{V}, H$, and r are $D_0(a), D_1(b_1), \ldots, D_{n-1}(b_{n-1}), P_1(a, b_1), \ldots, P_n(b_{n-1}, b)$ where $b_1, \ldots, b_{n-1}$ are freshly generated anonymous individuals. The *generalized ABox generated from* $\langle \mathcal{V}, H, \mathcal{R} \rangle$ is the set of containing:

- for each class mapping rule r, the class assertion generated from $\mathcal{V}, H$ and r;
- for each property generating rule r, the generalized class assertions and property assertions generated from $\mathcal{V}$, H, and r.

Definition 5.11

Let $\mathcal{V}$ be a flat ontology, $\mathcal{G}$ a schema graph consistent with $\mathcal{V}$, $\mathcal{P}$ a content pattern. We say that $\langle \mathcal{V}, \mathcal{G} \rangle$ is a *producer view* of $\mathcal{P}$ if there exists a nonempty set $\mathcal{R}$ such that

(i) $\mathcal{R}$ only contains class mapping rules from $\mathcal{V}$ to $\mathcal{P}$ and property generating rules from $\mathcal{V}$ to $\mathcal{P}$;

(ii) for every $\mathcal{V}$-annotated RDF graph H that is consistent with $\mathcal{V}$ and conforms with $\mathcal{G}$, the generalized ABox $\mathcal{A}$ generated by $\langle \mathcal{V}, H, \mathcal{R} \rangle$ is consistent with $\mathcal{P}$.

The tuple $\langle \mathcal{V}, \mathcal{G}, \mathcal{R} \rangle$ such that $\mathcal{R}$ satisfies the above conditions is called a *producer view implementation* or *view expansion*.

We could create a producer view by essentially "reversing" the arrow of property shortcut rules.

Example 5.12

Consider the pattern $\mathcal{O}$ in Fig. 5.3 and a flat ontology $\mathcal{V}$ given in Fig. 5.4. Let $\mathcal{G}$ be a consistent schema graph for $\mathcal{V}$ defined to be the set $\{\langle \texttt{chess:ChessGame}, \texttt{chess:hasWhitePlayerName}, \texttt{xsd:string}\rangle\}$. We define the following rule r:

$$\texttt{chess:ChessGame}(x) \wedge \texttt{chess:hasWhitePlayerName}(x, z)$$
$$\rightarrow \exists y.\Big(\texttt{chess:ChessGame}(x) \wedge \texttt{chess:hasWhitePlayer}(x, y) \wedge \texttt{chess:Agent}(y) \wedge \texttt{chess:hasName}(y, z)\Big)$$

Then, $\langle \mathcal{V}, \mathcal{G}\rangle$ is a producer view over $\mathcal{O}$ due to the set of rule $\{r\}$.

The main benefit of having producer view is that it acts as an intermediary schema between data producers and the patterns. The generalized ABox assertions generated by the mapping rules constitute the piece of enriched data that conforms to the patterns. Understanding this as RDF graph, the anonymous individuals generated in this process can be represented by blank nodes. So, in principle, data producers can now publish their data to the patterns without actually generating all the complications existing in the patterns. Note that in real applications, these anonymous individuals or blank nodes may need to be replaced with concrete URIs, but data producers can ignore this, and leave the decision to the mediator.

Also, like consumer views, multiple producer views can be created for a pattern or pattern collection. Consequently, data producers may use their own independently developed producer view implementation.

5.4 Expressing the Mapping in OWL and SPARQL

To incorporate producer and consumer views into an ontology-pattern-based data integration framework, it would beneficial to think whether it is possible to express the mapping rules as OWL axioms or SPARQL queries. In particular, if the former is possible, one can actually distribute the mapping rules as a separate OWL ontology. This idea can in fact be thought of as an example of Alignment Pattern. Meanwhile, for the latter, there is no standard way to distribute SPARQL queries in the manner of distributing OWL ontologies or RDF documents. However, a collection of such SPARQL queries would certainly be useful for implementation of a data integration framework.

Let us consider the class mapping rule first. The class mapping rule is of the form

$$C(x) \rightarrow D(x)$$

where C is a class name in the source and D is a class name in the target. For consumer view, the source is the pattern and the target is the consumer view, while for producer view, the direction is the opposite. It is obvious that the rule corresponds to the axiom

$$C \sqsubseteq D$$

In addition, the mapping rule above can also be expressed in SPARQL with the following form:

```
CONSTRUCT { ?x rdf:type t:D . }
WHERE { ?x rdf:type s:C . }
```

where we use the namespace prefix `s:` to indicate the source and `t:` to indicate the target.

The case for property shortcut rule is more complex, but still possible in OWL with caveat. To see why this is the case, we introduce the idea of *typecasting* in

OWL, which was introduced by Krisnadhi et al. [81]. Typecasting in OWL refers to conversion between multiple representational choices of using either individual, class, or property to represent a particular notion. For our needs here, we need to typecast from class to property. Consider the following example of a property shortcut rule:

$$A(x) \wedge P(x,y) \wedge B(y) \wedge Q(y,z) \wedge C(z) \rightarrow R(x,z)$$

where A, B, C are class names and P, Q are object property expressions. Using the so-called *rolification* [79, 113], we can introduce the following axioms where R_A, R_B, and R_C are fresh object properties. These axioms express the semantics of the above property shortcut rule.

$$A \equiv \exists R_A.\mathsf{Self}$$

$$B \equiv \exists R_B.\mathsf{Self}$$

$$C \equiv \exists R_C.\mathsf{Self}$$

$$R_A \circ P \circ R_B \circ Q \circ R_C \sqsubseteq R$$

Here, the class A, B, and C are typecasted into new properties R_A, R_B, and R_C. Note, however, that property chains in any OWL ontology must satisfy regularity restriction [96]. So, the above typecasting is usable as long as the introduced properties and property chain do not lead to violation of that restriction.

The property shortcut rule can also be easily expressed in a SPARQL query. For example, the property shortcut rule presented above can be written as the following SPARQL query:

```
CONSTRUCT { ?x t:R ?z . }
WHERE {
  ?x rdf:type s:A ;  s:P ?y .
  ?y rdf:type s:B ;  s:Q ?z .
  ?z rdf:type s:C .
}
```

The above method can still be applied even when the property shortcut rule is of the following form where the head is a conjunction of atoms.

$$A(x) \wedge P(x,y) \wedge B(y) \wedge Q(y,z) \wedge C(z) \rightarrow D(x) \wedge R(x,z) \wedge E(z)$$

The reason is because the above rule is equivalent to the following three rules:

$$A(x) \wedge P(x,y) \wedge B(y) \wedge Q(y,z) \wedge C(z) \rightarrow R(x,z)$$

$$A(x) \wedge P(x,y) \wedge B(y) \wedge Q(y,z) \wedge C(z) \rightarrow D(x)$$

$$A(x) \wedge P(x,y) \wedge B(y) \wedge Q(y,z) \wedge C(z) \rightarrow E(z)$$

In OWL, we can express the above three rules together via the following axioms:

$$A \equiv \exists R_A.\mathsf{Self}, \quad B \equiv \exists R_B.\mathsf{Self}, \quad C \equiv \exists R_C.\mathsf{Self}$$

$$R_A \circ P \circ R_B \circ Q \circ R_C \sqsubseteq R$$

$$A \sqcap \exists P.(B \sqcap \exists Q.C) \sqsubseteq D$$

$$C \sqcap \exists Q^{-}.(B \sqcap \exists P^{-}.A) \sqsubseteq E$$

In SPARQL, the solution is even simpler via the following query:

```
CONSTRUCT {
  ?x rdf:type t:D ; t:R ?z .
  ?z rdf:type t:E .
}
WHERE {
  ?x rdf:type s:A ;  s:P ?y .
  ?y rdf:type s:B ;  s:Q ?z .
  ?z rdf:type s:C .
}
```

Finally, for property generating rules, the situation is different because they cannot be expressed in OWL, though it is still possible in SPARQL. For example, consider the property generating rule:

$$D(x) \wedge R(x,z) \wedge E(z) \rightarrow \exists y.\Big(A(x) \wedge P(x,y) \wedge B(y) \wedge Q(y,z) \wedge C(z)\Big)$$

where this time, D, R, and E belong to the source and A, P, B, Q, and C belong to the target. The above rule can essentially be expressed by the following SPARQL query, which generates blank nodes for each existential variable in the rule:

```
CONSTRUCT {
  ?x rdf:type t:A ;
     t:P [  rdf:type t:B ;
            t:Q ?z ].
  ?z rdf:type t:C .
}
WHERE {
  ?x rdf:type s:D ; s:R ?z .
  ?z rdf:type s:E .
}
```

5.5 Views for GeoLink Patterns

The idea of pattern view above is employed in the GeoLink data integration project. Concretely, we develop a producer view that turned out to be suitable for most of the data providers involved in the project. As a result, the data integration currently only needs one producer view. The latest version of this producer view is not yet released, but available online on GitHub[1]

For example, the view defines two classes: `Person` and `Cruise`, and one object property `hasChiefScientist`. The domain and range restriction of this property indicates that `Person` is its range and `Cruise` is its domain. Here, a data provider X may populate the view, say churning out the triples in Fig. 5.6.

```
x:cr1 a view:Cruise ;
      view:hasChiefScientist x:pr1 .
x:pr1 a view:Person .
```

Figure 5.6: Example of data based on GeoLink view

[1] https://github.com/ec-geolink/design/blob/master/patterns/producer-centric-views/main.owl

Meanwhile, the Cruise pattern as depicted in Fig. 4.7 employs Agent Role pattern to model involvement of agents, including persons. So, a view expansion to the pattern can be essentially expressed as a CONSTRUCT query in Fig. 5.7. Here, the assumption is of course that the property hasChiefScientist corresponds to ChiefScientistRole in the Cruise pattern.

```
PREFIX view: <http://schema.geolink.org/dev/view#>
CONSTRUCT {
  ?x a :Cruise ;
     :providesAgentRole _:bn1 .
  _:bn1 :isPerformedBy ?y ;
     a :ChiefScientistRole .
  ?y a :Person .
} WHERE {
  ?x a view:Cruise ;
     view:hasChiefScientist ?y .
  ?y a view:Person
}
```

Figure 5.7: View expansion example with CONSTRUCT statement applied to triples in Fig. 5.6. For simplicity, patterns are assumed to reside in the default namespace.

The pattern contraction in the opposite direction can be obtained from the view expansion in Fig. 5.7 by swapping the graph patterns in the WHERE clause with the graph patterns in the CONSTRUCT clause (with some adjustment in the namespace definition, of course). Using CONSTRUCT statement for view expansion, however, does have some problem particularly if we want to adhere to Linked Data principles. That is, the view expansion generates a blank node, which will be published as an instance of ChiefScientistRole. Ideally, this instance should have a concrete URI, not a blank node. This can be alleviated by the use of `UUID()` function in SPARQL 1.1, though the result still needs to be converted into HTTP URI, which requires a rather complicated string literal processing in the WHERE clause. As a workaround, when some data are published via the view, we need to run one post-processing step to convert the generated blank nodes into concrete URIs.

5.6 Discussion

In this chapter, we have introduced the notion of pattern view to bridge the conceptualization gap between data providers and patterns. We have demonstrated how pattern views (both producer and consumer views) can act as intermediary schema, particularly for producer views. The structure of the view is based on a flat ontology. Together with an appropriate schema graph, this allows data producers to grasp the schematic structure of the RDF graphs they need to procure to populate the patterns, while at the same time, they are shielded from the complexity of the pattern collection, which actually acts as the global schema of the data integration framework.

By having producer views and consumer views, the architecture of the data integration may now consist of four layers. At the bottom layer, we still have all the data sources. Producer views would then reside on top of data sources. The producer views should be developed to reflect local conceptualization on the data producers' side. There can be multiple producer views, and one producer view may also be shared by multiple data producers. In the next layer, we have a limited set of content patterns acting as the global schema. Above this layer, one could add a number of consumer views as needed. In relation to Hypothesis 2 as stated in Section 1.2, this chapter provided at least half of the answer. What remains is assessing how helpful the pattern view is for our data integration framework. This will be topic of discussion in Chapter 6.

Certainly, there is still quite a lot of work that needs to be done with regards to this notion. Earlier studies by Bachtarzi et al. [7], Hung et al. [69], and Volz et al. [131] considered views in the context of more efficient access to RDF data. Meanwhile, Cho et al. [31] studied views in the context of access control to RDF data. Another future direction worth pursuing beyond this thesis is the issue of data validation and

integrity checking with respect to patterns, particularly in the light of the recent W3C public working draft over SHACL as part of the effort toward standard language for checking constraints on RDF graphs [77]. In this case, the question also includes whether having pattern view would help or jeopardize the task.

6 Evaluation

In this chapter, we discuss the assessment that we did regarding the use of pattern view in a data integration framework.

6.1 Qualitative Evaluation: Rationale

The objective of the evaluation is to assess whether providing pattern view for data providers really helps bridging the gap between abstraction in the patterns and local conceptualization at the data providers' side. Performing a truly comprehensive evaluation over this objective is problematic at least due to two reasons.

First, it is hard to quantify the advantages of using pattern view with patterns over not using one. This is partially caused by the motivation of having pattern view in the first place, which is to act as an intermediate abstraction between patterns and data providers such that publishing data via pattern views is *easier* than to the patterns directly. The easiness in this context is a quality that largely depends on human judgements and is affected by a myriad of factors, e.g., familiarity with the notions modeled in the patterns and pattern views, availability of technological infrastructure as well as familiarity of the human operating it, etc. It is thus hard to design a user study that is truly comprehensive.

Second, this dissertation is written on the background of an ongoing project. To evaluate the above objective, one needs to compare two separate data publishing effort by the same data providers, and this is something that data providers would be unwilling to do if it is only for the sake of evaluating the publishing the *same* set

of data.

An alternative approach is to obtain an evaluation story by unearthing the experience of the data providers when publishing data as part of the project activities. Such a story can be collected through a qualitative study. This is possible in principle because from the way the project has gone, some of the data providers have attempted to publish against the patterns directly, and now with the availability of pattern views, they republish or are in the process of republishing against the pattern views. Obviously, a full and comprehensive evaluation story can be obtained only after the project complete, which is beyond the timeline of this dissertation work. Our choice is thus to essentially conduct a mini version of the comprehensive qualitative study, which would provide sufficient evidences to full the objective mentioned in the beginning of this chapter.

6.2 Data Collection Procedures

The qualitative data were obtained using structured interviews with a mixed of open-ended and close-ended questions. The questions were designed to draw out the participants' experience during their involvement in the GeoLink data integration project, which is used as the background of this research work. The interviews themselves were conducted online via Google Hangouts and Skype calls. Conversations during the interviews were recorded with permission from the participants.

There were three separate interviews to three participants, each was conducted roughly for 20-25 minutes. The first one was conducted on November 13, 2015, the second was on November 17, 2015, and the third was on November 18, 2015. The three participants represent three different repositories maintained by separate institutions. They were selected out of seven partner repositories of the GeoLink project because compared to the rest of the partners, they have been involved in this data integration

project the longest, including the time they participated in the OceanLink project, which preceded the GeoLink project.

In terms of familiarity with data in their corresponding repositories, all three participants are comparable since they all have hands-on involvement in designing conceptual model of the data, maintaining the data infrastructure, as well as, performing various activities related to curation and quality control. In terms of technical background, the first participant has more than 10 years experience as a software engineer, including a few years working with linked data technologies prior to his involvement in the OceanLink and GeoLink projects. The second participant has extensive experience as data scientist, including a good amount of experience in linked data technology. The third participant has more than 20 years of experience as a programmer, analyst, and database administrator, including several years of using linked data technologies. All but the second participant have a limited experience in ontology modeling prior the GeoLink/OceanLink project, and they have only little knowledge regarding ontology languages such as OWL. Meanwhile, the second participant was experienced in working with domain ontologies and modeling them using OWL. None of the three participants has an extensive experience in ontology pattern modeling prior to this project.

6.3 Questions

The qualitative evaluation is mainly guided by the second research question from Section 1.2: "How do we address the gap between the abstraction within ODPs and the local conceptualization on the data providers side, so that data publishing can be done according to Linked Data principles, retaining the simplicity, while keeping the benefits of semantic interoperability from ODPs?"

To answer this question, in Chapter 5, we have presented pattern views, which can act as intermediary schema between ODPs and the local conceptualization of

data providers. The pattern views were defined in terms of flat ontologies, schema graphs, and the mapping rules, consisting of class mapping, property shortcut, and property generating rules. For data publishing, producer views were introduced as a pattern views that enables more complicated RDF graph structures generated from the simple ones. As a result, data producers only need to know a simpler structure than the one modeled in the patterns, and thus, appealing to the simplicity aspect of linked data publishing. We seek to confirm this intuition through a qualitative evaluation in this chapter.

In a more general picture, the interview questions were formulated to capture comparison of experience of using ODPs with and without views. This led to the following list of questions:

1. "What did you find most difficult when using ODPs without views for data population/publishing?"
2. "What did you find most difficult when using ODPs with views for data population/publishing?"
3. "What did you find easier when (or like most about) using ODPs without views for data population/publishing?"
4. "What did you find easier when (or like most about) using ODPs with views for data population/publishing?"
5. "Which of the two approaches (ODPs without views and ODPs with views) is the more difficult one for populating with data? Which approach do you prefer?"
6. "If you are a data provider and you are confronted with a new set of ODPs, would you think that the view significantly helps you?"
7. "Would the views help average data providers?"
8. "Would the views help data consumers (people from your community) as well?"

9. "Technically or conceptually, did you lose anything when using ODPs without views?"

6.4 Participants' Responses

We present the responses from the participants' of the interviews. We denote the first participant as P1, second participant as P2, and third participant as P3.

Question 1. "*What did you find most difficult when using ODPs without views for data population/publishing?*"

P1: Aligning our own conceptual understanding of how the information is represented to what is in the patterns, for example, understanding what information objects are and why they are important. Once they are understood, the ODPs really make a lot of sense. In terms of publishing, it is technically not too difficult once we have the conceptual understanding regarding the aligned patterns. It is only more work since there are more triples than expected. The gap between our conceptual understanding and the patterns is not that big. If anything, it highlighted few things that we did wrong and could have done better. It is very helpful to see it fleshed out in a more normalized way with the ODPs. As a software developer, I can write an even more lightweight software against these ODPs than the business logic that I still have to encode inside software with our local ontology.

P2: There are so many different pieces and it is difficult to understand how the different pieces would go together. Some pieces are reused in many other components, which are one of the benefits. Nevertheless, it is hard to understand what piece of information we need to supply to fill out these pieces. It was a bit daunting to see what was common and reusable and what was not.

P3: [Most difficult is] the fact that I could not easily see the relationships between the patterns. To me, the great power of the view is that it realizes the relationships between the patterns. It is analogous to my database schema. You cannot get these relationships easily without the views, because I have never a good way (or tools) to visualize that, and also, some of the patterns are rather abstract, though the rationale is well-understood, they are not mapped easily to the concepts in the data.

Question 2. *"What did you find most difficult when using ODPs with views for data population/publishing?"*

P1: Nothing really.

P2: I found the views a very streamlined approach to ODPs. We (data providers) just need to supply the minimal information needed and it will be converted into the full-blown pattern. This is, in particular, a big step for geoscientists not to worry what is going on in the backend. Views are actually easier to use because it is not about the number of classes and properties, rather, it was how the relationships would bring the pieces together. Also, once you achieve that, there is also the technical hurdle of how to get the properties and relationships correct.

P3: I misunderstood the use of the view on the input side vs. the output side. Originally, we populate the patterns (and we tried), and then create a view on the output, but then we latter realize we need a view on the input side. We probably wasted some time trying to publish data to the raw pattern. As a curator, I must think about the input view. Also, later I realize that input and output views are analogous, because the way I design my database reflects the science perspective to satisfy the needs of the scientists, not engineering perspective.

Question 3. *"What did you find easier when using ODPs without views for data population/publishing?"*

P1: In terms of integrating with other repositories, we could avoid doing it multiple times, each time with a particular repository via a specifically designed solution. Rather, doing it as a group and coming up with a mediation layer that everyone agrees upon was really helpful to understand how we all fit together.

P2: I could expose the pieces of my dataset that aligned or were best used within the patterns, contrasted that with the domain ontology: I would either have to conform with everything in the ontology or not use the ontology. I could exclude those parts of my data that are not relevant for the integration.

P3: My previous experience was only with SKOS concept schemes. Moving to OWL and patterns felt better because we are correcting many inconsistent use of SKOS. With the richness of OWL, we can define arbitrarily many relationships, rather than only a limited number if SKOS is used.

Question 4. *"What did you find easier when (like most about) using ODPs with views for data population/publishing?"*

P1: Apart from publishing faster (the views are almost like a mediation layer between us and the patterns), it allows for a quicker grasp of the conceptualization.

P2: In terms of publication, ODPs with views brings an improvement of time and effort of getting data published: much easier than publishing against the full pattern. Also, it gives the ability to show to end users different aspects of the data, or present the data that make sense to a particular community. Also, all benefits of using ODPs are still present.

P3: The ability to focus on a single concept is rather new to me, because when we define a relational schema, everything is to be related in the beginning.

Question 5. *"Which of the two approaches (ODPs without views and ODPs with views) is the more difficult one for populating with data? Which approach do you prefer?"*

P1: In terms of difficulty, they are not too much different technically. We prefer ODPs without views only because we already understood the aligning between our conceptualization with the patterns, and that was what we started with. Having views means another exercise of aligning it with our conceptualization.

P2: ODPs without views is more difficult. In domain like geosciences, the difficulty is increased, because end users from such domains would have difficulties in expressing data in the full logic and semantic statements, and the views take this burden away from them. In terms of preference, I would prefer ODPs with the views. Nevertheless, I think we need to know what the full patterns would look like in order to adequately express the views.

P3: It would be more difficult to work with the patterns without the views because some of us have systems rooted in relational schema. ODPs with views is thus naturally preferred here.

Question 6. *"If you are a data provider and you are confronted with a new set of ODPs, would you think that the view significantly helps you?"*

P1: Yes, even if it is just because of coming to terms with the conceptual model, it is easier to understand.

P2: Yes. Most of the time you are not going to align completely with the other patterns. The views give a nice intermediary of determining what aspects to align with, and doing it via views would be easier and probably, more efficient.

P3: Patterns reside in different namespace. The views practically collapse all of the concepts into one namespace and it is much easier for creating RDF graphs.

Question 7. *"Would the views help average data providers?"*

P1: Yes. Certainly.

P2: Yes. They would find an integration via views a bit easier than without them.

P3: I hope so. Some providers could ignore some sections of the views. To me, it is more practical to create a single view in the beginning, and amend it gradually, rather than creating separate views for every data provider. Of course, over time it is possible that a new view is more appropriate in the future if GeoLink expands to other areas such as biology, etc.

Question 8. *"Would the views help data consumers (people from your community) as well?"*

P1: Yes. It is more palatable, easier for them to understand, adopt, and come aboard.

P2: Yes. For consumers specifically, we can present the bigger picture to them in a specialized way. Exposing also the CONSTRUCT queries (i.e., mapping rules) provides additional benefit in terms of helping the understanding, but not nearly helpful for them create the queries themselves.

P3: Yes. As data provider, I must ultimately ensure that I provide all the necessary data to satisfy the consumer view, so producer and consumer views may practically be the same. If I have a data model in mind that is different than a scientist who is my consumer, then I make a big mistake because that means I do not understand my community.

Question 9. *"Technically or conceptually, did you lose anything when using ODPs without views?"*

P1: A little bit, but that is only if the sole objective was to get all of our content exposed through GeoLink integration, and this is not necessarily an important

objective of the project itself. It is important to be okay with not needing to expose everything. For example, not all of our "Deployment", which is on the same level as a cruise are exposed, and that is okay for now and can be resolved later. What is really helpful to our commmunity may not be as important for discovery for the larger community.

P2: I don't think I lost anything. With the views, we are still expressing the same information in the same way, but in a more convenient fashion. There is a bit of learning curve in doing it against the full patterns, whereas with the views, it seems a lot more straightforward.

6.5 Findings and Discussion

Recall that the aim of the qualitative evaluation is to find out whether equipping the patterns with pattern views really helps data providers to realize the data integration. Findings obtained from the responses of the evaluation participants provide evidences that it is indeed the case.

1. *Is there really a benefit in using ontology design patterns (as a global schema) for data integration?* Particularly from responses of Question 3, we understand that using ontology design patterns offers some benefits over using domain ontology or weaker formalisms such as SKOS. With respect to the former, P2 stated from his experience that when working with domain ontologies, either one has to use the whole domain ontology at hand completely or does not use it at all. In contrast, P2 stated that there is *flexibility* if patterns are used: one can choose to expose only some parts of his data that are aligned or best used within the patterns. On the other hand, compared to weaker or less formal framework such as SKOS, patterns that are encoded with an ontology language such as OWL offer *richer semantics* as stated by P3. Meanwhile, P1 pointed out the general

benefit of data integration via mediation, as opposed to peer-to-peer, which is not limited to data integration framework based on patterns. So, overall, the answer to the question is affirmative.

2. *Is there a problem arising when patterns are used as a global schema for data integration?* Looking at responses from Question 1, both P2 and P3 mentioned the difficulties of understanding the larger picture. In particular, P2 and P3 found it hard to grasp how different patterns in the collection are related to each other to form the global schema. P2 stated that it is hard to distinguish patterns that are actually common and reusable, and to figure out what piece of information is needed to populate the patterns with data. Meanwhile, P3 mentioned that the whole collection of patterns is hard to visualize and also, some patterns are rather abstract and not easily mapped to concepts internal within P3's conceptualization. P1 also mentioned some gap of abstraction that needs to be overcome. However, P1's opinion is that the gap is not too big and he was able to surmount it, so he was able to populate the patterns directly. So, the answer of this question is also affirmative where the problem is that there is indeed a gap of abstractions between the patterns and the data providers' conceptualization, and more data providers were bothered by it.

3. *Can the aforementioned gap of abstraction be bridged by having the pattern views?* We look at responses of Question 2 and 4 together. From responses of Question 4, despite not mentioning such a gap of abstraction, P1 and P2 mentioned that having pattern views can speed up as well as lessen the effort in getting the data published. P2's response to Question 2, although did not address Question 2 directly, did in fact affirm that the aforementioned gap of abstraction can be bridged by having pattern views. He argued that with pattern views, one only needs to supply a minimal amount of information as needed to populate the full blown structure of the patterns without worrying

about what is going on in the back-end. He also mentioned that pattern views are easier to use because they show how relationships between patterns bring the pieces together. So, our answer to the question is also affirmative.

4. *Is publishing data directly to the patterns more difficult than via pattern views?* From responses of Question 5, P2 and P3 agreed that publishing data directly to the patterns without views is more difficult than publishing them via pattern views. P2 thought that the difficulty is even increased in a domain like geosciences where end users would not be familiar with formalisms used in expressing patterns. P3 stated that working without pattern views is more difficult because pattern views appear to be closer to relational modeling, which he is much more familiar with. Meanwhile, P1's response is slightly different. P1 believed that the difficulty is not much different between the two. Since P1 was able to grasp the conceptualization within the patterns, as mentioned in his response to Question 1, P1 was of the opinion that having pattern views simply means another exercise of aligning them with P1's conceptualization. So, the answer to this question is indeed publishing data directly to the patterns is typically more difficult than via pattern views. P1's case is rather special given that he was better than the others in understanding the patterns' conceptualization.

5. *Do pattern views help data providers who joined the data integration effort late?* All P1, P2, and P3 answers affirmatively to this question based on their responses to Question 6. All three thought that pattern views are easier to understand and relate to conceptual model of the data providers who join late.

6. *Do pattern views help data providers in general?* P1, P2, and P3 also gave a positive answer based on responses to Question 7.

7. *Do pattern views help data consumers?* Again, P1, P2, and P3 gave a positive answer based on responses to Question 8.

In summary, we can see that pattern views are helpful for data integration framework based on ontology patterns. All representatives of the data providers interviewed for this evaluation concurred that in general, having pattern views together with the patterns is preferable for average data providers. Moreover, pattern views can also be used to help data consumers viewing the data in a more simplified structure. Thus, we now have completely fulfilled Hypothesis 2 (page 9) and by doing it, we also fulfill Hypothesis 1 (page 8).

7 Conclusion

This research work aims to demonstrate the use of ontology design patterns for cross-repository data integration. We summarize the contributions of this dissertation in the effort to achieve this objective. We then point out possible directions of future work.

7.1 Summary

This dissertation presented our three main contributions described below.

7.1.1 Ontology Pattern-based Data Integration Framework

An ontology pattern-based data integration framework is a data integration framework in which the global schema constitutes a set of content ontology design patterns. We demonstrated the feasibility of such a framework by describing its key components and providing evidences that these key components suffice to realize the framework in principle.

Every data integration framework can be described as a triple of a global schema, a local schema, and mapping between these two. This was formally presented in Section 3. By virtue of discussion in Section 3.2, we argued that ontology can play a role as the global schema in a data integration framework. Furthermore, in Section 3.3, we made an assumption that all data sources are in fact linked data repositories, hence simplifying the issue of syntactic interoperability.

The remaining part was to argue that content ontology design patterns can also

act as the global schema. The argument was made in Section 4.6 after presenting the second contribution of this dissertation discussed below. The feasibility is then strengthened by the third contribution of this dissertation below.

7.1.2 Content Patterns for Ocean Science

A second contribution of this dissertation work is a collection of content ontology design patterns representing a number of key notions in the ocean science. Content patterns we modeled for this dissertation are not only important in the ocean science, but they were modeled for the purpose of integrating data across repositories involved in the GeoLink project described in Section 4.3. As such, they also conceptually occur in some of the repositories.

We described in Section 4.4, a collaborative modeling approach that is suitable for pattern modeling. This approach is based on a heavily modified form of the eXtreme design method [108], which to date is the only one known systematic method for designing an ontology design pattern. This method embodies aspects such as user involvement, collaborative modeling with iterative refinement, user requirements gathering through competency questions and contextual statements, and task-oriented design focusing on the user requirements. Our approach modified the eXtreme design method as follows:

- time spent on listing all competency questions is shortened by mixing it with directly listing what information that a pattern should provide and what constraints should it satisfy;
- testing the pattern against the user requirements is done manually on-paper;
- pair design is not performed, and replaced with frequent teleconferences.

The above modification was done due to limitation in the project condition such as time constraints, less opportunities to do face-to-face meetings, lack of readily acces-

sible Linked Data infrastructure to support pattern testing, and lack of manpower to do proper pair design. This resulted in an approach illustrated by Fig. 4.2.

By following the above approach, during the project, we were able to create 17 content patterns: Agent, Agent Role, Event, Information Object, Identifier, Person, Personal Info Item, Person Name, Organization, Funding Award, Program, Place, Cruise, Platform, Vessel, Physical Sample, and Property Value. All but the last two are included in Section 4.5 and Appendix A. Note that since the GeoLink project is still ongoing, the collection of patterns actually employed in the project may grow in the future, but shall not be a part of this dissertation. Each pattern is equipped with a set of OWL axioms that fixes its meaning, and possibly, several sets of alignment axioms, which provide alignments to other patterns in the collection. Exposition of each pattern in Section 4.5 also included description of modeling choices made during the modeling, which led to some ontological commitment imposed by the pattern. The resulting patterns together form a modular ontology where each pattern can be extracted as a module.

Feedback from the data providers detailed in Section 4.6 indicated that the pattern colllection as a whole turned out to be too complicated. This prevented data providers in populating the patterns with their data. To alleviate this problem, this dissertation work proposed an idea of pattern views, which became the third contribution of this dissertation.

7.1.3 Pattern Views

The idea of pattern views was described in Chapter 5. If one understands this in the context of Fig. 3.1 where the patterns reside at the mediator layer, a pattern view is an intermediate schema between the patterns and the users (called *consumer view*) or between the patterns and the data sources (called *producer view*). A pattern view should be understood as a simplified form of a pattern for a *particular* perspective of

a user or a data provider. A pattern view should be simple enough that it is almost self-explanatory and populating a pattern view should require much less effort than populating a pattern directly. A key intuition of a pattern view is that it contains shortcuts of a number of relationships from a pattern.

Out of the two types of pattern view, consumer view was described in Section 5.2, while producer view was described in Section 5.3. Both consumer view and producer view are built from a flat ontology and schema graph of a flat ontology. A pair of a flat ontology and schema graph constitutes a consumer view of a pattern, whenever there exists appropriate mapping rules from the pattern to the flat ontology such that from any piece of data that populates the pattern, one can obtain another piece of data that populates the flat ontology by applying the mapping rules. Besides class mapping rules, mapping rules for a pattern view are called property shortcut rule. Producer view was defined similarly, but with the mapping rules going on the opposite direction. The mapping rules for producer view include the so-called property generating rule.

We then showed in Section 5.4 how to express the aforementioned mapping rules as OWL axioms and SPARQL queries, if at all possible. We also described in Section 5.5 the use of pattern views, particularly producer views in the GeoLink project. Finally, some qualitative evaluation was done to assess whether pattern views really help data providers in GeoLink project publish their data for the integration.

7.2 Future Work

The work presented in this dissertation only scratches the surface since there are a number of future work that can be pursued beyond this dissertation. They include theoretical as well as practical research directions.

First of all, although ontology based data integration has been well-researched since the 80s, it is such a hugely challenging problem with multitude of aspects and

myriad of possibilities that many people are working on it. Interestingly, the use of ontology patterns as the core part of data integration is still relatively unexplored. This dissertation, to the best of our knowledge, represents one of, if not the, initial foray into this topic. Establishing a more structured and systematic method to develop patterns for the *purpose of data integration* would be a worthwhile topic to pursue. Study about quality of patterns for data integration, as well as, methods of evaluating it are also possible research directions.

Other aspects of ontology design pattern research includes better way of documenting patterns, visualizing patterns, including visual pattern language that can help pattern development, as well as general issues of pattern quality. The use of pattern for integrity checking over data is also an interesting research area. Regarding pattern view, one could try exploring this idea and formulate a more rigorous theory. An interesting issue regarding pattern view and pattern is whether one can somehow automate the creation of one from the other, or to what extent can it be done. Investigation toward the usefulness of pattern and pattern view for linked data publishing and consumption would also be worthwhile.

A GeoLink Pattern Collection

In this appendix, we describe the set of Content Patterns that we modeled for data integration in the GeoLink project. Each pattern description follows the following structure:

1. an informal description, obtained from the pattern requirements, and some visual depiction of the pattern;
2. an axiomatization of the pattern written in the DL notation or Datalog notation, formalizing semantic relationships described in the informal description; declaration of the signature of a pattern is given in a modified form of OWL Manchester syntax;
3. description of alignment with other patterns with axiomatization;
4. miscellenaeous remarks if necessary.

A.1 Agent

A.1.1 Description

Figure A.1: The Agent pattern

The Agent pattern is a very simplistic pattern stub intended to model agents such as people or organization. Agents may perform a role (an instance of AgentRole class), e.g., in the context of events, organizations, etc. Alignments are specified with the Agent Role pattern (Appendix A.2) depicted in Fig. A.2.

A.1.2 Axiomatization

```
Prefix: : <http://schema.geolink.org/dev/agent#>
Ontology: <http://schema.geolink.org/dev/agent>
Import: <http://schema.geolink.org/dev/agent-to-agentrole>
ObjectProperty: performsAgentRole
Class: AgentRole, Agent
```

Guarded domain and range restrictions of performsAgentRole.

$$\exists \mathsf{performsAgentRole}.\mathsf{AgentRole} \sqsubseteq \mathsf{Agent} \tag{A.1}$$

$$\mathsf{Agent} \sqsubseteq \forall \mathsf{performsAgentRole}.\mathsf{AgentRole} \tag{A.2}$$

Also, we assert the following pairwise-disjointness axiom.

$$\mathsf{Agent} \sqcap \mathsf{AgentRole} \sqsubseteq \bot \tag{A.3}$$

A.1.3 Alignment Axioms

The Agent pattern specifies an alignment only with the Agent Role pattern.

Alignment with Agent Role pattern

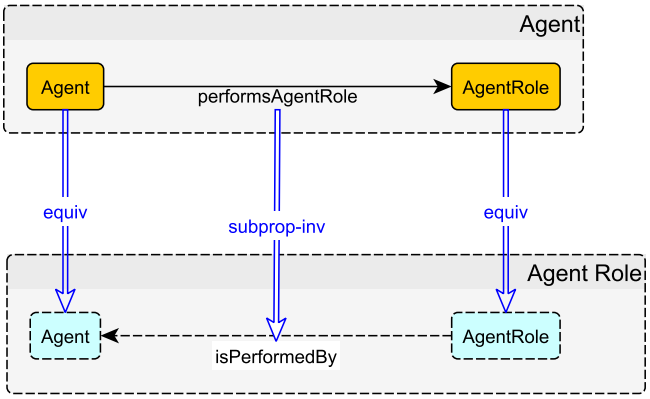


Figure A.2: Alignment of Agent to Agent Role

We align Agent and AgentRole classes in the Agent pattern to the Agent and AgentRole class in the Agent Role pattern (Appendix A.2) as depicted in Fig. A.2. We also align the performsAgentRole property to the isPerformedBy property in the Agent Role pattern by setting the former as a subproperty of the inverse of the latter.

```
Prefix: ecglag: <http://schema.geolink.org/dev/agent#>
Prefix: ecglar: <http://schema.geolink.org/dev/agentrole#>
Ontology: <http://schema.geolink.org/dev/agent-to-agentrole>
ObjectProperty: ecglag:performsAgentRole, ecglar:isPerformedBy
Class: ecglag:Agent, ecglag:AgentRole, ecglar:Agent, ecglar:AgentRole
```

$$\text{ecglag:Agent} \equiv \text{ecglar:Agent} \tag{A.4}$$

$$\text{ecglag:AgentRole} \equiv \text{ecglar:AgentRole} \tag{A.5}$$

$$\text{ecglag:performsAgentRole} \sqsubseteq \text{ecglar:isPerformedBy}^{-} \tag{A.6}$$

A.2 Agent Role

A.2.1 Description

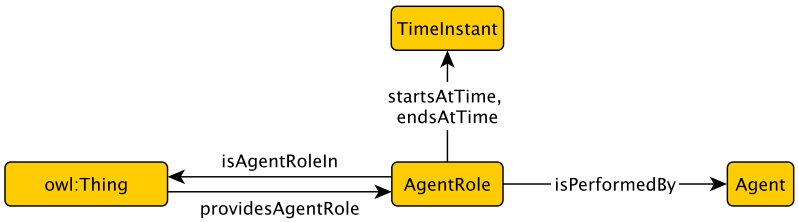


Figure A.3: The Agent Role pattern

This pattern was already described in Section 4.5.2. In short, an AgentRole is performed by exactly one Agent, has exactly one starting time and one ending time, and is an agent role in exactly one thing. We only restate the axioms below, as well as the alignment of this pattern with Agent pattern and OWL Time ontology [65] as depicted in Fig. A.4 and A.5.

A.2.2 Axiomatization

IRI Declarations

```
Prefix: : <http://schema.geolink.org/dev/agentrole#>
Ontology: <http://schema.geolink.org/dev/agentrole>
Import: <http://schema.geolink.org/dev/agentrole-to-agent>
Import: <http://schema.geolink.org/dev/agentrole-to-owltime>
ObjectProperty: isAgentRoleIn, providesAgentRole, isPerformedBy,
                startsAtTime, endsAtTime
Class: AgentRole, Agent, TimeInstant
```

Core Axioms

An AgentRole is performed by exactly one Agent, has exactly one starting time and one ending time, and is an agent role in exactly one thing.

$$\mathsf{AgentRole} \sqsubseteq (=1\ \mathsf{isPerformedBy}.\mathsf{Agent}) \sqcap (=1\ \mathsf{isAgentRoleIn}.\top)$$

$$\sqcap\ (=1\ \mathsf{startsAtTime.TimeInstant}) \sqcap (=1\ \mathsf{endsAtTime.TimeInstant}) \quad \text{(A.7)}$$

$$\mathsf{providesAgentRole} \equiv \mathsf{isAgentRoleIn}^{-} \quad \text{(A.8)}$$

We next assert the domain and range restrictions of the properties in this pattern. Specifically for the isAgentRoleIn property, since it ranges over all individuals, range restriction is not needed and its domain restriction is unguarded. For the providesAgentRole property, it is the other way around: domain restriction is not needed while its range restriction is unguarded. For the other object properties, domain and range restrictions are guarded.

$$\exists \mathsf{isPerformedBy.Agent} \sqsubseteq \mathsf{AgentRole} \quad \text{(A.9)}$$

$$\mathsf{AgentRole} \sqsubseteq \forall \mathsf{isPerformedBy.Agent} \quad \text{(A.10)}$$

$$\exists \mathsf{startsAtTime.TimeInstant} \sqsubseteq \mathsf{AgentRole} \quad \text{(A.11)}$$

$$\mathsf{AgentRole} \sqsubseteq \forall \mathsf{startsAtTime.TimeInstant} \quad \text{(A.12)}$$

$$\exists \mathsf{endsAtTime.TimeInstant} \sqsubseteq \mathsf{AgentRole} \quad \text{(A.13)}$$

$$\mathsf{AgentRole} \sqsubseteq \forall \mathsf{endsAtTime.TimeInstant} \quad \text{(A.14)}$$

$$\exists \mathsf{isAgentRoleIn}.\top \sqsubseteq \mathsf{AgentRole} \quad \text{(A.15)}$$

$$\top \sqsubseteq \forall \mathsf{providesAgentRole.AgentRole} \quad \text{(A.16)}$$

We express axiom (A.15) and (A.16) in the corresponding OWL file through the use of range and domain restrictions on the properties. Finally, we assert the following class disjointness axioms.

$$\mathsf{alldisjoint(AgentRole, Agent, TimeInstant)} \quad \text{(A.17)}$$

A.2.3 Alignment

The Agent Role pattern specifies alignments with the Agent pattern and OWL Time ontology [65].

Alignment with Agent pattern

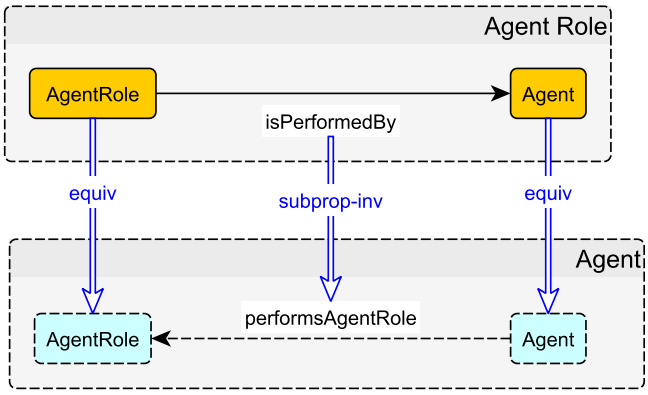


Figure A.4: Agent Role aligned to Agent

We align Agent and AgentRole classes in the Agent Role pattern to the Agent and AgentRole class in the Agent pattern (Appendix A.1) as depicted in Fig. A.4. Furthermore, we also align the performsAgentRole property to the isPerformedBy property in the Agent Role pattern by setting the former as a subproperty of the inverse of the latter.

```
Prefix: ecglag: <http://schema.geolink.org/dev/agent#>
Prefix: ecglar: <http://schema.geolink.org/dev/agentrole#>
Ontology: <http://schema.geolink.org/dev/agentrole-to-agent>
ObjectProperty: ecglag:performsAgentRole, ecglar:isPerformedBy
Class: ecglar:AgentRole, ecglar:Agent, ecglag:AgentRole, ecglag:Agent
```

$$\text{ecglar:AgentRole} \equiv \text{ecglag:AgentRole} \tag{A.18}$$

$$\text{ecglar:Agent} \equiv \text{ecglag:Agent} \tag{A.19}$$

$$\text{ecglar:isPerformedBy} \sqsubseteq \text{ecglar:performsAgentRole}^{-} \tag{A.20}$$

Alignment with OWL Time Ontology

We align TimeInstant with the time:Instant class from the OWL Time ontology as depicted in Fig. A.5.

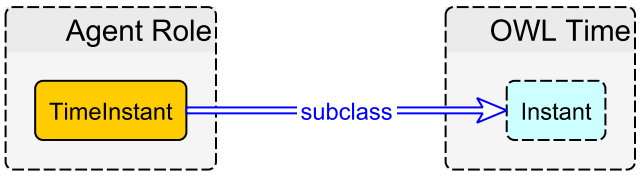


Figure A.5: Agent Role aligned to OWL Time

```
Prefix: time: <http://www.w3.org/2006/time#>
Prefix: ecglar: <http://schema.geolink.org/dev/agentrole#>
Ontology: <http://schema.geolink.org/dev/agentrole-to-owltime>
Class: ecglar:TimeInstant, time:Instant
```

$$\mathsf{ecglar{:}TimeInstant} \sqsubseteq \mathsf{time{:}Instant} \tag{A.21}$$

A.3 Event

A.3.1 Description

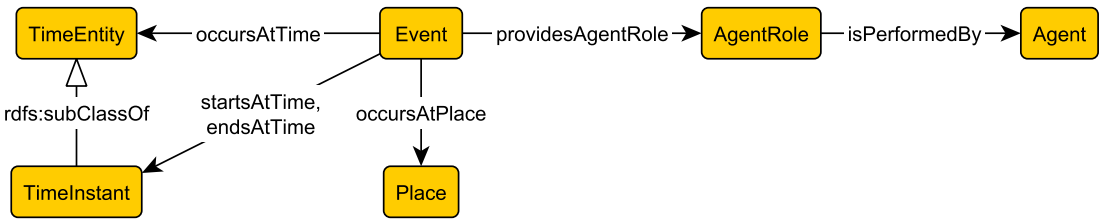


Figure A.6: Event pattern

This is a simple pattern describing events. The modeling approach is based on the Simple Event Model (SEM) [58]. Essentially, an event occurs at some place and some time. Also, an event may provide roles performed by agents. This relationship is modeled based on the Agent Role pattern (Appendix A.2) to which the Event pattern is aligned. When reusing this pattern, one may also restrict the set of all possible agent-role type an event may have, so that all agent-roles the event provides only come from that set, although this is not at all enforced in this pattern.

Information about place and time of an event is modeled in a generic way. For the former, we define the Place class as a hook to the Place pattern describing a generic place or point-of-interest. For the latter, we define the TimeInstant class as a hook to time:Instant from OWL Time ontology [65]. Time interval is accommodated through the startsAtTime and endsAtTime property. Note that these two are different from the properties with a similar name defined in the ⟨AgentRole⟩ pattern.

The Event pattern is (re-)used by Person pattern (Appendix A.4) to model birth event. The Cruise pattern (Appendix A.13) also uses this pattern.

A.3.2 Axiomatization

IRI Declarations

```
Prefix: : <http://schema.geolink.org/dev/event#>
Ontology: <http://schema.geolink.org/dev/event>
Import: <http://schema.geolink.org/dev/event-to-agent>
Import: <http://schema.geolink.org/dev/event-to-agentrole>
Import: <http://schema.geolink.org/dev/event-to-place>
Import: <http://schema.geolink.org/dev/event-to-owltime>
ObjectProperty: occursAtPlace, occursAtTime, startsAtTime,
                endsAtTime, providesAgentRole, isPerformedBy
Class: Event, Place, TimeInstant, AgentRole, Agent
```

Core Axioms

An event always occurs at some place and time. Time here is intended to be a generic time entity, which includes time points or intervals.

$$\mathsf{Event} \sqsubseteq \exists \mathsf{occursAtPlace}.\mathsf{Place} \sqcap \exists \mathsf{occursAtTime}.\mathsf{TimeEntity} \tag{A.22}$$

Starting and ending time are also a time at which the event occurs. We do not model the temporal ordering in this pattern. Since we want Event pattern to be generic, we also do not assert that an event can have at most one starting and ending time. In OWL 2 context, this means that startsAtTime and occursAtTime are not forced to be simple properties, hence allowing them to be implied by some property chain later on. We do, however, specify later that the range of startsAtTime and endsAtTime is TimeInstant.

$$\mathsf{startsAtTime} \sqsubseteq \mathsf{occursAtTime} \tag{A.23}$$

$$\mathsf{endsAtTime} \sqsubseteq \mathsf{occursAtTime} \tag{A.24}$$

$$\mathsf{TimeInstant} \sqsubseteq \mathsf{TimeEntity} \tag{A.25}$$

The Event pattern contains a specialization of the Agent Role pattern. As described in Fig. A.6, the Event pattern defines AgentRole and Agent class, as well

as providesAgentRole and isPerformedBy property. All of these entities in Event pattern are aligned to the corresponding classes and properties in the Agent Role pattern. Note that both AgentRole class and providesAgentRole property in the Event pattern are more *specific* than those defined by the Agent Role pattern, hence the subclass/subproperty relationship for the alignment. Furthermore, we do *not* assert that every event has to provide some agent-role. We do, however, assert that any agent-role has to be performed by exactly one agent, i.e.,

$$\mathsf{AgentRole} \sqsubseteq (=1\ \mathsf{isPerformedBy.Agent}) \tag{A.26}$$

In some specialization of Event, one can close the set of possible agent-role types an event may have – this is, however, not part of the axiomatization of Event pattern.

Next, we assert guarded domain and range restrictions for the properties defined in Event pattern.

$$\exists \mathsf{occursAtPlace.Place} \sqsubseteq \mathsf{Event} \tag{A.27}$$

$$\mathsf{Event} \sqsubseteq \forall \mathsf{occursAtPlace.Place} \tag{A.28}$$

$$\exists \mathsf{occursAtTime.TimeEntity} \sqsubseteq \mathsf{Event} \tag{A.29}$$

$$\mathsf{Event} \sqsubseteq \forall \mathsf{occursAtTime.TimeEntity} \tag{A.30}$$

$$\exists \mathsf{startsAtTime.TimeInstant} \sqsubseteq \mathsf{Event} \tag{A.31}$$

$$\mathsf{Event} \sqsubseteq \forall \mathsf{startsAtTime.TimeInstant} \tag{A.32}$$

$$\exists \mathsf{endsAtTime.TimeInstant} \sqsubseteq \mathsf{Event} \tag{A.33}$$

$$\mathsf{Event} \sqsubseteq \forall \mathsf{endsAtTime.TimeInstant} \tag{A.34}$$

$$\exists \mathsf{providesAgentRole.AgentRole} \sqsubseteq \mathsf{Event} \tag{A.35}$$

$$\mathsf{Event} \sqsubseteq \forall \mathsf{providesAgentRole.AgentRole} \tag{A.36}$$

$$\exists \mathsf{isPerformedBy.Agent} \sqsubseteq \mathsf{AgentRole} \tag{A.37}$$

$$\mathsf{AgentRole} \sqsubseteq \forall \mathsf{isPerformedBy.Agent} \tag{A.38}$$

Finally, we assert class disjointness axioms as follows.

$$\mathsf{alldisjoint(Event, Place, TimeEntity, AgentRole, Agent)} \tag{A.39}$$

A.3.3 Alignment

Alignment with Agent pattern

We align Event pattern with Agent pattern on the Agent class as depicted in Fig. A.7.

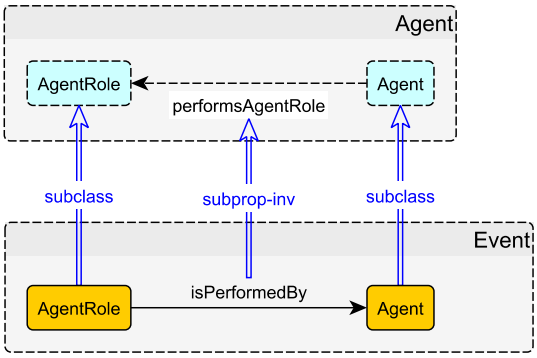


Figure A.7: Event aligned to Agent

```
Prefix: ecglev: <http://schema.geolink.org/dev/event#>
Prefix: ecglag: <http://schema.geolink.org/dev/agent#>
Ontology: <http://schema.geolink.org/dev/event-to-agent>
ObjectProperty: ecglev:isPerformedBy,  ecglag:performsAgentRole
Class: ecglev:Agent, ecglev:AgentRole, ecglag:Agent, ecgleg:AgentRole
```

$$\mathsf{ecglev{:}Agent} \sqsubseteq \mathsf{ecglag{:}Agent} \tag{A.40}$$

$$\mathsf{ecglev{:}AgentRole} \sqsubseteq \mathsf{ecglag{:}AgentRole} \tag{A.41}$$

$$\mathsf{ecglev{:}isPerformedBy} \sqsubseteq \mathsf{ecglag{:}performsAgentRole}^{-} \tag{A.42}$$

Alignment with Agent Role pattern

We align Event pattern with Agent Role pattern on AgentRole class, as well as providesAgentRole property, as depicted in Fig. A.8.

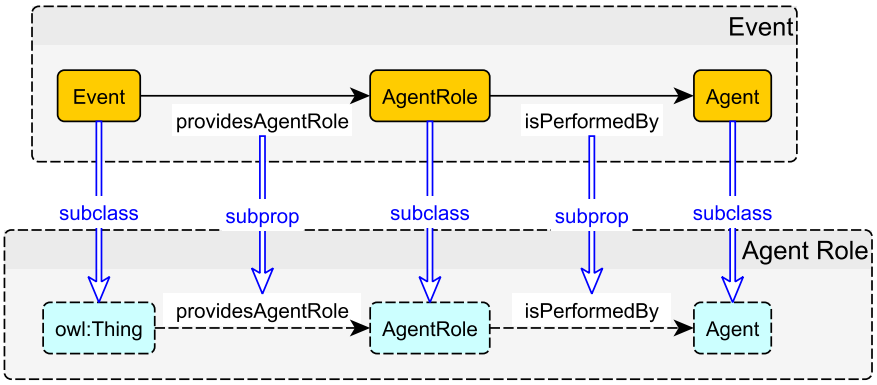


Figure A.8: Event aligned to Agent Role

```
Prefix: ecglar: <http://schema.geolink.org/dev/agentrole#>
Prefix: ecglev: <http://schema.geolink.org/dev/event#>
Ontology: <http://schema.geolink.org/dev/event-to-agentrole>
ObjectProperty: ecglev:providesAgentRole, ecglev:isPerformedBy,
                ecglar:providesAgentRole, ecglar:isPerformedBy
Class: ecglev:AgentRole,  ecglev:Agent, ecglar:AgentRole, ecglar:Agent
```

$$\text{ecglev:AgentRole} \sqsubseteq \text{ecglar:AgentRole} \tag{A.43}$$

$$\text{ecglev:Agent} \sqsubseteq \text{ecglar:Agent} \tag{A.44}$$

$$\text{ecglev:providesAgentRole} \sqsubseteq \text{ecglar:providesAgentRole} \tag{A.45}$$

$$\text{ecglev:isPerformedBy} \sqsubseteq \text{ecglar:isPerformedBy} \tag{A.46}$$

Alignment with Place pattern

We align Event pattern with Place pattern on the Place class as depicted in Fig. A.9.

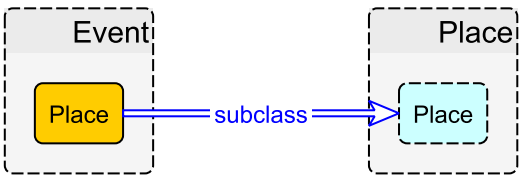


Figure A.9: Event aligned to Place

```
Prefix: ecglev: <http://schema.geolink.org/dev/event#>
Prefix: ecglpl: <http://schema.geolink.org/dev/place#>
Ontology: <http://schema.geolink.org/dev/event-to-place>
Class: ecglev:Place,  ecglpl:Place
```

$$\text{ecglev:Place} \sqsubseteq \text{ecglpl:Place} \tag{A.47}$$

Alignment with OWL Time ontology

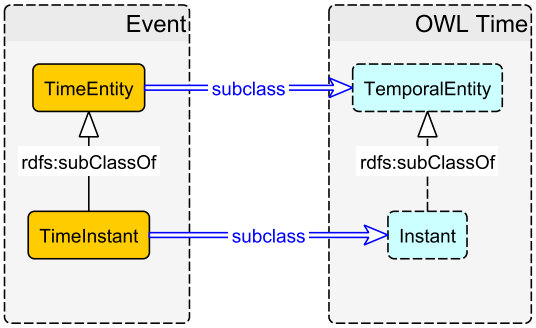


Figure A.10: Event aligned to OWL Time

We align Event pattern with OWL Time ontology [65] as depicted in Fig. A.10.

```
Prefix: time: <http://www.w3.org/2006/time#>
Prefix: ecglev: <http://schema.geolink.org/dev/event#>
Ontology: <http://schema.geolink.org/dev/event-to-owltime>
Class: ecglev:TimeInstant, ecglev:TimeEntity, time:Instant, time:TemporalEntity
```

$$\text{ecglev:TimeInstant} \sqsubseteq \text{time:Instant} \tag{A.48}$$

$$\text{ecglev:TimeEntity} \sqsubseteq \text{time:TemporalEntity} \tag{A.49}$$

A.4 Person

A.4.1 Description

Figure A.11: The Person pattern

The Person pattern has been described in Section 4.5.1. Essentially, we only say that a person may have a personal information item, and in addition, a person is a special kind of agent. The latter is modeled through alignment with the Agent pattern, which we describe separately in Section A.4.3. The rest are restating the axioms as well as alignment with the Personal Info Item pattern.

A.4.2 Axiomatization

IRI Declarations

```
Prefix: : <http://schema.geolink.org/dev/person#>
Ontology: <http://schema.geolink.org/dev/person>
Import: <http://schema.geolink.org/dev/person-to-agent>
Import: <http://schema.geolink.org/dev/person-to-personalinfoitem>
ObjectProperty: hasPersonalInfoItem
Class: Person, PersonalInfoItem
```

Core Axioms

Every person is an agent. This is, however, axiomatized in the alignment with the Agent pattern (Section A.4.3). A person may also have a personal information item. There is no axiomatization except the domain and range restrictions of the hasPersonalInfoItem property and class disjointness axiom.

$$\exists \mathsf{hasPersonalInfoItem}.\mathsf{PersonalInfoItem} \sqsubseteq \mathsf{Person} \tag{A.50}$$

$$\text{Person} \sqsubseteq \forall \text{hasPersonalInfoItem}.\text{PersonalInfoItem} \tag{A.51}$$

$$\text{alldisjoint}(\text{Person}, \text{PersonalInfoItem}) \tag{A.52}$$

A.4.3 Alignment

Alignment with Agent pattern

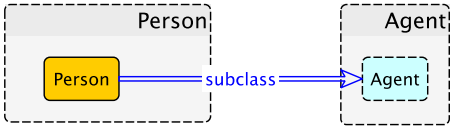


Figure A.12: Person aligned with Agent

The alignment is depicted in Fig. A.12

```
Prefix: ecglag: <http://schema.geolink.org/dev/agent#>
Prefix: ecglpr: <http://schema.geolink.org/dev/person#>
Ontology: <http://schema.geolink.org/dev/person-to-agent>
Class: ecglpr:Person, ecglag:Agent
```

$$\text{ecglpr:Person} \sqsubseteq \text{ecglag:Agent} \tag{A.53}$$

Alignment with Personal Info Item pattern

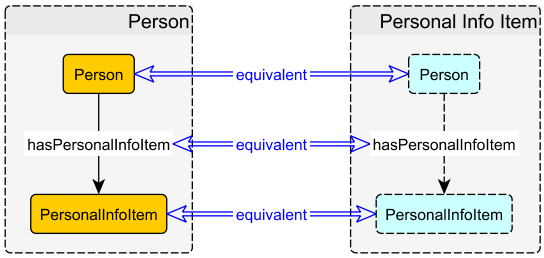


Figure A.13: Person aligned with Personal Info Item

The alignment is depicted in Fig. A.13.

```
Prefix: ecglpi: <http://schema.geolink.org/dev/personalinfoitem#>
Prefix: ecglpr: <http://schema.geolink.org/dev/person#>
Ontology: <http://schema.geolink.org/dev/person-to-personalinfoitem>
ObjectProperty: ecglpi:hasPersonalInfoItem, ecglpr:hasPersonalInfoItem
Class: ecglpi:Person, ecglpi:PersonalInfoItem, ecglpr:Person,
       ecglpr:PersonalInfoItem
```

$$\text{ecglpr:Person} \equiv \text{ecglpi:Person} \tag{A.54}$$

$$\text{ecglpr:PersonalInfoItem} \equiv \text{ecglpi:PersonalInfoItem} \tag{A.55}$$

$$\text{ecglpr:hasPersonalInfoItem} \equiv \text{ecglpi:hasPersonalInfoItem} \tag{A.56}$$

A.5 Personal Info Item

A.5.1 Description

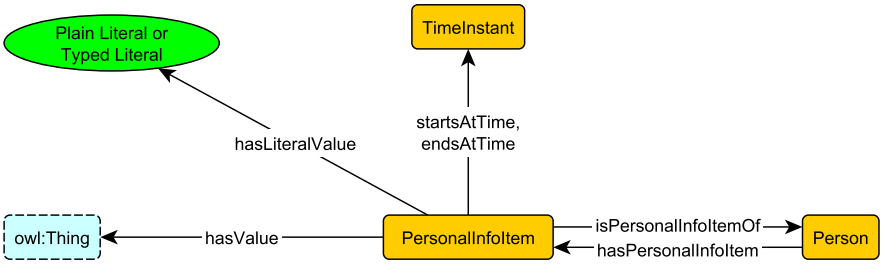


Figure A.14: The Personal Info Item pattern

The Personal Info Item pattern encapsulates any time-dependent attribute of a Person, for example, name, address, nationality, etc. A PersonalInfoItem starts at exactly one time:Instant and also ends at exactly one time:Instant. Currently, a correct temporal ordering is not enforced by this pattern. A PersonalInfoItem is a personal info item of exactly one Person and the alignment with the hasPersonalInfoItem property from the Person pattern is specified as the inverse of this relationship.

Each personal information item is typically associated with some value that represents the actual personal information. Such a personal information value depends on the type of personal information being presented. For example, a simple literal string may be enough for email address, while for a person's name, we may need three different literal strings to represent given name, surname, and the full name as it is customarily written. Such a value may also be represented with some controlled vocabulary terms, e.g., to model nationality, we may want to borrow a set of country names from an established standard. In this pattern stub, we provide one object property (hasValue) and one data property (hasLiteralValue) to associate the personal information item with its actual value. This is, however, done with an understanding that a specialization of this pattern will introduce its own properties that specifically

deal with the actual personal information value. Therefore, this pattern stub does not enforce that a personal information item has to have either a value or a literal value. In addition, a specialization of this pattern stub may also include some constraints on the range of personal information values allowed for it. We refer the reader to Appendix A.6 for a concrete specialization of this pattern stub.

A.5.2 Axiomatization

IRI Declarations

```
Ontology: <http://schema.geolink.org/dev/personalinfoitem>
Import: <http://schema.geolink.org/personalinfoitem-to-person>
Import: <http://schema.geolink.org/dev/personalinfoitem-to-owltime>
ObjectProperty: isPersonalInfoItemOf, hasPersonalInfoItem, startsAtTime,
                endsAtTime, hasValue
DataProperty: hasLiteralValue
Class: PersonalInfoItem, TimeInstant, Person
```

Core Axioms

Every PersonalInfoItem is a personal information item of exactly one person, starts exactly at one time point, and ends at no more than one time point.

$$\mathsf{PersonalInfoItem} \sqsubseteq (=1\ \mathsf{isPersonalInfoItemOf.Person}) \tag{A.57}$$

$$\mathsf{PersonalInfoItem} \sqsubseteq (=1\ \mathsf{startsAtTime.time{:}Instant}) \sqcap (\leqslant 1\ \mathsf{endsAtTime.time{:}Instant}) \tag{A.58}$$

$$\mathsf{hasPersonalInfoItem} \equiv \mathsf{isPersonalInfoItemOf}^{-} \tag{A.59}$$

We also want to say that every personal information item must be associated with some concrete value. However, this is not axiomatized here because the way such a value is provided really depend on the actual type of personal information item being modeled. For example, the way one represents a person's name can obviously be different than that of a person's address. In particular, we do *not* intend to restrict

such a value only as a single literal, hence the hasLiteralValue data property is not mandatory here. We now axiomatize the guarded domain and range restrictions for properties in this pattern.

$$\exists \mathsf{isPersonalInfoItemOf.Person} \sqsubseteq \mathsf{PersonalInfoItem} \tag{A.60}$$

$$\mathsf{PersonalInfoItem} \sqsubseteq \forall \mathsf{isPersonalInfoItemOf.Person} \tag{A.61}$$

$$\exists \mathsf{hasPersonalInfoItem.PersonalInfoItem} \sqsubseteq \mathsf{Person} \tag{A.62}$$

$$\mathsf{Person} \sqsubseteq \forall \mathsf{hasPersonalInfoItem.PersonalInfoItem} \tag{A.63}$$

$$\exists \mathsf{startsAtTime.time{:}Instant} \sqsubseteq \mathsf{PersonalInfoItem} \tag{A.64}$$

$$\mathsf{PersonalInfoItem} \sqsubseteq \forall \mathsf{startsAtTime.time{:}Instant} \tag{A.65}$$

$$\exists \mathsf{endsAtTime.time{:}Instant} \sqsubseteq \mathsf{PersonalInfoItem} \tag{A.66}$$

$$\mathsf{PersonalInfoItem} \sqsubseteq \forall \mathsf{endsAtTime.time{:}Instant} \tag{A.67}$$

$$\mathsf{dom(hasValue)} \sqsubseteq \mathsf{PersonalInfoItem} \tag{A.68}$$

$$\mathsf{dom(hasLiteralValue)} \sqsubseteq \mathsf{PersonalInfoItem} \tag{A.69}$$

Finally, we assert class disjointness axioms for classes in this pattern.

$$\mathsf{alldisjoint(PersonalInfoItem, Person, TimeInstant)} \tag{A.70}$$

A.5.3 Alignment

Alignment with Person pattern

```
Prefix: ecglpi: <http://schema.geolink.org/dev/personalinfoitem#>
Prefix: ecglpr: <http://schema.geolink.org/dev/person#>
Ontology: <http://schema.geolink.org/dev/personalinfoitem-to-person>
ObjectProperty: ecglpi:hasPersonalInfoItem, ecglpr:hasPersonalInfoItem
Class: ecglpi:Person, ecglpi:PersonalInfoItem,  ecglpr:Person,
       ecglpr:PersonalInfoItem
```

$$\mathsf{ecglpi{:}Person} \equiv \mathsf{ecglpr{:}Person} \tag{A.71}$$

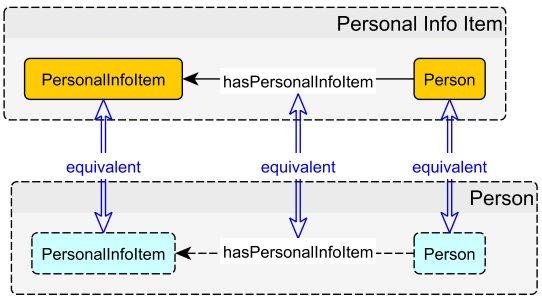


Figure A.15: Personal Info Item pattern aligned to Person pattern

$$\text{ecglpi:PersonalInfoItem} \equiv \text{ecglpr:PersonalInfoItem} \tag{A.72}$$

$$\text{ecglpi:hasPersonalInfoItem} \equiv \text{ecglpr:hasPersonalInfoItem} \tag{A.73}$$

Alignment with OWL Time ontology

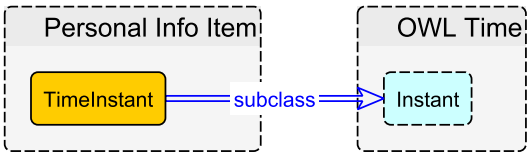


Figure A.16: Personal Info Item pattern aligned to OWL Time ontology

```
Prefix: time: <http://www.w3.org/2006/time#>
Prefix: ecglpi: <http://schema.geolink.org/dev/personalinfoitem#>
Ontology: <http://schema.geolink.org/dev/personalinfoitem-to-owltime>
Class: time:Instant, ecglpi:TimeInstant
```

$$\text{ecglpi:TimeInstant} \sqsubseteq \text{time:Instant} \tag{A.74}$$

A.6 Person Name

A.6.1 Description

Figure A.17: The Person Name pattern

The Person Name pattern is a specialization of the Personal Info Item pattern for describing a person's name. We are aware that names and their usages are very culturally dependent. Making a versatile pattern for person names which is applicable in a multitude of cultural, national, and legislative contexts is a formidable challenge in its own right, and we consider this out of scope for our current purposes. In this sense, our person name pattern is a stub pattern, i.e. it is not fully developed. The intended usage is that fullNameAsString points to the full official name of the person, transcribed to latin letters, while firstOrGivenName and familyOrSurname point to the corresponding two parts of the name as that person would usually give them. While this is not a satisfactory solution for many contexts, it will serve our purpose for now – and we understand that a more sophisticated solution, which is out of scope for us at this stage, would be preferable.

A.6.2 Axiomatization

IRI Declaration

```
Prefix: : <http://schema.geolink.org/dev/personname#>
Ontology: <http://schema.geolink.org/dev/personname>
Import: <http://schema.geolink.org/dev/personname-to-personalinfoitem>
Import: <http://schema.geolink.org/dev/personname-to-person>
ObjectProperty: hasPersonName
DataProperty: fullNameAsString, firstOrGivenName, familyOrSurname
Class: Person,  PersonName
```

Core Axioms

A person's name, represented by the PersonName class, is a type of personal information item. The subclass relationship between this pattern's PersonName and the Personal Info Item pattern's PersonalInfoItem is defined in the alignment pattern between the two patterns.

Every PersonName has exactly 1 full name string, at most one first/given name and at most one family/surname. Also, every person has to have a name (at some point of time). This pattern does not express that every person has to have a name at *any* point of time after his birth, since this may require a complicated form of temporal expression.

$$\mathsf{PersonName} \sqsubseteq (=1\ \mathsf{fullNameAsString.xsd{:}string}) \sqcap (\leqslant 1\ \mathsf{firstOrGivenName.xsd{:}string}) \sqcap (\leqslant 1\ \mathsf{familyOrSurname.xsd{:}string}) \tag{A.75}$$

$$\mathsf{Person} \sqsubseteq \exists \mathsf{hasPersonName.PersonName} \tag{A.76}$$

Next, we assert domain and range restrictions for fullNameAsString, firstOrGivenName, and familyOrSurname properties. Note that these are data properties whose range is rdf:PlainLiteral

$$\exists \mathsf{hasPersonName.PersonName} \sqsubseteq \mathsf{Person} \tag{A.77}$$

$$\mathsf{Person} \sqsubseteq \forall \mathsf{hasPersonName.PersonName} \tag{A.78}$$

$$\exists \mathsf{fullNameAsString.xsd{:}string} \sqsubseteq \mathsf{PersonName} \tag{A.79}$$

$$\mathsf{PersonName} \sqsubseteq \forall \mathsf{fullNameAsString.xsd{:}string} \tag{A.80}$$

$$\exists \mathsf{firstOrGivenName.xsd{:}string} \sqsubseteq \mathsf{PersonName} \tag{A.81}$$

$$\mathsf{PersonName} \sqsubseteq \forall \mathsf{firstOrGivenName.xsd{:}string} \tag{A.82}$$

$$\exists \mathsf{familyOrSurname.xsd{:}string} \sqsubseteq \mathsf{PersonName} \tag{A.83}$$

$$\mathsf{PersonName} \sqsubseteq \forall \mathsf{familyOrSurname.xsd{:}string} \tag{A.84}$$

Finally, the class disjointness is asserted.

$$\mathsf{PersonName} \sqcap \mathsf{Person} \sqsubseteq \bot \tag{A.85}$$

A.6.3 Alignment

Alignment with Personal Info Item pattern

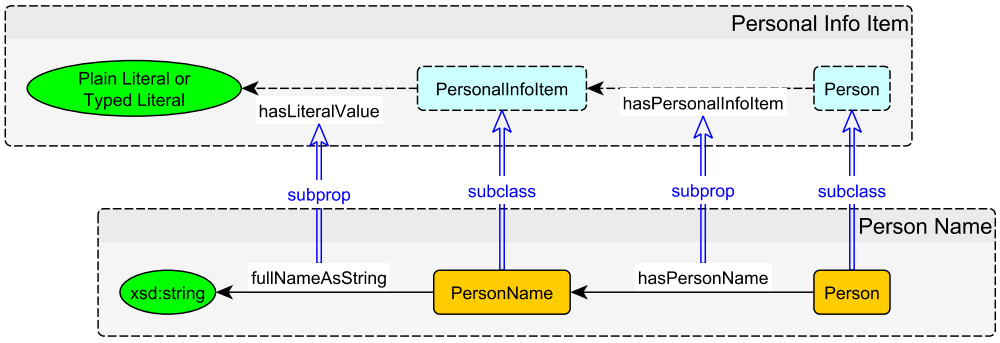


Figure A.18: Person Name pattern aligned to Personal Info Item pattern

```
Prefix: ecglpn: <http://schema.geolink.org/dev/personname#>
Prefix: ecglpi: <http://schema.geolink.org/dev/personalinfoitem#>
Ontology: <http://schema.geolink.org/dev/personname-to-personalinfoitem>
ObjectProperty: ecglpn:hasPersonName, ecglpi:hasPersonalInfoItem
DataProperty: ecglpn:fullNameAsString, ecglpi:hasLiteralValue
Class: ecglpn:Person, ecglpn:PersonName, ecglpi:Person,
       ecglpi:PersonalInfoItem
```

$$\mathsf{ecglpn{:}Person} \sqsubseteq \mathsf{ecglpi{:}Person} \tag{A.86}$$

$$\mathsf{ecglpn{:}PersonName} \sqsubseteq \mathsf{ecglpi{:}PersonalInfoItem} \tag{A.87}$$

$$\mathsf{ecglpn{:}hasPersonName} \sqsubseteq \mathsf{ecglpi{:}hasPersonalInfoItem} \tag{A.88}$$

$$\mathsf{ecglpn{:}fullNameAsString} \sqsubseteq \mathsf{ecglpi{:}hasLiteralValue} \tag{A.89}$$

Alignment with Person pattern

```
Prefix: ecglpn: <http://schema.geolink.org/dev/personname#>
```

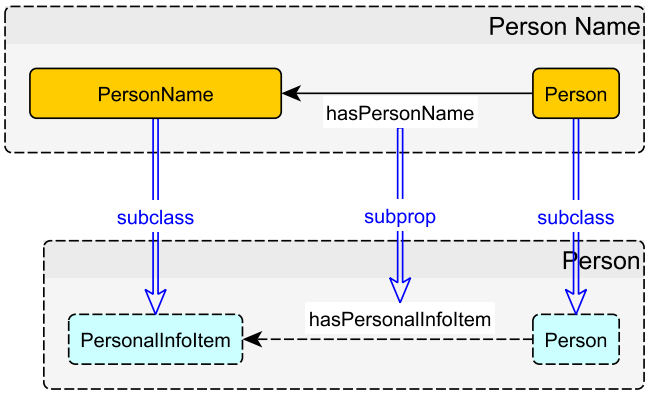


Figure A.19: Person Name pattern aligned to Personal Info Item pattern

```
Prefix: ecglpr: <http://schema.geolink.org/dev/person#>
Ontology: <http://schema.geolink.org/dev/personname-to-person>
ObjectProperty: ecglpn:hasPersonName, ecglpr:hasPersonalInfoItem
Class: ecglpn:Person, ecglpn:PersonName, ecglpr:Person,
       ecglpr:PersonalInfoItem
```

$$\text{ecglpn:Person} \sqsubseteq \text{ecglpr:Person} \tag{A.90}$$

$$\text{ecglpn:PersonName} \sqsubseteq \text{ecglpr:PersonalInfoItem} \tag{A.91}$$

$$\text{ecglpn:hasPersonName} \sqsubseteq \text{ecglpr:hasPersonalInfoItem} \tag{A.92}$$

A.7 Identifier

A.7.1 Description

Figure A.20: Identifier pattern

The Identifier pattern, as depicted in Fig. A.20, captures arbitrary identifiers that a data provider may use on their data. This is especially useful for non-URI identifiers such as local catalog numbers, etc. We expect identifier information to be used in conjunction with the Information Object pattern (Appendix A.8), although we do not axiomatize it here. We model an identifier to have exactly one string literal as its value, and at most one identifier scheme.

A.7.2 Axiomatization

IRI Declaration

```
Prefix: : <http://schema.geolink.org/dev/identifier#>
Ontology: <http://schema.geolink.org/dev/identifier>
ObjectProperty: hasIdentifierScheme
DataProperty: hasIdentifierValue
Class: Identifier, IdentifierSCheme
```

Core Axioms

$$\mathsf{Identifier} \sqsubseteq (=1\ \mathsf{hasIdentifierValue.xsd{:}string}) \tag{A.93}$$

$$\mathsf{Identifier} \sqsubseteq (\leqslant 1\ \mathsf{hasIdentifierScheme.IdentifierScheme}) \tag{A.94}$$

$$\exists \mathsf{hasIdentifierValue.xsd{:}string} \sqsubseteq \mathsf{Identifier} \tag{A.95}$$

$$\mathsf{Identifier} \sqsubseteq \forall \mathsf{hasIdentifierValue.xsd{:}string} \tag{A.96}$$

$$\exists \mathsf{hasIdentifierScheme.IdentifierScheme} \sqsubseteq \mathsf{Identifier} \tag{A.97}$$

$$\mathsf{Identifier} \sqsubseteq \forall \mathsf{hasIdentifierScheme}.\mathsf{IdentifierScheme} \tag{A.98}$$

A.8 Information Object

A.8.1 Description

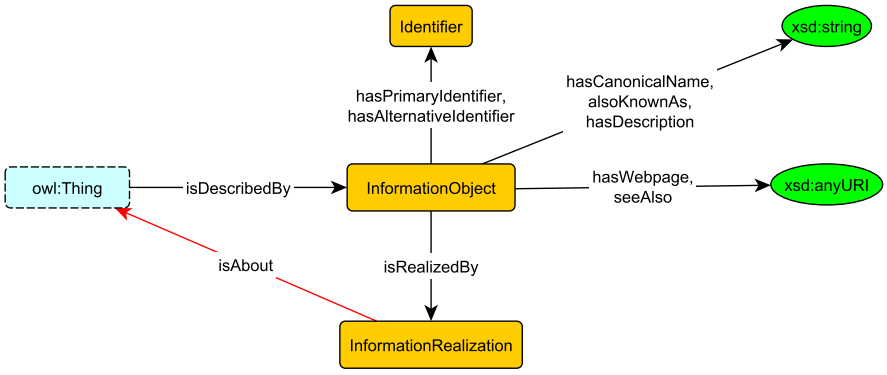


Figure A.21: Information Object pattern

The Information Object pattern in Fig. A.21 models the additional information assigned to an object. Essentially, anything can be described by an information object, and the information object in turn carries adornment, such as descriptions and webpages, about the target thing. The notion of information object is inspired by the information object component of the DOLCE ontology [102]. In the future, this pattern can be aligned to the W3C Prov-O.

Everything can have at most one information object associated with it. However, one information object can have one or more information realizations, such as csv files, word documents, and database tables. We consider the class DigitalObject in the Digital Object pattern as a subclass of InformationRealization. This will be asserted not in this pattern, but rather in the alignment from Digital Object pattern to this pattern later. Currently, this pattern stub provides a number of properties whose range is either strings or URIs. The property hasDescription provides a simple string description of any instance. The properties hasWebpage and seeAlso provide URL information that can be used to find further information. The property hasCanonicalName provides the canonical name of an instance as string. The

property alsoKnownAs provides a string that can be used as an alternative name aside from the one provided by hasCanonicalName. Identifier of an object (not the information object that describes it) can also be included via hasPrimaryIdentifier and hasAlternativeIdentifier property. The detail of identifier is modeled by a separate Identifier pattern in Appendix A.7.

A.8.2 Axiomatization

IRI Declaration

```
Prefix: : <http://schema.geolink.org/dev/informationobject#>
Ontology: <http://schema.geolink.org/dev/informationobject>
Import: <http://schema.geolink.org/dev/informationobject-to-identifier>
ObjectProperty: isDescribedBy, isRealizedBy, hasPrimaryIdentifier,
                hasAlternativeIdentifier
DataProperty: hasWebpage, alsoKnownAs, hasDescription, seeAlso,
              hasCanonicalName
Class: InformationObject, Identifier, InformationRealization
```

Core Axioms

For InformationObject, we only assert that everything, except information object, identifier, and information realization, is described by at most one InformationObject, while any InformationObject describes exactly one thing. The isAbout property is implied by the property chain connecting InformationRealization to the thing being described by the InformationObject.

$$\top \sqsubseteq (\leqslant 1\ \mathsf{isDescribedBy}.\mathsf{InformationObject}) \tag{A.99}$$

$$\mathsf{InformationObject} \sqsubseteq (=1\ \mathsf{isDescribedBy}^{-}.\top) \tag{A.100}$$

$$\mathsf{InformationObject} \sqsubseteq \neg\exists \mathsf{isDescribedBy}.\mathsf{InformationObject} \tag{A.101}$$

$$\mathsf{InformationRealization} \sqsubseteq \neg\exists \mathsf{isDescribedBy}.\mathsf{InformationObject} \tag{A.102}$$

$$\mathsf{Identifier} \sqsubseteq \neg\exists \mathsf{isDescribedBy}.\mathsf{InformationObject} \tag{A.103}$$

$$\mathsf{isRealizedBy}^{-} \circ \mathsf{isDescribedBy}^{-} \sqsubseteq \mathsf{isAbout} \tag{A.104}$$

Domain and range restrictions

$$\mathsf{range(isDescribedBy)} \sqsubseteq \mathsf{InformationObject} \tag{A.105}$$

$$\exists \mathsf{isRealizedBy.InformationRealization} \sqsubseteq \mathsf{InformationObject} \tag{A.106}$$

$$\mathsf{InformationObject} \sqsubseteq \forall \mathsf{isRealizedBy.InformationRealization} \tag{A.107}$$

$$\mathsf{dom(isAbout)} \sqsubseteq \mathsf{InformationRealization} \tag{A.108}$$

$$\exists \mathsf{hasCanonicalName.xsd:string} \sqsubseteq \mathsf{InformationObject} \tag{A.109}$$

$$\mathsf{InformationObject} \sqsubseteq \forall \mathsf{hasCanonicalName.xsd:string} \tag{A.110}$$

$$\exists \mathsf{alsoKnownAs.xsd:string} \sqsubseteq \mathsf{InformationObject} \tag{A.111}$$

$$\mathsf{InformationObject} \sqsubseteq \forall \mathsf{alsoKnownAs.xsd:string} \tag{A.112}$$

$$\exists \mathsf{hasDescription.xsd:string} \sqsubseteq \mathsf{InformationObject} \tag{A.113}$$

$$\mathsf{InformationObject} \sqsubseteq \forall \mathsf{hasDescription.xsd:string} \tag{A.114}$$

$$\exists \mathsf{hasWebpage.xsd:anyURI} \sqsubseteq \mathsf{InformationObject} \tag{A.115}$$

$$\mathsf{InformationObject} \sqsubseteq \forall \mathsf{hasWebpage.xsd:anyURI} \tag{A.116}$$

$$\exists \mathsf{seeAlso.xsd:anyURI} \sqsubseteq \mathsf{InformationObject} \tag{A.117}$$

$$\mathsf{InformationObject} \sqsubseteq \forall \mathsf{seeAlso.xsd:anyURI} \tag{A.118}$$

$$\exists \mathsf{hasAlternativeIdentifier.Identifier} \sqsubseteq \mathsf{InformationObject} \tag{A.119}$$

$$\mathsf{InformationObject} \sqsubseteq \forall \mathsf{hasAlternativeIdentifier.Identifier} \tag{A.120}$$

$$\exists \mathsf{hasPrimaryIdentifier.Identifier} \sqsubseteq \mathsf{InformationObject} \tag{A.121}$$

$$\mathsf{InformationObject} \sqsubseteq \forall \mathsf{hasPrimaryIdentifier.Identifier} \tag{A.122}$$

Disjointness axioms:

$$\mathsf{alldisjoint(InformationObject, InformationRealization, Identifier)} \tag{A.123}$$

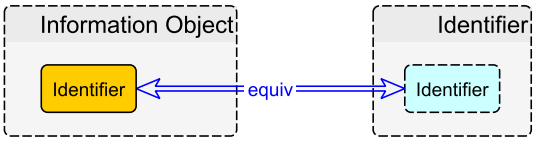


Figure A.22: Information Object pattern aligned with Identifier pattern

A.8.3 Alignment

Alignment with Identifier pattern

```
Prefix: ecglio: <http://schema.geolink.org/dev/informationobject#>
Prefix: ecglid: <http://schema.geolink.org/dev/identifier#>
Ontology: <http://schema.geolink.org/dev/informationobject-to-identifier>
Class: ecglio:Identifier, ecglid:Identifier
```

$$\text{ecglio:Identifier} \equiv \text{egclid:Identifier} \tag{A.124}$$

A.9 Organization

A.9.1 Description

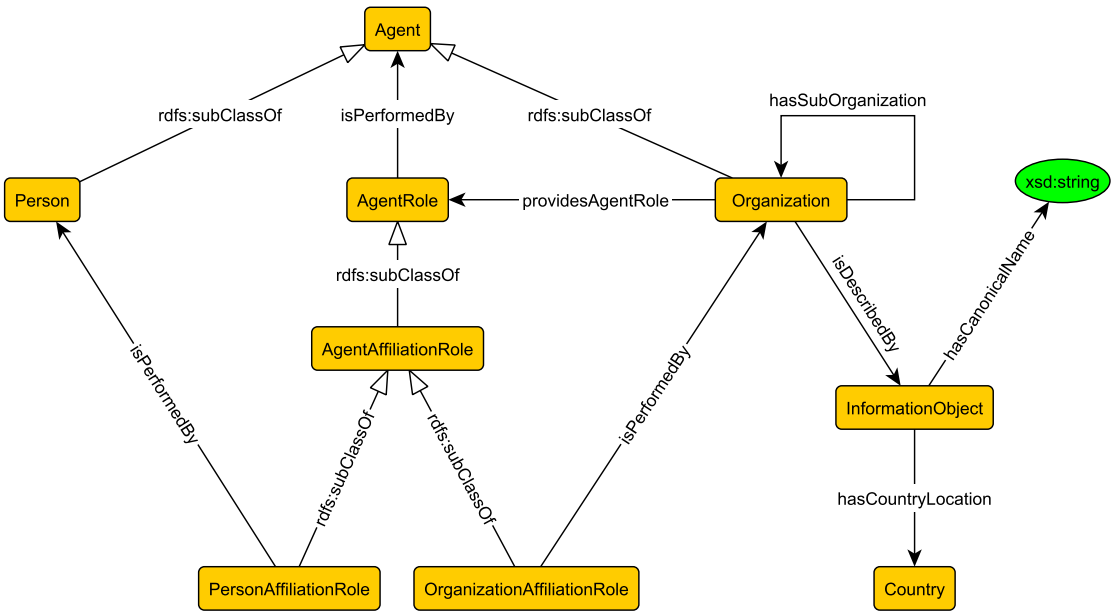


Figure A.23: The Organization pattern

This pattern describes organizations, as depicted in Fig. A.23. Organizations, as modeled here, have two important characteristics. First, organizations (represented by Organization as the main class) are agents. That is, they can perform some agent-roles in certain situations. Second, organizations may provide agent-role that can be performed by agents, including persons and other organizations (hence the use of providesAgentRole property).

A type of role that an organization can definitely provide is affiliation-role (via AgentAffiliationRole class). In particular, affiliation-role that is performed by a person is important in the context of GeoLink, hence the PersonAffiliationRole class. For the sake of symmetry, OrganizationAffiliationRole class is also provided. Note that PersonAffiliationRole is always performed by a Person, whereas a OrganizationAffiliationRole is always performed by an Organization.

The introduction of PersonAffiliationRole motivates a local definition of Person class. Now, since we have both Organization and Person locally defined, instead of aligning Organization directly to the Agent and Agent Role patterns, we introduce a local definition of Agent, declare Organization and Person as its subclass, and then align it with Agent in both Agent and Agent Role patterns. In addition, Person is aligned to the Person class in Person pattern.

Other information about an organization is modeled through the use of Information Object pattern. We introduce OrganizationInformationObject as its subclass (via alignment). This is to accommodate information items specific only for organizations. Currently, only country and canonical name information are included here (the latter is included since we want to assert that such a canonical name always exists for an organization). Finally, this pattern also allows a partonomic relationship between organization through the hasSubOrganization property.

A.9.2 Axiomatization

IRI Declarations

```
Prefix: : <http://schema.geolink.org/dev/organization#>
Prefix: xsd: <http://www.w3.org/2001/XMLSchema#>
Ontology: <http://schema.geolink.org/dev/organization>
Import: <http://schema.geolink.org/dev/organization-to-agent>
Import: <http://schema.geolink.org/dev/organization-to-person>
Import: <http://schema.geolink.org/dev/organization-to-informationobject>
Import: <http://schema.geolink.org/dev/organization-to-agentrole>
Datatype: xsd:string
ObjectProperty: hasSubOrganization, isDescribedBy, isPerformedBy,
                providesAgentRole, hasCountryLocation
DataProperty: hasCanonicalName
Class: Organization, Person, InformationObject, Country, AgentRole,
       Agent, AgentAffiliationRole, OrganizationAffiliationRole,
       PersonAffiliationRole
```

Core Axioms

Each organization is an agent and is described by exactly one instance of InformationObject. Further, every InformationObject describes exactly one organization.

$$\mathsf{Organization} \sqsubseteq \mathsf{Agent} \tag{A.125}$$

$$\mathsf{Person} \sqsubseteq \mathsf{Agent} \tag{A.126}$$

$$\mathsf{Organization} \sqsubseteq (=1\ \mathsf{isDescribedBy.InformationObject}) \tag{A.127}$$

$$\mathsf{InformationObject} \sqsubseteq (=1\ \mathsf{isDescribedBy}^{-}\mathsf{.Organization}) \tag{A.128}$$

The hasSubOrganization property is transitive.

$$\mathsf{hasSubOrganization} \circ \mathsf{hasSubOrganization} \sqsubseteq \mathsf{hasSubOrganization} \tag{A.129}$$

Information objects that describe organizations have exactly one canonical name.

$$\mathsf{InformationObject} \sqsubseteq (=\ 1\ \mathsf{hasCanonicalName.xsd:string}) \tag{A.130}$$

Hierarchy of agent-roles:

$$\mathsf{AgentAffiliationRole} \sqsubseteq \mathsf{AgentRole} \tag{A.131}$$

$$\mathsf{PersonAffiliationRole} \sqsubseteq \mathsf{AgentAffiliationRole} \tag{A.132}$$

$$\mathsf{PersonAffiliationRole} \sqsubseteq (=1\ \mathsf{isPerformedBy.Person}) \tag{A.133}$$

$$\mathsf{OrganizationAffiliationRole} \sqsubseteq \mathsf{AgentAffiliationRole} \tag{A.134}$$

$$\mathsf{OrganizationAffiliationRole} \sqsubseteq (=1\ \mathsf{isPerformedBy.Organization}) \tag{A.135}$$

Also, we assert the guarded domain and range restrictions. We specify domain and/or range restrictions for the performsAgentRole and isPerformedBy properties only w.r.t. the specifi type of roles we defined above.

$$\exists \mathsf{isPerformedBy.Agent} \sqsubseteq \mathsf{AgentRole} \tag{A.136}$$

$$\mathsf{AgentRole} \sqsubseteq \forall \mathsf{isPerformedBy.Agent} \tag{A.137}$$

$$\mathsf{PersonAffiliationRole} \sqsubseteq \forall \mathsf{isPerformedBy.Person} \tag{A.138}$$

$$\mathsf{OrganizationAffiliationRole} \sqsubseteq \forall \mathsf{isPerformedBy.Organization} \tag{A.139}$$

$$\exists \mathsf{providesAgentRole.AgentRole} \sqsubseteq \mathsf{Organization} \tag{A.140}$$

$$\mathsf{Organization} \sqsubseteq \forall \mathsf{providesAgentRole.AgentRole} \tag{A.141}$$

$$\exists \mathsf{hasSubOrganization.Organization} \sqsubseteq \mathsf{Organization} \tag{A.142}$$

$$\mathsf{Organization} \sqsubseteq \forall \mathsf{hasSubOrganization.Organization} \tag{A.143}$$

$$\exists \mathsf{isDescribedBy.InformationObject} \sqsubseteq \mathsf{Organization} \tag{A.144}$$

$$\mathsf{Organization} \sqsubseteq \forall \mathsf{isDescribedBy.InformationObject} \tag{A.145}$$

$$\exists \mathsf{hasCountryLocation.Country} \sqsubseteq \mathsf{InformationObject} \tag{A.146}$$

$$\mathsf{InformationObject} \sqsubseteq \forall \mathsf{hasCountryLocation.Country} \tag{A.147}$$

$$\exists \mathsf{hasCanonicalName.xsd:string} \sqsubseteq \mathsf{InformationObject} \tag{A.148}$$

$$\mathsf{InformationObject} \sqsubseteq \forall \mathsf{hasCanonicalName.xsd:string} \tag{A.149}$$

Finally, we assert the following disjointness axioms.

$$\mathsf{alldisjoint(Agent, AgentRole, OrganizationInformationObject,Country)} \tag{A.150}$$

$$\mathsf{Organization} \sqcap \mathsf{Person} \sqsubseteq \bot \tag{A.151}$$

$$\mathsf{PersonAffiliationRole} \sqcap \mathsf{OrganizationAffiliationRole} \sqsubseteq \bot \tag{A.152}$$

A.9.3 Alignment

Alignment with Agent pattern

```
Prefix: ecglor: <http://schema.geolink.org/dev/organization#>
Prefix: ecglag: <http://schema.geolink.org/dev/agent#>
Ontology: <http://schema.geolink.org/dev/organization-to-agent>
ObjectProperty: ecglor:isPerformedBy,  ecglag:performsAgentRole
Class: ecglor:Agent, ecglor:AgentRole, ecglag:Agent, ecglag:AgentRole
```

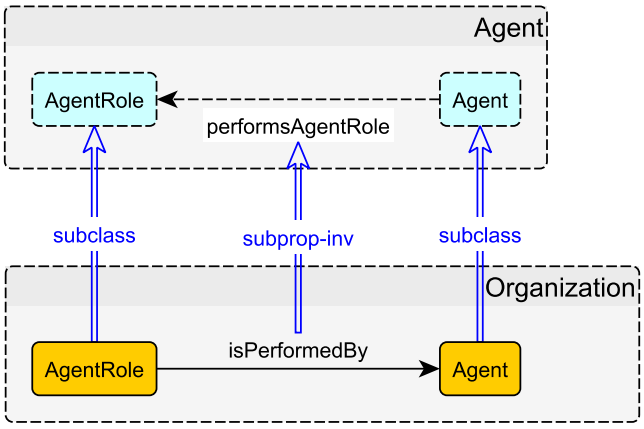


Figure A.24: The Organization pattern aligned to Agent pattern

$$\text{ecglor:Agent} \sqsubseteq \text{ecglag:Agent} \tag{A.153}$$

$$\text{ecglor:AgentRole} \sqsubseteq \text{ecglag:AgentRole} \tag{A.154}$$

$$\text{ecglor:isPerformedBy} \sqsubseteq \text{ecglag:performsAgentRole}^{-} \tag{A.155}$$

Alignment with Agent Role pattern

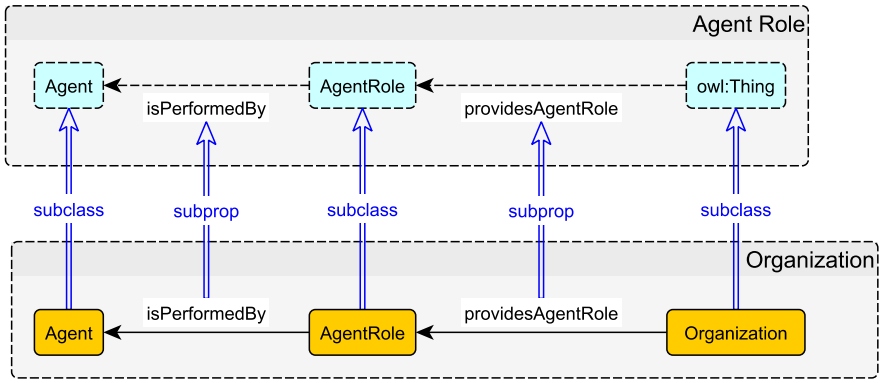


Figure A.25: The Organization pattern aligned to Agent Role pattern

Fig. A.25 depicts the alignment. Note that Organization is by default a subclass of owl:Thing.

```
Prefix: ecglor: <http://schema.geolink.org/dev/organization#>
```

```
Prefix: ecglar: <http://schema.geolink.org/dev/agentrole#>
Ontology: <http://schema.geolink.org/dev/organization-to-agentrole>
ObjectProperty: ecglor:isPerformedBy, ecglor:providesAgentRole
                ecglar:isPerformedBy, ecglar:providesAgentRole
Class: ecglor:Agent, ecglor:AgentRole,  ecglar:Agent, ecglar:AgentRole
```

$$\text{ecglor:Agent} \sqsubseteq \text{ecglar:Agent} \tag{A.156}$$

$$\text{ecglor:AgentRole} \sqsubseteq \text{ecglar:AgentRole} \tag{A.157}$$

$$\text{ecglor:isPerformedBy} \sqsubseteq \text{ecglar:isPerformedBy} \tag{A.158}$$

$$\text{ecglor:providesAgentRole} \sqsubseteq \text{ecglar:providesAgentRole} \tag{A.159}$$

Alignment with Person pattern

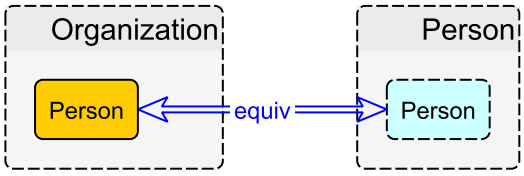


Figure A.26: The Organization pattern aligned to Person pattern

```
Prefix: ecglor: <http://schema.geolink.org/dev/organization#>
Prefix: ecglpr: <http://schema.geolink.org/dev/person#>
Ontology: <http://schema.geolink.org/dev/organization-to-person>
Class: ecglor:Person,  ecglpr:Person
```

$$\text{ecglor:Person} \equiv \text{ecglpr:Person} \tag{A.160}$$

Alignment with Information Object pattern

Fig. A.27 depicts the alignment. Organization is by definition a subclass of owl:Thing, hence an axiom is not needed.

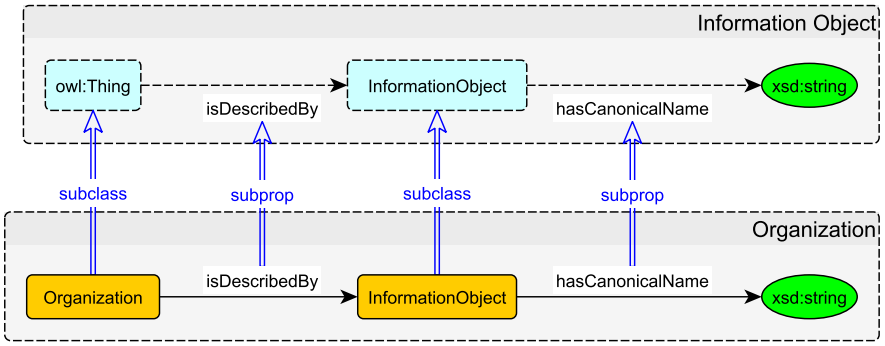


Figure A.27: The Organization pattern aligned to Information Object pattern

```
Prefix: ecglor: <http://schema.geolink.org/dev/organization#>
Prefix: ecglio: <http://schema.geolink.org/dev/informationobject#>
Ontology: <http://schema.geolink.org/dev/organization-to-informationobject>
ObjectProperty: ecglor:isDescribedBy,  ecglio:isDescribedBy
DataProperty: ecglor:hasCanonicalName, ecglio:hasCanonicalName
Class: ecglor:InformationObject, ecglio:InformationObject
```

$$\text{ecglor:InformationObject} \sqsubseteq \text{ecglio:InformationObject} \tag{A.161}$$

$$\text{ecglor:isDescribedBy} \sqsubseteq \text{ecglio:isDescribedBy} \tag{A.162}$$

$$\text{ecglor:hasCanonicalName} \sqsubseteq \text{ecglio:hasCanonicalName} \tag{A.163}$$

A.10 Funding Award

A.10.1 Description

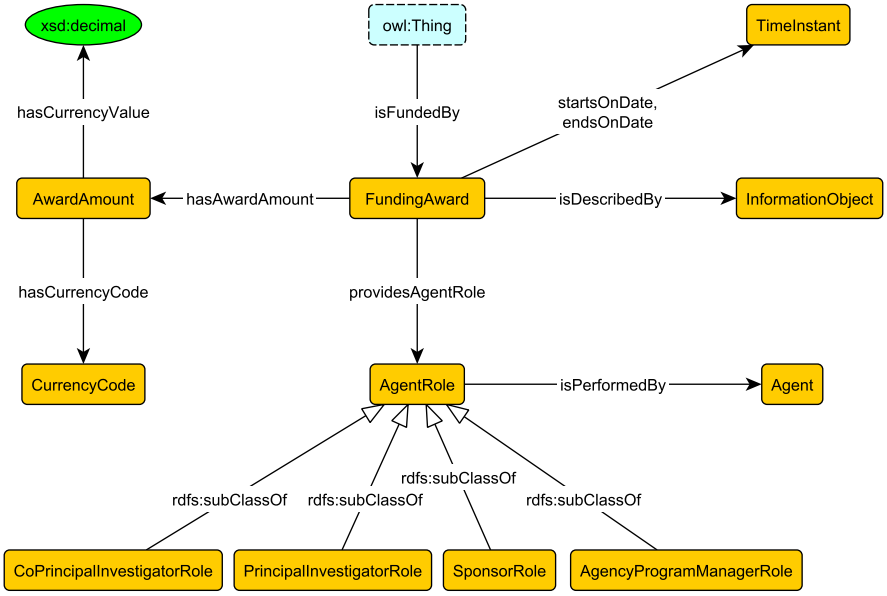


Figure A.28: The Funding Award pattern

The Funding Award pattern describes the funding awards that fund all kinds of ocean science research activities. We use the isFundedBy property to connect anything to a funding award if the funding award funds it. Each funding award has exactly one starting and ending date (aligned with time:Instant). It provides at most one award amount, which is described via a pair of decimal value and currency code. The currency code is not specified here, but existing standards can be used, e.g., ISO 4217. There may be people or organizations that have a role in a funding award. This is modeled by re-using (and aligning with) the Agent Role pattern.

In this version, we include the following types of agent-roles, represented as classes: SponsorRole, AgencyProgramManagerRole, PrincipalInvestigatorRole, and CoPrincipalInvestigatorRole. Additional roles are possible in the future versions.

Each funding award is described by a InformationObject, which when aligned to the Information Object pattern, allows one to represent additional information such as identifier, description, etc.

A.10.2 Axiomatization

IRI Declarations

```
Prefix: : <http://schema.geolink.org/dev/fundingaward#>
Prefix: xsd: <http://www.w3.org/2001/XMLSchema#>
Ontology: <http://schema.geolink.org/dev/fundingaward>
Import: <http://schema.geolink.org/dev/fundingaward-to-informationobject>
Import: <http://schema.geolink.org/dev/fundingaward-to-agentrole>
Import: <http://schema.geolink.org/dev/fundingaward-to-owltime>
Datatype: xsd:decimal
ObjectProperty: isFundedBy, startsOnDate, endsOnDate, isDescribedBy,
                providesAgentRole, isPerformedBy, hasAwardAmount,
                hasCurrencyCode
DataProperty: hasCurrencyValue
Class: FundingAward, Agent, TimeInstant, AgentRole, SponsorRole,
       PrincipalInvestigatorRole, CoPrincipalInvestigatorRole,
       AgencyProgramManagerRole, InformationObject, AwardAmount,
       CurrencyCode
```

Core Axioms

We assert that each funding award is described by exactly one InformationObject. Note that InformationObject can accommodate identifier information when aligned to Information Object pattern. Also, note that InformationObject in this pattern is really intended only for FundingAward.

$$\mathsf{FundingAward} \sqsubseteq (=1\ \mathsf{isDescribedBy}.\mathsf{InformationObject}) \tag{A.164}$$

Each funding award has exactly one starting time, exactly one ending time, at least one funding sponsor, and at most one award amount. The award amount corresponds

to exactly one currency value and one currency code.

$$\begin{aligned} \mathsf{FundingAward} &\sqsubseteq (=1\ \mathsf{startsOnDate.TimeInstant}) \\ &\quad \sqcap (=1\ \mathsf{endsOnDate.TimeInstant}) && \text{(A.165)} \\ \mathsf{FundingAward} &\sqsubseteq \exists \mathsf{providesAgentRole.SponsorRole} && \text{(A.166)} \\ \mathsf{SponsorRole} &\sqsubseteq \mathsf{AgentRole} && \text{(A.167)} \\ \mathsf{PrincipalInvestigatorRole} &\sqsubseteq \mathsf{AgentRole} && \text{(A.168)} \\ \mathsf{CoPrincipalInvestigatorRole} &\sqsubseteq \mathsf{AgentRole} && \text{(A.169)} \\ \mathsf{AgencyProgramManagerRole} &\sqsubseteq \mathsf{AgentRole} && \text{(A.170)} \\ \mathsf{FundingAward} &\sqsubseteq (\leqslant 1\ \mathsf{hasAwardAmount.AwardAmount}) && \text{(A.171)} \\ \mathsf{AwardAmount} &\sqsubseteq (=1\ \mathsf{hasCurrencyValue.xsd{:}decimal}) && \text{(A.172)} \\ \mathsf{AwardAmount} &\sqsubseteq (=1\ \mathsf{hasCurrencyCode.CurrencyCode}) && \text{(A.173)} \end{aligned}$$

We assert domain and range restrictions for properties defined in this pattern. Here, isDescribedBy is given a guarded domain and range restrictions that are specific for Funding Award pattern. Also, isFundedBy property has only an unguarded range restriction and no domain restriction.

$$\begin{aligned} \mathsf{range(isFundedBy)} &\sqsubseteq \mathsf{FundingAward} && \text{(A.174)} \\ \exists \mathsf{startsOnDate.time{:}Instant} &\sqsubseteq \mathsf{FundingAward} && \text{(A.175)} \\ \mathsf{FundingAward} &\sqsubseteq \forall \mathsf{startsOnDate.time{:}Instant} && \text{(A.176)} \\ \exists \mathsf{endsOnDate.time{:}Instant} &\sqsubseteq \mathsf{FundingAward} && \text{(A.177)} \\ \mathsf{FundingAward} &\sqsubseteq \forall \mathsf{endsOnDate.time{:}Instant} && \text{(A.178)} \\ \exists \mathsf{providesAgentRole.AgentRole} &\sqsubseteq \mathsf{FundingAward} && \text{(A.179)} \\ \mathsf{FundingAward} &\sqsubseteq \forall \mathsf{providesAgentRole.AgentRole} && \text{(A.180)} \\ \exists \mathsf{isPerformedBy.Agent} &\sqsubseteq \mathsf{AgentRole} && \text{(A.181)} \end{aligned}$$

$$\text{AgentRole} \sqsubseteq \forall \text{isPerformedBy.Agent} \tag{A.182}$$

$$\exists \text{isDescribedBy.InformationObject} \sqsubseteq \text{FundingAward} \tag{A.183}$$

$$\text{FundingAward} \sqsubseteq \forall \text{isDescribedBy.InformationObject} \tag{A.184}$$

$$\exists \text{hasAwardAmount.AwardAmount} \sqsubseteq \text{FundingAward} \tag{A.185}$$

$$\text{FundingAward} \sqsubseteq \forall \text{hasAwardAmount.AwardAmount} \tag{A.186}$$

$$\exists \text{hasCurrencyValue.xsd:decimal} \sqsubseteq \text{AwardAmount} \tag{A.187}$$

$$\text{AwardAmount} \sqsubseteq \forall \text{hasCurrencyValue.xsd:decimal} \tag{A.188}$$

$$\exists \text{hasCurrencyCode.CurrencyCode} \sqsubseteq \text{AwardAmount} \tag{A.189}$$

$$\text{AwardAmount} \sqsubseteq \forall \text{hasCurrencyCode.CurrencyCode} \tag{A.190}$$

Disjointness axioms:

$$\text{alldisjoint(FundingAward, InformationObject, AwardAmount, CurrencyCode,}$$
$$\text{AgentRole, Agent, TimeInstant)} \tag{A.191}$$

A.10.3 Alignment

Alignment with Agent pattern

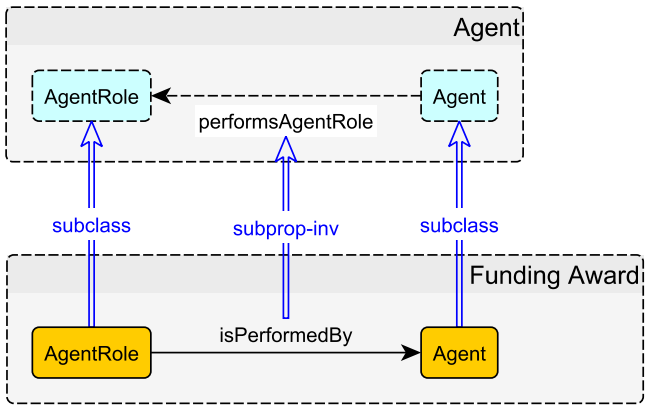


Figure A.29: The Funding Award pattern aligned to Agent pattern

```
Prefix: ecglfa: <http://schema.geolink.org/dev/fundingaward#>
Prefix: ecglag: <http://schema.geolink.org/dev/agent#>
Ontology: <http://schema.geolink.org/dev/fundingaward-to-agent>
ObjectProperty: ecglfa:isPerformedBy, ecglag:performsAgentRole
Class: ecglfa:Agent, ecglfa:AgentRole, ecglag:Agent, ecglag:AgentRole
```

$$\text{ecglfa:Agent} \sqsubseteq \text{ecglag:Agent} \tag{A.192}$$

$$\text{ecglfa:AgentRole} \sqsubseteq \text{ecglag:AgentRole} \tag{A.193}$$

$$\text{ecglfa:isPerformedBy} \sqsubseteq \text{ecglag:performsAgentRole}^{-} \tag{A.194}$$

Alignment with Agent Role pattern

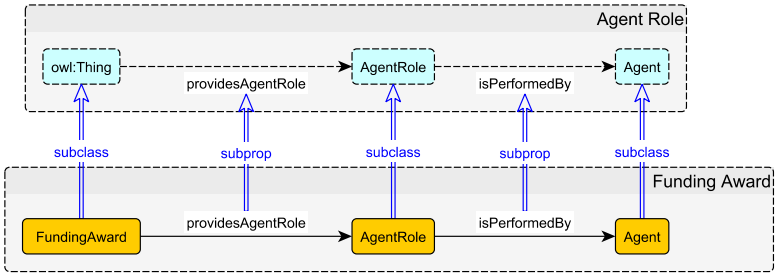


Figure A.30: The Funding Award pattern aligned with Agent Role pattern

```
Prefix: ecglfa: <http://schema.geolink.org/dev/fundingaward#>
Prefix: ecglar: <http://schema.geolink.org/dev/agentrole#>
Ontology: <http://schema.geolink.org/dev/fundingaward-to-agentrole>
ObjectProperty: ecglfa:isPerformedBy, ecglfa:providesAgentRole,
                ecglar:isPerformedBy, ecglar:providesAgentRole
Class: ecglfa:Agent, ecglfa:AgentRole, ecglar:Agent,
       ecglar:AgentRole
```

$$\text{ecglfa:Agent} \sqsubseteq \text{ecglar:Agent} \tag{A.195}$$

$$\text{ecglfa:AgentRole} \sqsubseteq \text{ecglar:AgentRole} \tag{A.196}$$

$$\text{ecglfa:isPerformedBy} \sqsubseteq \text{ecglar:isPerformedBy} \tag{A.197}$$

$$\text{ecglfa:providesAgentRole} \sqsubseteq \text{ecglar:providesAgentRole} \tag{A.198}$$

Alignment with Information Object pattern

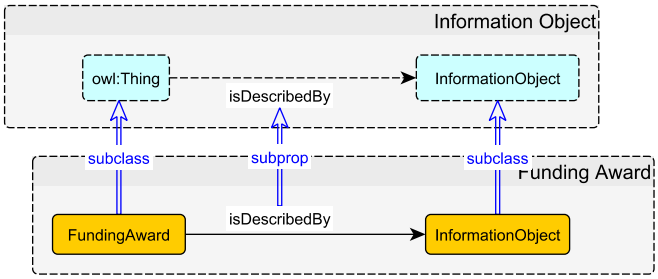


Figure A.31: The Funding Award pattern aligned with Information Object pattern

```
Prefix: ecglfa: <http://schema.geolink.org/dev/fundingaward#>
Prefix: ecglio: <http://schema.geolink.org/dev/informationobject#>
Ontology: <http://schema.geolink.org/dev/fundingaward-to-informationobject>
ObjectProperty: ecglfa:isDescribedBy, ecglio:isDescribedBy
Class: ecglfa:InformationObject, ecglio:InformationObject
```

$$\text{ecglfa:InformationObject} \sqsubseteq \text{ecglio:InformationObject} \tag{A.199}$$

$$\text{ecglfa:isDescribedBy} \sqsubseteq \text{ecglio:isDescribedBy} \tag{A.200}$$

Alignment with OWL Time

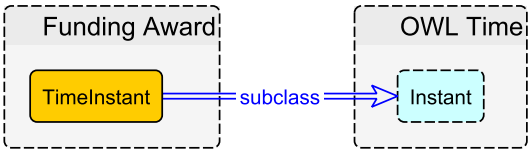


Figure A.32: The Funding Award pattern aligned with OWL Time

```
Prefix: ecglfa: <http://schema.geolink.org/dev/fundingaward#>
Prefix: time: <http://www.w3.org/2006/time#>
Ontology: <http://schema.geolink.org/dev/fundingaward-to-informationobject>
Class: ecglfa:TimeInstant, time:Instant
```

$$\text{ecglfa:TimeInstant} \sqsubseteq \text{time:Instant} \tag{A.201}$$

A.11 Program

A.11.1 Description

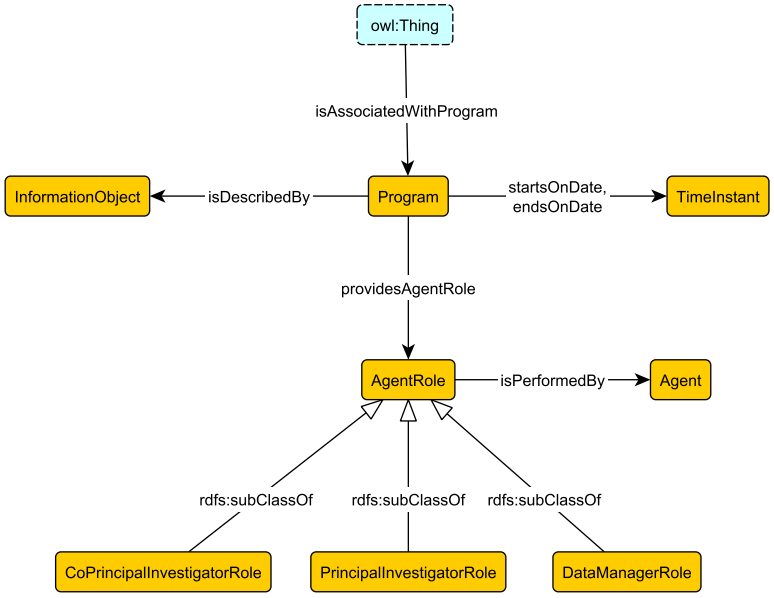


Figure A.33: The Program pattern

A program in the context of geoscience research community is a loose collection of things, including cruises, funding awards, activities, events, which are loosely grouped together. We model program in the Program pattern stub in Fig. A.33. First, we introduce isAssociatedWithProgram property, which can be used to connect anything to an instance of Program. For instance, if one wants to connect funding awards or digital object records to programs, (s)he could use this property. Next, any involvement of people or organization in the program can be modeled through the re-use of (and alignment with) Agent Role pattern. In this version of the pattern, we explicitly include data manager role, principal investigator role, and co-principal investigator role. Other kinds of agent-roles are possible in the future. Finally, other information regarding a program, e.g., name, webpage, description, etc., can be accommodated through the use of InformationObject pattern.

A.11.2 Axiomatization

IRI Declarations

```
Prefix: : <http://schema.geolink.org/dev/program#>
Prefix: xsd: <http://www.w3.org/2001/XMLSchema#>
Ontology: <http://schema.geolink.org/dev/program>
Import: <http://schema.geolink.org/dev/program-to-informationobject>
Import: <http://schema.geolink.org/dev/program-to-agentrole>
Import: <http://schema.geolink.org/dev/program-to-owltime>
ObjectProperty: isAssociatedWithProgram, startsOnDate, endsOnDate, isDescribedBy,
                providesAgentRole, isPerformedBy
Class: Program, Agent, TimeInstant, AgentRole, InformationObject,
       PrincipalInvestigatorRole, CoPrincipalInvestigatorRole,
       ProgramManagerRole
```

Core Axioms

Every program is described by exactly one information object. It also has at most one starting date and at most one ending date.

$$\mathsf{Program} \sqsubseteq (=1\ \mathsf{isDescribedBy}.\mathsf{InformationObject}) \tag{A.202}$$

$$\mathsf{InformationObject} \sqsubseteq (=1\ \mathsf{isDescribedBy}^{-}.\mathsf{Program}) \tag{A.203}$$

$$\mathsf{Program} \sqsubseteq (\leqslant 1\ \mathsf{startsOnDate}.\mathsf{TimeInstant}) \sqcap (\leqslant 1\ \mathsf{endsOnDate}.\mathsf{TimeInstant}) \tag{A.204}$$

PrincipalInvestigatorRole, CoPrincipalInvestigatorRole, and DataManagerRole are types of AgentRole.

$$\mathsf{PrincipalInvestigatorRole} \sqsubseteq \mathsf{AgentRole} \tag{A.205}$$

$$\mathsf{CoPrincipalInvestigatorRole} \sqsubseteq \mathsf{AgentRole} \tag{A.206}$$

$$\mathsf{DataManagerRole} \sqsubseteq \mathsf{AgentRole} \tag{A.207}$$

We assert guarded domain and range restrictions below.

$$\mathsf{range}(\mathsf{isAssociatedWithProgram}) \sqsubseteq \mathsf{Program} \tag{A.208}$$

$$\exists \text{isDescribedBy.InformationObject} \sqsubseteq \text{Program} \tag{A.209}$$

$$\text{Program} \sqsubseteq \forall \text{isDescribedBy.InformationObject} \tag{A.210}$$

$$\exists \text{startsOnDate.TimeInstant} \sqsubseteq \text{Program} \tag{A.211}$$

$$\text{Program} \sqsubseteq \forall \text{startsOnDate.TimeInstant} \tag{A.212}$$

$$\exists \text{endsOnDate.TimeInstant} \sqsubseteq \text{Program} \tag{A.213}$$

$$\text{Program} \sqsubseteq \forall \text{endsOnDate.TimeInstant} \tag{A.214}$$

$$\exists \text{providesAgentRole.AgentRole} \sqsubseteq \text{Program} \tag{A.215}$$

$$\text{Program} \sqsubseteq \forall \text{providesAgentRole.AgentRole} \tag{A.216}$$

$$\exists \text{isPerformedBy.Agent} \sqsubseteq \text{AgentRole} \tag{A.217}$$

$$\text{AgentRole} \sqsubseteq \forall \text{isPerformedBy.Agent} \tag{A.218}$$

Disjointness axioms

$$\text{alldisjoint(Program, InformationObject, TimeInstant, AgentRole, Agent)} \tag{A.219}$$

$$\text{alldisjoint(PrincipalInvestigatorRole, CoPrincipalInvestigatorRole, DataManagerRole)} \tag{A.220}$$

A.11.3 Alignment

Alignment with Agent pattern

```
Prefix: ecglpg: <http://schema.geolink.org/dev/program#>
Prefix: ecglag: <http://schema.geolink.org/dev/agent#>
Ontology: <http://schema.geolink.org/dev/program-to-agent>
ObjectProperty: ecglpg:isPerformedBy, ecglag:performsAgentRole
Class: ecglpg:Agent, ecglpg:AgentRole, ecglag:Agent, ecglag:AgentRole
```

$$\text{ecglpg:Agent} \sqsubseteq \text{ecglag:Agent} \tag{A.221}$$

$$\text{ecglpg:AgentRole} \sqsubseteq \text{ecglag:AgentRole} \tag{A.222}$$

$$\text{ecglpg:isPerformedBy} \sqsubseteq \text{ecglag:performsAgentRole}^{-} \tag{A.223}$$

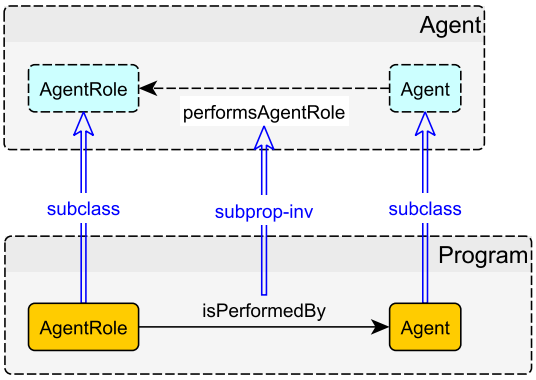


Figure A.34: The Program pattern aligned to Agent pattern

Alignment with Agent Role pattern

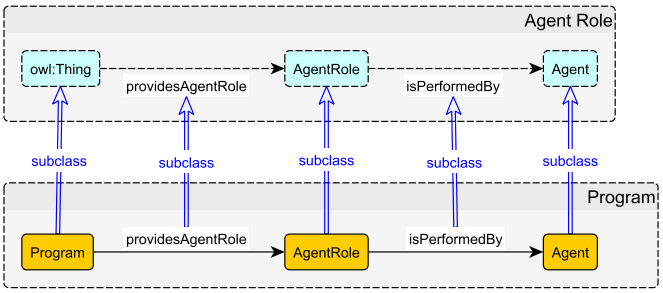


Figure A.35: The Program pattern aligned to Agent Role pattern

```
Prefix: ecglpg: <http://schema.geolink.org/dev/program#>
Prefix: ecglar: <http://schema.geolink.org/dev/agentrole#>
Ontology: <http://schema.geolink.org/dev/program-to-agentrole>
ObjectProperty: ecglpg:isPerformedBy, ecglpg:providesAgentRole,
                ecglar:isPerformedBy,  ecglar:providesAgentRole
Class: ecglpg:Agent, ecglpg:AgentRole, ecglar:Agent, ecglar:AgentRole
```

$$\text{ecglpg:Agent} \sqsubseteq \text{ecglar:Agent} \tag{A.224}$$

$$\text{ecglpg:AgentRole} \sqsubseteq \text{ecglar:AgentRole} \tag{A.225}$$

$$\text{ecglpg:isPerformedBy} \sqsubseteq \text{ecglar:isPerformedBy} \tag{A.226}$$

$$\text{ecglpg:providesAgentRole} \sqsubseteq \text{ecglar:providesAgentRole} \tag{A.227}$$

Alignment with Information Object pattern

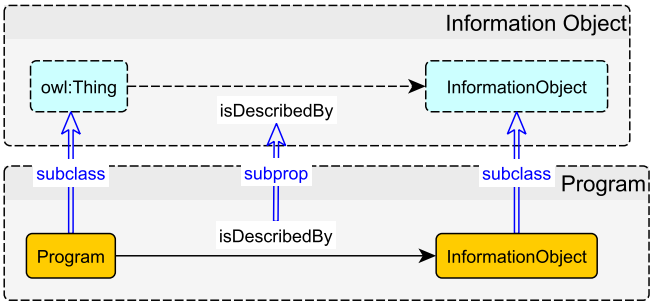


Figure A.36: The Program pattern aligned to Information Object pattern

```
Prefix: ecglpg: <http://schema.geolink.org/dev/program#>
Prefix: ecglio: <http://schema.geolink.org/dev/informationobject#>
Ontology: <http://schema.geolink.org/dev/program-to-informationobject>
ObjectProperty: ecglpg:isDescribedBy, ecglio:isDescribedBy
Class: ecglpg:InformationObject, ecglio:InformationObject
```

$$\text{ecglpg:InformationObject} \sqsubseteq \text{ecglio:InformationObject} \tag{A.228}$$

$$\text{ecglpg:isDescribedBy} \sqsubseteq \text{ecglio:isDescribedBy} \tag{A.229}$$

Alignment with OWL Time

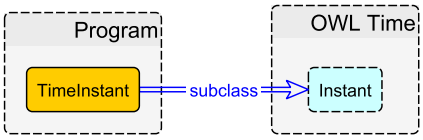


Figure A.37: The Program pattern aligned to OWL Time

```
Prefix: ecglfa: <http://schema.geolink.org/dev/program#>
Prefix: time: <http://www.w3.org/2006/time#>
Ontology: <http://schema.geolink.org/dev/program-to-owltime>
Class: ecglpg:TimeInstant, time:Instant
```

$$\text{ecglpg:TimeInstant} \sqsubseteq \text{time:Instant} \tag{A.230}$$

A.12 Place

A.12.1 Description

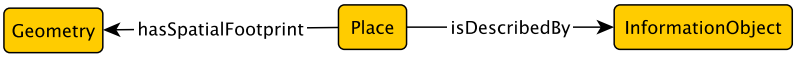


Figure A.38: The Place pattern stub

The Place pattern is intended to describe place of interests (POIs) that is associated with certain geospatial features. At this stage, however, we have not yet developed a more precise specification of this pattern, and plan to do so as a future work. In this version, we model a place simply as something that is described by some information object, which allows us to attach non-geospatial information (e.g., name, web page, etc.), and may have some spatial footprint, which is a geometric feature in space, e.g., a point, a line, a polygon, etc. For now, we omit the detailed specification of Geometry, which may need its own pattern. The intention is to have Geometry aligned to the GeoSPARQL standard[1].

A.12.2 Axiomatization

IRI Declaration

```
Prefix: : <http://schema.geolink.org/dev/place#>
Ontology: <http://schema.geolink.org/dev/place>
Import: <http://schema.geolink.org/dev/place-to-informationobject>
Import: <http://schema.geolink.org/dev/place-to-geosparql>
ObjectProperty: hasSpatialFootprint, isDescribedBy
Class: Place, Geometry, InformationObject
```

Core Axioms

$$\mathsf{Place} \sqsubseteq (=1\ \mathsf{isDescribedBy}.\mathsf{InformationObject}) \quad \text{(A.231)}$$

[1] http://www.opengeospatial.org/standards/geosparql

$$\text{InformationObject} \sqsubseteq (=1\ \text{isDescribedBy}^{-}.\text{Place}) \tag{A.232}$$

$$\exists \text{isDescribedBy}.\text{InformationObject} \sqsubseteq \text{Place} \tag{A.233}$$

$$\text{Place} \sqsubseteq \forall \text{isDescribedBy}.\text{InformationObject} \tag{A.234}$$

$$\exists \text{hasSpatialFootprint}.\text{Geometry} \sqsubseteq \text{Place} \tag{A.235}$$

$$\text{Place} \sqsubseteq \forall \text{hasSpatialFootprint}.\text{Geometry} \tag{A.236}$$

A.12.3 Alignment

Alignment with Information Object pattern

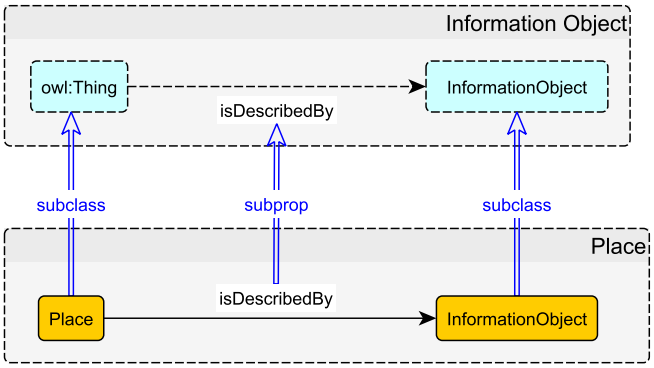


Figure A.39: The Place pattern stub aligned with Information Object pattern

```
Prefix: ecglpl: <http://schema.geolink.org/dev/place#>
Prefix: ecglio: <http://schema.geolink.org/dev/informationobject#>
Ontology: <http://schema.geolink.org/dev/place-to-informationobject>
ObjectProperty: ecglpl:isDescribedBy, ecglio:isDescribedBy
Class: ecglpl:InformationObject, ecglio:InformationObject
```

$$\text{ecglpl:InformationObject} \sqsubseteq \text{ecglio:InformationObject} \tag{A.237}$$

$$\text{ecglpl:isDescribedBy} \sqsubseteq \text{ecglio:isDescribedBy} \tag{A.238}$$

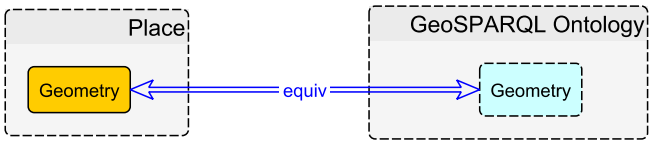


Figure A.40: The Place pattern stub aligned with GeoSPARQL ontology

Alignment with GeoSPARQL Ontology

```
Prefix: ecglpl: <http://schema.geolink.org/dev/place#>
Prefix: geosparql: <http://www.opengis.net/ont/geosparql#>
Ontology: <http://schema.geolink.org/dev/place-to-geosparql>
Class: ecglpl:Geometry, geosparql:Geometry
```

$$\text{ecglpl:Geometry} \equiv \text{geosparql:Geometry} \tag{A.239}$$

A.13 Cruise

A.13.1 Description

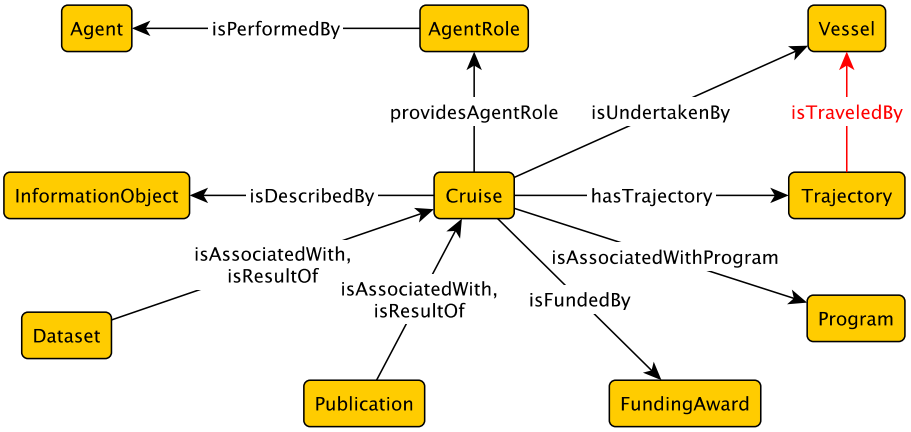


Figure A.41: The Cruise pattern overview

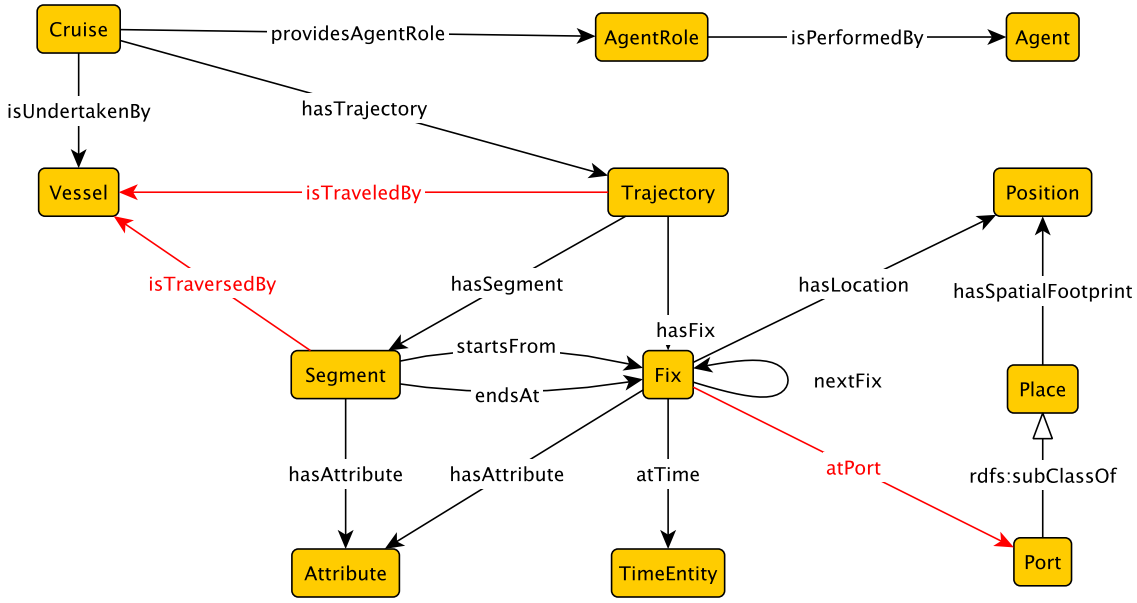


Figure A.42: The Cruise as Events: Trajectory and Agent Roles

The Cruise pattern has been discussed rather extensively in Section 4.5.3. The overview is given in Fig. A.41 and here, we restate the axioms, some of which will be presented as part of the alignments of the Cruise pattern with the other patterns. In

addition, we also include some discussion how this pattern can incorporate specific sets of user's vocabulary, e.g., regarding agent role types, cruise types, etc. Modeling of a cruise as an event whose spatiotemporal boundary is provided by its trajectory is done according to Fig. A.42 and the alignment of Cruise pattern with Event pattern as described in Appendix A.13.4.

A.13.2 Specific Vocabulary for Cruise

Specific vocabulary for cruise includes types of cruises, types of agent-roles for cruises, and attributes for fixes in the trajectory of a cruise. All of these are modeled according to the requests from the data providers. In the current version, cruise types are given as subclass of Cruise in Figure A.43. Agent roles for cruises are given in Figure A.44, while attributes of the fixes of the cruise are given in Figure A.45. These terms are defined within the Cruise pattern's URI namespace, but will be declared in a separate OWL file. This implies that the types of these agent-roles are really specific for the Cruise pattern. That is, for example, if there is a scientist role in a different pattern, then that is really a different role than the scientist role in the Cruise pattern. Finally, we model operational cruises by asserting that a cruise is operational, if and only if it has a chief scientist and is funded by some funding award.

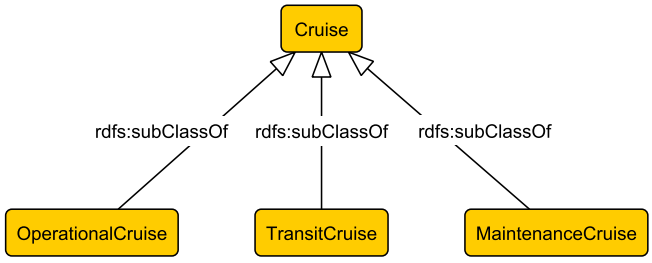


Figure A.43: Cruise types

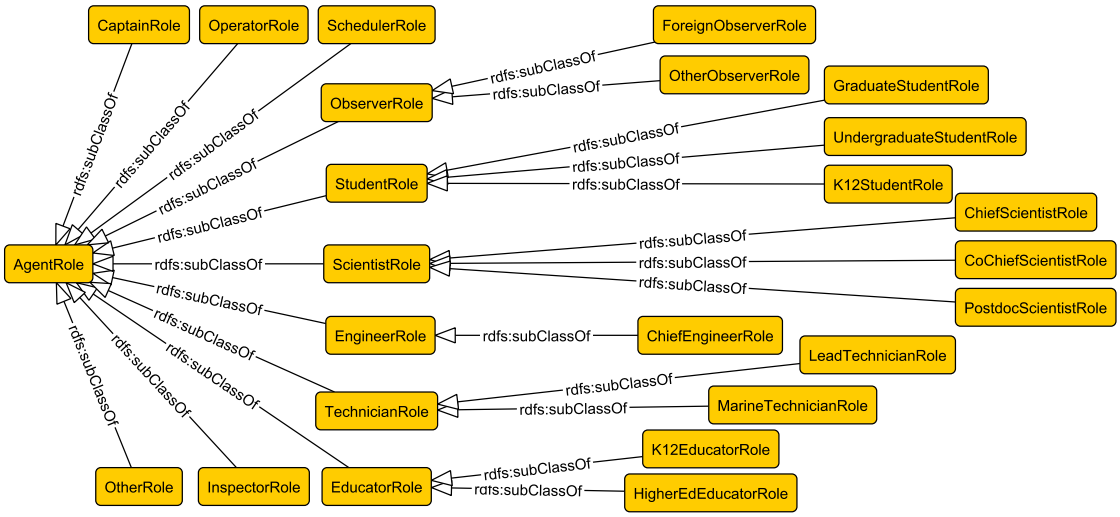


Figure A.44: Types of agent-role for a cruise

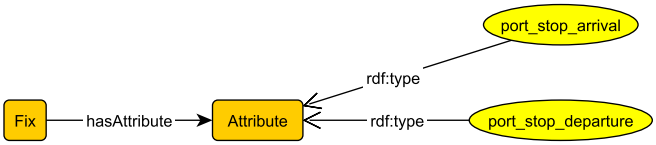


Figure A.45: Attributes for fixes in a cruise trajectory

A.13.3 Axiomatization

IRI Declaration

```
Prefix: : <http://schema.geolink.org/dev/cruise#>
Ontology: <http://schema.geolink.org/dev/cruise>
Import: <http://schema.geolink.org/dev/cruise-to-agent>
Import: <http://schema.geolink.org/dev/cruise-to-event>
Import: <http://schema.geolink.org/dev/cruise-to-program>
Import: <http://schema.geolink.org/dev/cruise-to-agentrole>
Import: <http://schema.geolink.org/dev/cruise-to-fundingaward>
Import: <http://schema.geolink.org/dev/cruise-to-place>
Import: <http://schema.geolink.org/dev/cruise-to-informationobject>
Import: <http://schema.geolink.org/dev/cruise-to-owltime>
Import: <http://schema.geolink.org/dev/cruise-to-vessel>
ObjectProperty: startsFrom, isTraversedBy, hasFix, nextFix, atPort,
                isAssociatedWithProgram,  hasSpatialFootprint, isTraveledBy,
                hasTrajectory, atTime, isUndertakenBy, providesAgentRole,
                isDescribedBy, isFundedBy, rollifiedPort, hasSegment,
```

```
                hasAttribute, endsAt, isPerformedBy, hasLocation,
                isAssociatedWith
Class: Port, Vessel, EndingFix, Segment, Place, FundingAward, Fix,
       InformationObject, Program, AgentRole, Position, Trajectory,
       Attribute, TimeEntity, Agent, Cruise, StartingFix
```

Core Axioms

We assert that a cruise is an event through the alignment with Event pattern (see Section A.13.4). We then assert that a cruise has exactly one trajectory, is undertaken by exactly one vessel, and is described by exactly one InformationObject. Additionally, this instance of InformationObject describes exactly only the cruise. We also assert that if a cruise is undertaken by a vessel, then the trajectory of the cruise has to be traveled by the vessel.

$$\mathsf{Cruise} \sqsubseteq (=1\ \mathsf{hasTrajectory}.\mathsf{Trajectory}) \tag{A.240}$$

$$\mathsf{Cruise} \sqsubseteq (=1\ \mathsf{isUndertakenBy}.\mathsf{Vessel}) \tag{A.241}$$

$$\mathsf{Cruise} \sqsubseteq (=1\ \mathsf{isDescribedBy}.\mathsf{InformationObject}) \tag{A.242}$$

$$\mathsf{InformationObject} \sqsubseteq (=1\ \mathsf{isDescribedBy}^{-}.\mathsf{Cruise}) \tag{A.243}$$

$$\mathsf{hasTrajectory}^{-} \circ \mathsf{isUndertakenBy} \sqsubseteq \mathsf{isTraveledBy} \tag{A.244}$$

Note that since isTraveledBy is implied by a property chain, OWL 2 specification forbids us to express a cardinality restriction using this peroperty. Consequently, we cannot axiomatize that the trajectory of a cruise can only be traveled by one vessel. Next, we state the guarded domain and range restrictions for the properties mentioned above.

$$\exists \mathsf{hasTrajectory}.\mathsf{Trajectory} \sqsubseteq \mathsf{Cruise} \tag{A.245}$$

$$\mathsf{Cruise} \sqsubseteq \forall \mathsf{hasTrajectory}.\mathsf{Trajectory} \tag{A.246}$$

$$\exists \mathsf{isUndertakenBy}.\mathsf{Vessel} \sqsubseteq \mathsf{Cruise} \tag{A.247}$$

$$\mathsf{Cruise} \sqsubseteq \forall \mathsf{isUndertakenBy.Vessel} \tag{A.248}$$

$$\exists \mathsf{isDescribedBy.InformationObject} \sqsubseteq \mathsf{Cruise} \tag{A.249}$$

$$\mathsf{Cruise} \sqsubseteq \forall \mathsf{isDescribedBy.InformationObject} \tag{A.250}$$

$$\exists \mathsf{isTraveledBy.Vessel} \sqsubseteq \mathsf{Trajectory} \tag{A.251}$$

$$\mathsf{Trajectory} \sqsubseteq \forall \mathsf{isTraveledBy.Vessel} \tag{A.252}$$

$$\exists \mathsf{isAssociatedWith.Cruise} \sqsubseteq \mathsf{Publication} \sqcup \mathsf{Dataset} \tag{A.253}$$

$$\mathsf{Publication} \sqcup \mathsf{Dataset} \sqsubseteq \forall \mathsf{isAssociatedWith.Dataset} \tag{A.254}$$

$$\exists \mathsf{isResultOf.Cruise} \sqsubseteq \mathsf{Publication} \sqcup \mathsf{Dataset} \tag{A.255}$$

$$\mathsf{Publication} \sqcup \mathsf{Dataset} \sqsubseteq \forall \mathsf{isResultOf.Dataset} \tag{A.256}$$

$$\exists \mathsf{isFundedBy.FundingAward} \sqsubseteq \mathsf{Cruise} \tag{A.257}$$

$$\mathsf{Cruise} \sqsubseteq \forall \mathsf{isFundedBy.FundingAward} \tag{A.258}$$

$$\exists \mathsf{isAssociatedWithProgram.Program} \sqsubseteq \mathsf{Cruise} \tag{A.259}$$

$$\mathsf{Program} \sqsubseteq \forall \mathsf{isAssociatedWithProgram.Program} \tag{A.260}$$

Axioms for Cruise Trajectory

Some of the axioms relevant to the cruise trajectory are obtained by reusing, with some modifications, the axioms of from Semantic Trajectory pattern [68]. For clarity and completeness, we will restate those axioms here as needed.

We begin describing the cruise trajectory by defining its basic components: fixes and segments. A fix has a location and a time stamp, and always belongs to one particular trajectory. Also, a fix cannot be followed by more than one other fix, and cannot follow itself. This gives a linear structure in the ordering of the fixes.

$$\mathsf{Fix} \sqsubseteq \exists \mathsf{hasLocation.Position} \sqcap \exists \mathsf{atTime.TimeEntity} \sqcap (=1\ \mathsf{hasFix}^{-}.\mathsf{Trajectory}) \tag{A.261}$$

$$\mathsf{Fix} \sqsubseteq (\leqslant 1\ \mathsf{nextFix.Fix}) \sqcap \neg \exists \mathsf{nextFix.Self} \tag{A.262}$$

We next define starting and ending fixes as special kinds of fixes.

$$\mathsf{StartingFix} \equiv \mathsf{Fix} \sqcap \neg\exists \mathsf{nextFix}^{-}.\top \tag{A.263}$$

$$\mathsf{EndingFix} \equiv \mathsf{Fix} \sqcap \neg\exists \mathsf{nextFix}.\top \tag{A.264}$$

$$\mathsf{StartingFix} \sqcap \mathsf{EndingFix} \sqsubseteq \bot \tag{A.265}$$

A trajectory is linked to at least two consecutive fixes where the first fix is the starting fix. Also, if a fix belongs to a trajectory, then its successor fix also belongs to the same trajectory.

$$\mathsf{Trajectory} \sqsubseteq \exists \mathsf{hasFix}.(\mathsf{StartingFix} \sqcap \exists \mathsf{nextFix}.\mathsf{Fix}) \tag{A.266}$$

$$\mathsf{hasFix} \circ \mathsf{nextFix} \sqsubseteq \mathsf{hasFix} \tag{A.267}$$

A segment starts from exactly one fix, and for every fix with a successor fix, there is a segment that starts from it. If a fix belongs to a trajectory and there is a segment that starts from this fix, then the segment belongs to the trajectory. Furthermore, if a segment starts from a fix, then it ends at the successor of the fix.

$$\mathsf{Segment} \sqsubseteq (=1\ \mathsf{startsFrom}.\mathsf{Fix}) \tag{A.268}$$

$$\exists \mathsf{nextFix}.\mathsf{Fix} \sqsubseteq (=1\ \mathsf{startsFrom}^{-}.\mathsf{Segment}) \tag{A.269}$$

$$\mathsf{hasFix} \circ \mathsf{startsFrom}^{-} \sqsubseteq \mathsf{hasSegment} \tag{A.270}$$

$$\mathsf{startsFrom} \circ \mathsf{nextFix} \sqsubseteq \mathsf{endsAt} \tag{A.271}$$

The above axiomatization ensures that a trajectory is linked to all of its fixes and segments. Note that the above axioms do not model a trajectory to have a finite sequence of fixes of unknown length, which cannot actually be modeled in OWL 2. In our case, however, data providers will only provide cruise trajectory as a finite collection of fixes with a known ordering, which can be written as a set of ABox axioms of the form $\mathsf{Fix}(f_1), \ldots, \mathsf{Fix}(f_n), \mathsf{nextFix}(f_1, f_2), \ldots, \mathsf{nextFix}(f_{n-1}, f_n), \mathsf{StartingFix}(f_1), \mathsf{EndingFix}(f_n)$.

Since a fix cannot have more than one successor fix, we implicitly obtain a finite, linear ordering given by the transitive closure of nextFix.

We next define atPort as a shortcut via property chain involving hasLocation and hasSpatialFootprint, which can be written in Datalog as: $\mathsf{hasLocation}(x, y), \mathsf{hasSpatialFootprint}(z, y), \mathsf{Port}(z) \rightarrow \mathsf{atPort}(x, z)$. The following two axioms express the rule where rollifiedPort is a fresh property name defined solely for the class Port, which is defined as a subclass of Place.

$$\mathsf{hasLocation} \circ \mathsf{hasSpatialFootprint}^{-} \circ \mathsf{rollifiedPort} \sqsubseteq \mathsf{atPort} \tag{A.272}$$

$$\exists \mathsf{rollifiedPort.Self} \equiv \mathsf{Port} \tag{A.273}$$

$$\mathsf{Port} \sqsubseteq \mathsf{Place} \tag{A.274}$$

If a trajectory is traveled by a vessel, then every segment is traversed by that vessel.

$$\mathsf{hasSegment}^{-} \circ \mathsf{isTraveledBy} \sqsubseteq \mathsf{isTraversedBy} \tag{A.275}$$

We assert the following guarded domain and range restrictions.

$$\exists \mathsf{hasFix.Fix} \sqsubseteq \mathsf{Trajectory} \tag{A.276}$$

$$\mathsf{Trajectory} \sqsubseteq \forall \mathsf{hasFix.Fix} \tag{A.277}$$

$$\exists \mathsf{nextFix.Fix} \sqsubseteq \mathsf{Fix} \tag{A.278}$$

$$\mathsf{Fix} \sqsubseteq \forall \mathsf{nextFix.Fix} \tag{A.279}$$

$$\exists \mathsf{hasLocation.Position} \sqsubseteq \mathsf{Fix} \tag{A.280}$$

$$\mathsf{Fix} \sqsubseteq \forall \mathsf{hasLocation.Position} \tag{A.281}$$

$$\exists \mathsf{atPort.Port} \sqsubseteq \mathsf{Fix} \tag{A.282}$$

$$\mathsf{Fix} \sqsubseteq \forall \mathsf{atPort.Port} \tag{A.283}$$

$$\exists \mathsf{atTime.TimeEntity} \sqsubseteq \mathsf{Fix} \tag{A.284}$$

$$\mathsf{Fix} \sqsubseteq \forall \mathsf{atTime.TimeEntity} \tag{A.285}$$

$$\exists \mathsf{hasSpatialFootprint.Position} \sqsubseteq \mathsf{Place} \tag{A.286}$$

$$\mathsf{Place} \sqsubseteq \forall \mathsf{hasSpatialFootprint.Position} \tag{A.287}$$

$$\exists \mathsf{hasSegment.Segment} \sqsubseteq \mathsf{Trajectory} \tag{A.288}$$

$$\mathsf{Trajectory} \sqsubseteq \forall \mathsf{hasSegment.Segment} \tag{A.289}$$

$$\exists \mathsf{startsFrom.Fix} \sqsubseteq \mathsf{Segment} \tag{A.290}$$

$$\mathsf{Segment} \sqsubseteq \forall \mathsf{startsFrom.Fix} \tag{A.291}$$

$$\exists \mathsf{endsAt.Fix} \sqsubseteq \mathsf{Segment} \tag{A.292}$$

$$\mathsf{Segment} \sqsubseteq \forall \mathsf{endsAt.Fix} \tag{A.293}$$

$$\exists \mathsf{isTraversedBy.Vessel} \sqsubseteq \mathsf{Segment} \tag{A.294}$$

$$\mathsf{Segment} \sqsubseteq \forall \mathsf{isTraversedBy.Vessel} \tag{A.295}$$

$$\exists \mathsf{hasAttribute.Attribute} \sqsubseteq \mathsf{Segment} \sqcup \mathsf{Fix} \tag{A.296}$$

$$\mathsf{Fix} \sqsubseteq \forall \mathsf{hasAttribute.Attribute} \tag{A.297}$$

$$\mathsf{Segment} \sqsubseteq \forall \mathsf{hasAttribute.Attribute} \tag{A.298}$$

Axioms for Cruise Agent Roles

We next model the actors of a cruise, which is achieved by aligning with Agent Role pattern. In this context, a cruise may provide a numbef of special agent-roles performed by some agent. Various types of agent-roles a cruise may provide are included in a class hierarchy rooted at the AgentRole class, as specified in the nomenclature part of the Cruise micro-ontology. For now, we simply state the domain and range restrictions, as well as assert that every agent-role has to be performed by exactly one agent.

$$\exists \mathsf{providesAgentRole.AgentRole} \sqsubseteq \mathsf{Cruise} \tag{A.299}$$

$$\mathsf{Cruise} \sqsubseteq \forall \mathsf{providesAgentRole.AgentRole} \tag{A.300}$$

$$\exists \mathsf{isPerformedBy.Agent} \sqsubseteq \mathsf{AgentRole} \tag{A.301}$$

$$\mathsf{AgentRole} \sqsubseteq \forall \mathsf{isPerformedBy.Agent} \tag{A.302}$$

$$\mathsf{AgentRole} \sqsubseteq (=1\ \mathsf{isPerformedBy.Agent}) \tag{A.303}$$

Class Disjointness Axioms

We assert the following class disjointness axioms:

$$\begin{aligned}\mathsf{alldisjoint}(&\mathsf{Cruise}, \mathsf{InformationObject}, \mathsf{AgentRole}, \mathsf{Agent}, \mathsf{FundingAward}, \mathsf{Program},\\ &\mathsf{Trajectory}, \mathsf{Vessel}, \mathsf{Fix}, \mathsf{Segment}, \mathsf{Attribute}, \mathsf{TimeEntity}, \mathsf{Place}, \mathsf{Position},\\ &\mathsf{Dataset}, \mathsf{Publication})\end{aligned} \tag{A.304}$$

Axioms for Nomenclature of Cruise

For nomenclature part of the Cruise micro-ontology, we specify a few specific cruise types, which are built into a class hierarchy by the following:

$$\mathsf{OperationalCruise} \sqcup \mathsf{MaintenanceCruise} \sqcup \mathsf{TransitCruise} \sqsubseteq \mathsf{Cruise} \tag{A.305}$$

Note that if a cruise is neither operational, nor in maintenance, nor in transit, then there is no need to specify its type. Furthermore, a cruise is operational if and only if it has a chief scientist and is funded by some funding award.

$$\begin{aligned}\mathsf{OperationalCruise} \equiv \mathsf{Cruise} &\sqcap \exists \mathsf{providesAgentRole.ChiefScientistRole}\\ &\sqcap \exists \mathsf{isDescribedBy.}\exists \mathsf{isFundedBy.FundingAward}\end{aligned} \tag{A.306}$$

The nomenclature part also establishes a predefined set of cruise agent-role types.

$$\mathsf{CaptainRole} \sqcup \mathsf{OperatorRole} \sqcup \mathsf{SchedulerRole} \sqsubseteq \mathsf{AgentRole} \tag{A.307}$$

$$\mathsf{ObserverRole} \sqcup \mathsf{InspectorRole} \sqsubseteq \mathsf{AgentRole} \tag{A.308}$$

$$\mathsf{ForeignObserverRole} \sqcup \mathsf{OtherObserverRole} \sqsubseteq \mathsf{ObserverRole} \tag{A.309}$$

$$\mathsf{EngineerRole} \sqcup \mathsf{ScientistRole} \sqcup \mathsf{TechnicianRole} \sqsubseteq \mathsf{AgentRole} \tag{A.310}$$

$$\mathsf{ChiefEngineerRole} \sqsubseteq \mathsf{EngineerRole} \tag{A.311}$$

$$\mathsf{ChiefScientistRole} \sqcup \mathsf{CoChiefScientistRole} \sqsubseteq \mathsf{ScientistRole} \tag{A.312}$$

$$\mathsf{PostdocScientistRole} \sqsubseteq \mathsf{ScientistRole} \tag{A.313}$$

$$\mathsf{LeadTechnicianRole} \sqcup \mathsf{MarineTechnicianRole} \sqsubseteq \mathsf{TechnicianRole} \tag{A.314}$$

$$\mathsf{StudentRole} \sqcup \mathsf{EducatorRole} \sqsubseteq \mathsf{AgentRole} \tag{A.315}$$

$$\mathsf{GraduateStudentRole} \sqcup \mathsf{UndergraduateStudentRole} \sqsubseteq \mathsf{StudentRole} \tag{A.316}$$

$$\mathsf{K12StudentRole} \sqsubseteq \mathsf{StudentRole} \tag{A.317}$$

$$\mathsf{HigherEdEducatorRole} \sqcup \mathsf{K12EducatorRole} \sqsubseteq \mathsf{EducatorRole} \tag{A.318}$$

Finally, the nomenclature also contains a few attributes of fixes in a cruise trajectory. They are represented as named individuals, hence reside in the voc namespace. The following ABox axioms provide axiomatization for these attributes.

$$\mathsf{Attribute}(\mathsf{port\_stop\_departure}), \mathsf{Attribute}(\mathsf{port\_stop\_arrival}) \tag{A.319}$$

A.13.4 Alignment

Alignment with Agent pattern

```
Prefix: ecglag: <http://schema.geolink.org/dev/agent#>
Prefix: ecglcr: <http://schema.geolink.org/dev/cruise#>
Ontology: <http://schema.geolink.org/dev/cruise-to-agent>
ObjectProperty: ecglcr:isPerformedBy,  ecglag:performsAgentRole
Class: ecglcr:Agent, ecglcr:AgentRole, ecglag:Agent, ecglag:AgentRole
```

$$\mathsf{ecglcr{:}Agent} \sqsubseteq \mathsf{ecglag{:}Agent} \tag{A.320}$$

$$\mathsf{ecglcr{:}AgentRole} \sqsubseteq \mathsf{ecglag{:}AgentRole} \tag{A.321}$$

$$\mathsf{ecglcr{:}isPerformedBy} \sqsubseteq \mathsf{ecglag{:}performsAgentRole}^{-} \tag{A.322}$$

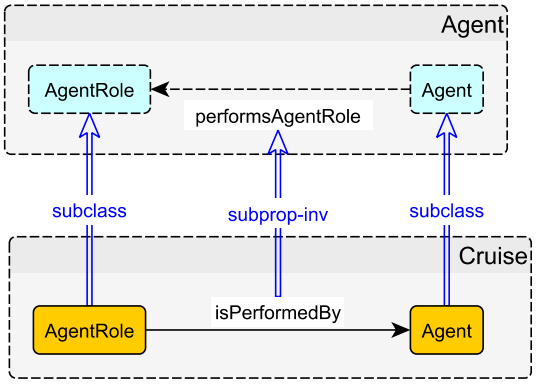


Figure A.46: The Cruise micro-ontology alignment with Agent pattern

Alignment with Agent Role pattern

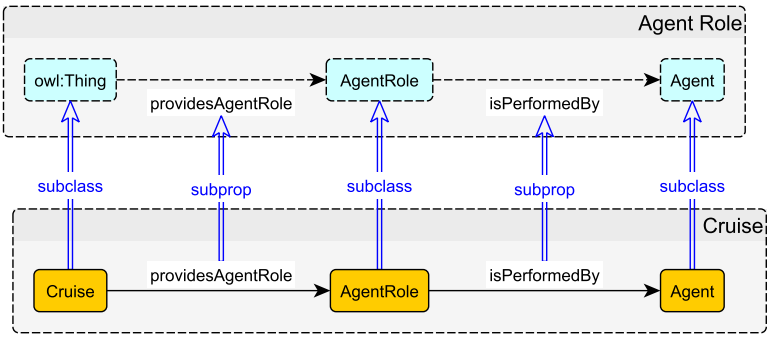


Figure A.47: The Cruise micro-ontology alignment with Agent Role pattern

```
Prefix: ecglar: <http://schema.geolink.org/dev/agentrole#>
Prefix: ecglcr: <http://schema.geolink.org/dev/cruise#>
Ontology: <http://schema.geolink.org/dev/cruise-to-agentrole>
ObjectProperty: ecglcr:providesAgentRole, ecglcr:isPerformedBy,
                ecglar:providesAgentRole, ecglar:isPerformedBy
Class: ecglcr:AgentRole, ecglcr:Agent, ecglar:AgentRole, ecglar:Agent
```

$$\text{ecglcr:Agent} \sqsubseteq \text{ecglar:Agent} \tag{A.323}$$

$$\text{ecglcr:AgentRole} \sqsubseteq \text{ecglar:AgentRole} \tag{A.324}$$

$$\text{ecglcr:providesAgentRole} \sqsubseteq \text{ecglar:providesAgentRole} \tag{A.325}$$

$$\text{ecglcr:isPerformedBy} \sqsubseteq \text{ecglar:isPerformedBy} \tag{A.326}$$

Alignment with Event pattern

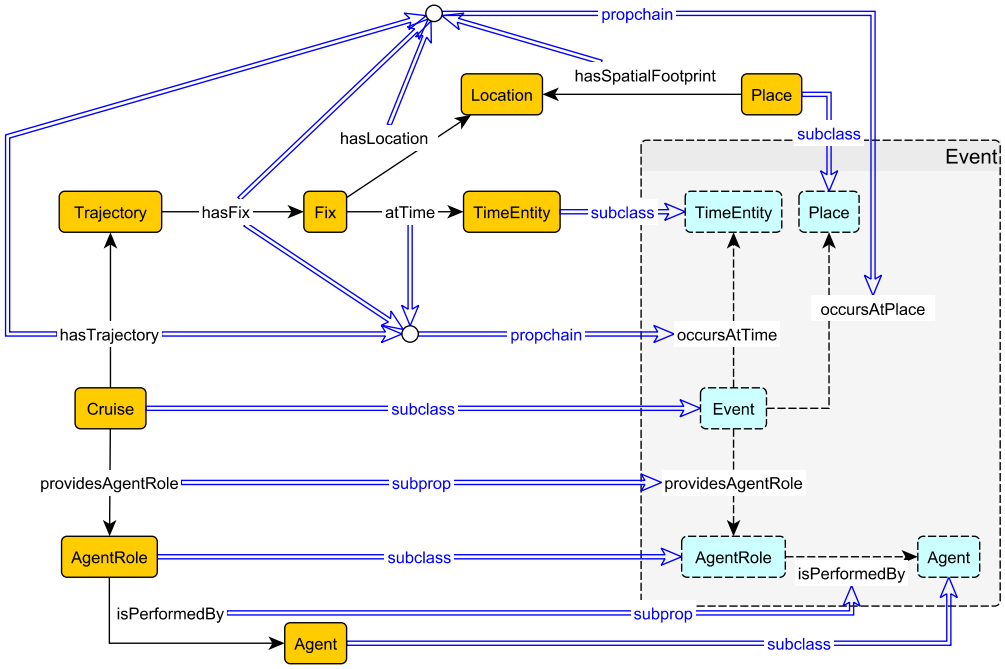


Figure A.48: The Cruise pattern alignment with Event pattern

```
Prefix: ecglev: <http://schema.geolink.org/dev/event#>
Prefix: ecglcr: <http://schema.geolink.org/dev/cruise#>
Ontology: <http://schema.geolink.org/dev/cruise-to-event>
ObjectProperty: ecglcr:hasLocation, ecglcr:hasSpatialFootprint,
                ecglcr:atPort, ecglcr:atTime, ecglcr:hasFix,
                ecglcr:hasTrajectory, ecglcr:providesAgentRole,
                ecglcr:isPerformedBy, ecglev:occursAtPlace,
                ecglev:occursAtTime, ecglev:providesAgentRole,
                ecglev:isPerformedBy
Class: ecglcr:Cruise, ecglcr:AgentRole, ecglcr:Agent, ecglcr:Port,
       ecglcr:TimeEntity, ecglcr:Event, ecglev:AgentRole, ecglev:Agent,
       ecglev:Place, ecglev:TimeEntity
```

$$\text{ecglcr:Cruise} \sqsubseteq \text{ecglev:Event} \tag{A.327}$$

$$\text{ecglcr:Port} \sqsubseteq \text{ecglev:Place} \tag{A.328}$$

$$\text{ecglcr:TimeEntity} \sqsubseteq \text{ecglev:TimeEntity} \tag{A.329}$$

$$\text{ecglcr:AgentRole} \sqsubseteq \text{ecglev:AgentRole} \tag{A.330}$$

$$\text{ecglcr:Agent} \sqsubseteq \text{ecglev:Agent} \tag{A.331}$$

$$\text{ecglcr:hasTrajectory} \circ \text{ecglcr:hasFix} \circ \text{ecglcr:hasLocation} \circ \text{ecglcr:hasSpatialFootprint}^{-} \sqsubseteq \text{ecglev:occursAtPlace} \tag{A.332}$$

$$\text{ecglcr:hasTrajectory} \circ \text{ecglcr:hasFix} \circ \text{ecglcr:atTime} \sqsubseteq \text{ecglev:occursAtTime} \tag{A.333}$$

$$\text{ecglcr:providesAgentRole} \sqsubseteq \text{ecglev:providesAgentRole} \tag{A.334}$$

$$\text{ecglcr:isPerformedBy} \sqsubseteq \text{ecglev:isPerformedBy} \tag{A.335}$$

Alignment with Funding Award pattern

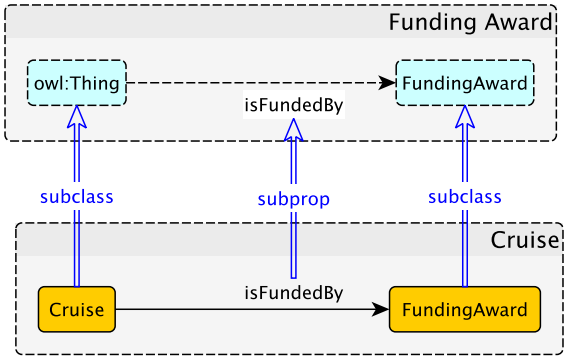


Figure A.49: The Cruise pattern alignment with Funding Award pattern

```
Prefix: ecglfa: <http://schema.geolink.org/dev/fundingaward#>
Prefix: ecglcr: <http://schema.geolink.org/dev/cruise#>
Ontology: <http://schema.geolink.org/dev/cruise-to-fundingaward>
ObjectProperty: ecglcr:isFundedBy, ecglfa:isFundedBy
Class: ecglcr:Cruise, ecglcr:FundingAward, ecglfa:FundingAward
```

$$\text{ecglcr:FundingAward} \sqsubseteq \text{ecglfa:FundingAward} \tag{A.336}$$

$$\text{ecglcr:isFundedBy} \sqsubseteq \text{ecglfa:isFundedBy} \tag{A.337}$$

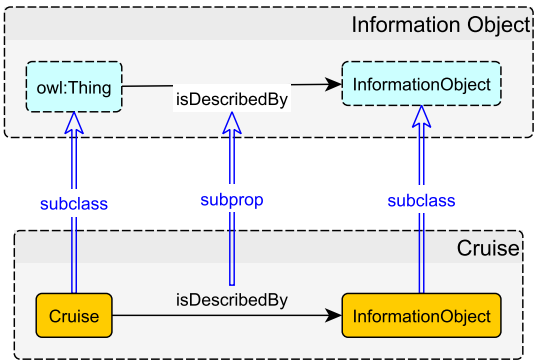


Figure A.50: The Cruise pattern alignment with Information Object pattern

Alignment with Information Object pattern

```
Prefix: ecglio: <http://schema.geolink.org/dev/informationobject#>
Prefix: ecglcr: <http://schema.geolink.org/dev/cruise#>
Ontology: <http://schema.geolink.org/dev/cruise-to-informationobject>
ObjectProperty: ecglcr:isDescribedBy, ecglio:isDescribedBy
Class: ecglcr:InformationObject, ecglio:InformationObject
```

$$\text{ecglcr:InformationObject} \sqsubseteq \text{ecglio:InformationObject} \tag{A.338}$$

$$\text{ecglcr:isDescribedBy} \sqsubseteq \text{ecglio:isDescribedBy} \tag{A.339}$$

Alignment with OWL Time ontology

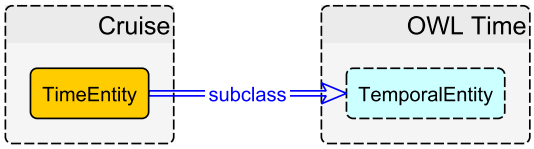


Figure A.51: The Cruise pattern alignment with OWL Time ontology

```
Prefix: time: <http://www.w3.org/2006/time#>
Prefix: ecglcr: <http://schema.geolink.org/dev/cruise#>
Ontology: <http://schema.geolink.org/dev/cruise-to-owltime>
Class: ecglcr:TimeEntity, time:TemporalEntity
```

$$\text{ecglcr:TimeEntity} \sqsubseteq \text{time:TemporalEntity} \tag{A.340}$$

Alignment with Place pattern

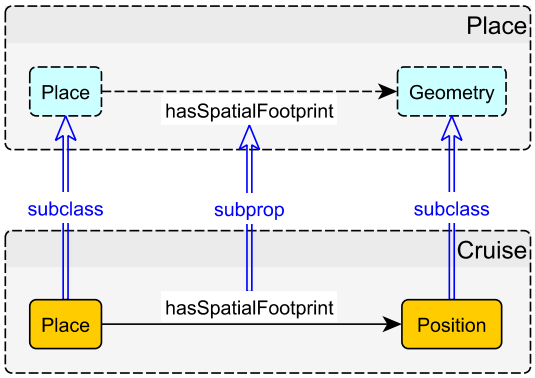


Figure A.52: The Cruise pattern alignment with Place pattern

```
Prefix: ecglpl: <http://schema.geolink.org/dev/place#>
Prefix: ecglcr: <http://schema.geolink.org/dev/cruise#>
Ontology: <http://schema.geolink.org/dev/cruise-to-place>
ObjectProperty: ecglcr:hasSpatialFootprint, ecglpl:hasSpatialFootprint
Class: ecglcr:Place, ecglcr:Position, ecglpl:Place, ecglpl:Geometry
```

$$\text{ecglcr:Place} \sqsubseteq \text{ecglpl:Place} \tag{A.341}$$

$$\text{ecglcr:Position} \sqsubseteq \text{ecglpl:Geometry} \tag{A.342}$$

$$\text{ecglcr:hasSpatialFootprint} \sqsubseteq \text{ecglpl:hasSpatialFootprint} \tag{A.343}$$

Alignment with Program pattern

```
Prefix: ecglpg: <http://schema.geolink.org/dev/program#>
Prefix: ecglcr: <http://schema.geolink.org/dev/cruise#>
Ontology: <http://schema.geolink.org/dev/cruise-to-program>
ObjectProperty: ecglcr:isAssociatedWith, ecglpg:isAssociatedWith
Class: ecglcr:Program, ecglpg:Program
```

$$\text{ecglcr:Program} \sqsubseteq \text{ecglpg:Program} \tag{A.344}$$

$$\text{ecglcr:isAssociatedWith} \sqsubseteq \text{ecglpg:isAssociatedWith} \tag{A.345}$$

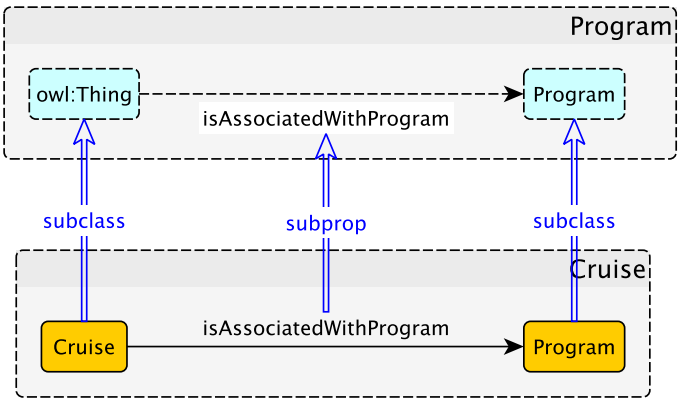


Figure A.53: The Cruise pattern alignment with Program pattern

Alignment with Vessel pattern

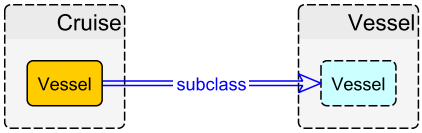


Figure A.54: The Cruise pattern alignment with Vessel pattern

```
Prefix: ecglvs: <http://schema.geolink.org/dev/vessel#>
Prefix: ecglcr: <http://schema.geolink.org/dev/cruise#>
Ontology: <http://schema.geolink.org/dev/cruise-to-vessel>
Class: ecglcr:Vessel, ecglvs:Vessel
```

$$\text{ecglcr:Vessel} \sqsubseteq \text{ecglvs:Vessel} \tag{A.346}$$

A.14 Platform Pattern

A.14.1 Description

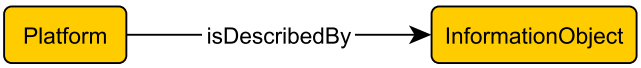


Figure A.55: Platform pattern

This is a very simple pattern stub, depicted in Fig. A.55, for representing ocean science platforms, which include vessels, submersibles, aircrafts, etc. No detail is modeled in this pattern, except for associating a platform with its information object. A particular specialization of this pattern is the Vessel pattern, presented in Appendix A.15.

A.14.2 Axiomatization

IRI Declarations

```
Prefix: : <http://schema.geolink.org/dev/platform#>
Ontology: <http://schema.geolink.org/dev/platform>
Import: <http://schema.geolink.org/dev/platform-to-informationobject>
ObjectProperty: isDescribedBy
Class: Platform, InformationObject
```

Core Axioms

$$\mathsf{Platform} \sqsubseteq (=1\ \mathsf{isDescribedBy}.\mathsf{InformationObject}) \tag{A.347}$$

$$\exists \mathsf{isDescribedBy}.\mathsf{InformationObject} \sqsubseteq \mathsf{Platform} \tag{A.348}$$

$$\mathsf{Platform} \sqsubseteq \forall \mathsf{isDescribedBy}.\mathsf{InformationObject} \tag{A.349}$$

$$\mathsf{Platform} \sqcap \mathsf{InformationObject} \sqsubseteq \bot \tag{A.350}$$

A.14.3 Alignment

Alignment with Information Object pattern

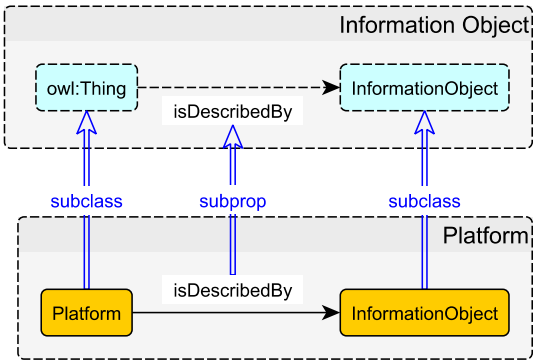


Figure A.56: The Platform pattern alignment with Information Object pattern

```
Prefix: ecglio: <http://schema.geolink.org/dev/informationobject#>
Prefix: ecglpf: <http://schema.geolink.org/dev/platform#>
Ontology: <http://schema.geolink.org/dev/platform-to-informationobject>
ObjectProperty: ecglpf:isDescribedBy, ecglio:isDescribedBy
Class: ecglpf:InformationObject, ecglio:InformationObject
```

$$\text{ecglpf:InformationObject} \sqsubseteq \text{ecglio:InformationObject} \tag{A.351}$$

$$\text{ecglpf:isDescribedBy} \sqsubseteq \text{ecglio:isDescribedBy} \tag{A.352}$$

A.15 Vessel pattern

A.15.1 Description

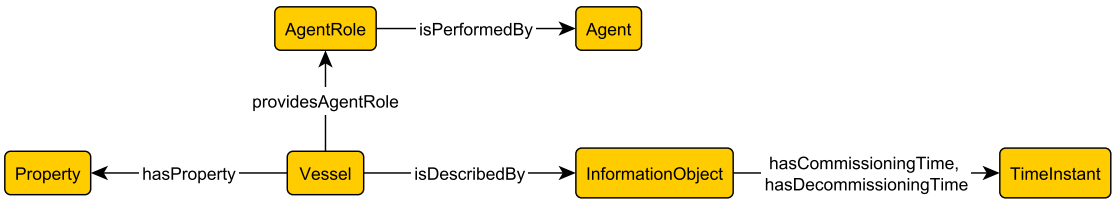


Figure A.57: Vessel pattern

The Vessel pattern represents vessels on which cruises are carried out. A vessel is considered a type of platform, hence this pattern is aligned to the Platform pattern from Appendix A.14. Each vessel can be described in terms of observable (physical) properties, e.g., length, beam, draft, and displacement. We reuse/align to the Property Value pattern to model such observable properties. Here, each property would have a name, a value, and a unit. The modeling of these properties can also be aligned to the *QUDT – Quantities, Units, Dimensions and Data Types Ontologies*[2].

A vessel may provide some agent-roles performed by some agents. An obvious example, a vessel most likely has an owner or affiliated with an organization. This modeled by re-using the Agent Role pattern.

Other pieces of information regarding a vessel that may be useful are included through the alignment withthe Information Object pattern. Such information does not only include vessel identifiers, names, and webpage, which can already be accommodated by the original Information Object pattern, but also other information such as year of commission and decommission, etc.

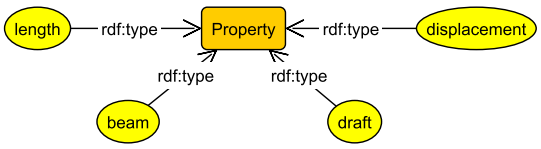


Figure A.58: Vessel property

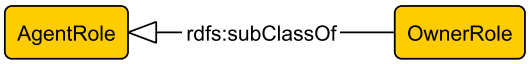


Figure A.59: Vessel agent roles

A.15.2 Specific Nomenclature for Vessel pattern

Nomenclature part of the Vessel pattern includes terms describing properties relevant for a vessel (Fig. A.58) and types of agent-roles provided by a vessel (Fig. A.59). In this part, properties of every vessel includes a length, beam, draft, and displacement. In addition, a vessel provides an owner role. Note that the owner role here is specific for vessels, and this is realized by defining it under Vessel pattern's URI namespace.

A.15.3 Axiomatization

IRI Declarations

```
Prefix: : <http://schema.geolink.org/dev/vessel#>
Ontology: <http://schema.geolink.org/dev/vessel>
Import: <http://schema.geolink.org/dev/vessel-to-informationobject>
Import: <http://schema.geolink.org/dev/vessel-to-property>
Import: <http://schema.geolink.org/dev/vessel-to-agentrole>
ObjectProperty: isDescribedBy, hasCommissioningTime, hasDecommissioningTime,
                hasProperty, providesAgentRole, isPerformedBy
Class: Vessel, InformationObject, TimeInstant, Property, AgentRole, Agent
```

[2] http://www.qudt.org/

Core Axioms

Every vessel has exactly one information object. Also, we assert the guarded domain and range restrictions.

$$\mathsf{Vessel} \sqsubseteq (=1\ \mathsf{isDescribedBy.InformationObject}) \tag{A.353}$$

$$\exists \mathsf{isDescribedBy.InformationObject} \sqsubseteq \mathsf{Vessel} \tag{A.354}$$

$$\mathsf{Vessel} \sqsubseteq \forall \mathsf{isDescribedBy.InformationObject} \tag{A.355}$$

$$\exists \mathsf{hasProperty.Property} \sqsubseteq \mathsf{Vessel} \tag{A.356}$$

$$\mathsf{Vessel} \sqsubseteq \forall \mathsf{hasProperty.Property} \tag{A.357}$$

$$\exists \mathsf{providesAgentRole.AgentRole} \sqsubseteq \mathsf{Vessel} \tag{A.358}$$

$$\mathsf{Vessel} \sqsubseteq \forall \mathsf{providesAgentRole.AgentRole} \tag{A.359}$$

$$\exists \mathsf{isPerformedBy.Agent} \sqsubseteq \mathsf{AgentRole} \tag{A.360}$$

$$\mathsf{AgentRole} \sqsubseteq \forall \mathsf{isPerformedBy.Agent} \tag{A.361}$$

$$\mathsf{AgentRole} \sqsubseteq (=1\ \mathsf{isPerformedBy.Agent}) \tag{A.362}$$

Class disjointness axiom:

$$\mathsf{alldisjoint(Vessel, InformationObject, TimeInstant, AgentRole, Agent, Property)} \tag{A.363}$$

Axioms for Vessel Nomenclature

$$\mathsf{OwnerRole} \sqsubseteq \mathsf{AgentRole} \tag{A.364}$$

$$\mathsf{Property(length), Property(beam), Property(draft), Property(displacement)} \tag{A.365}$$

A.15.4 Alignment

Alignment with Agent pattern

```
Prefix: ecglag: <http://schema.geolink.org/dev/agent#>
```

```
Prefix: ecglvs: <http://schema.geolink.org/dev/vessel#>

Ontology: <http://schema.geolink.org/dev/vessel-to-agent>
ObjectProperty: ecglvs:isPerformedBy, ecglag:performsAgentRole
Class: ecglvs:Agent, ecglvs:AgentRole, ecglag:Agent, ecglag:AgentRole
```

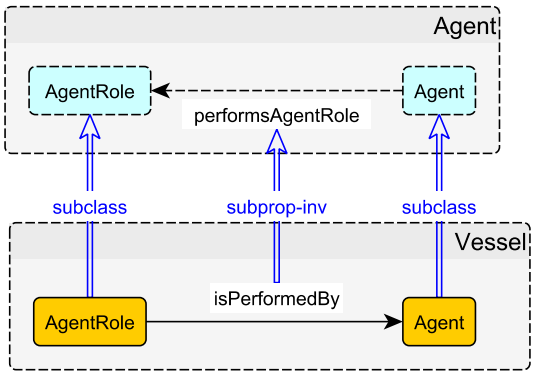


Figure A.60: The Vessel pattern alignment with Agent pattern

$$\text{ecglvs:Agent} \sqsubseteq \text{ecglag:Agent} \tag{A.366}$$

$$\text{ecglvs:AgentRole} \sqsubseteq \text{ecglag:AgentRole} \tag{A.367}$$

$$\text{ecglvs:isPerformedBy} \sqsubseteq \text{ecglag:performsAgentRole}^{-} \tag{A.368}$$

Alignment with Agent Role pattern

```
Prefix: ecglar: <http://schema.geolink.org/dev/agentrole#>
Prefix: ecglvs: <http://schema.geolink.org/dev/vessel#>
Ontology: <http://schema.geolink.org/dev/vessel-to-agentrole>
ObjectProperty: ecglvs:providesAgentRole, ecglvs:isPerformedBy,
                ecglar:providesAgentRole,  ecglar:isPerformedBy
Class: ecglvs:AgentRole, ecglvs:Agent, ecglar:AgentRole, ecglar:Agent
```

$$\text{ecglvs:Agent} \sqsubseteq \text{ecglar:Agent} \tag{A.369}$$

$$\text{ecglvs:AgentRole} \sqsubseteq \text{ecglar:AgentRole} \tag{A.370}$$

$$\text{ecglvs:providesAgentRole} \sqsubseteq \text{ecglar:providesAgentRole} \tag{A.371}$$

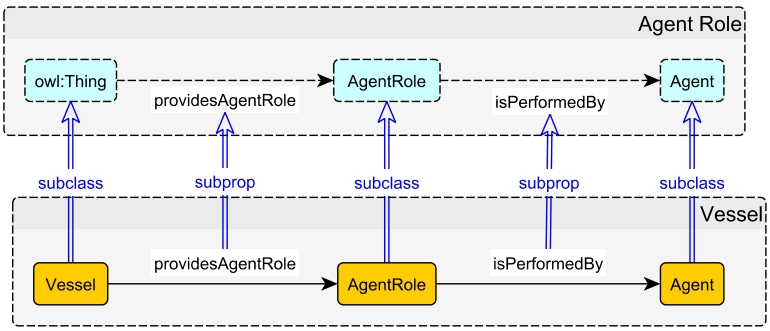


Figure A.61: The Vessel pattern alignment with Agent Role pattern

$$\text{ecglvs:isPerformedBy} \sqsubseteq \text{ecglar:isPerformedBy} \tag{A.372}$$

Alignment with Information Object pattern

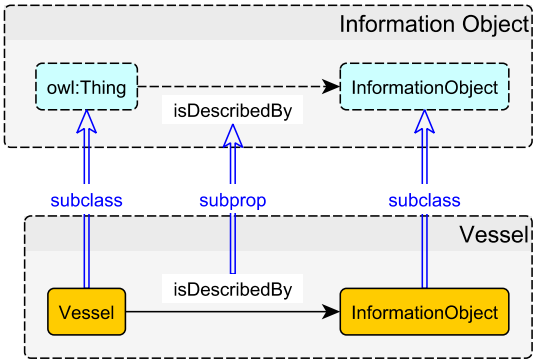


Figure A.62: The Vessel pattern alignment with Information Object pattern

```
Prefix: ecglio: <http://schema.geolink.org/dev/informationobject#>
Prefix: ecglvs: <http://schema.geolink.org/dev/vessel#>
Ontology: <http://schema.geolink.org/dev/vessel-to-informationobject>
ObjectProperty: ecglvs:isDescribedBy, ecglio:isDescribedBy
Class: ecglvs:InformationObject, ecglio:InformationObject
```

$$\text{ecglvs:InformationObject} \sqsubseteq \text{ecglio:InformationObject} \tag{A.373}$$

$$\text{ecglvs:isDescribedBy} \sqsubseteq \text{ecglio:isDescribedBy} \tag{A.374}$$

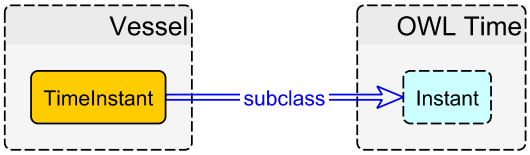


Figure A.63: The Vessel pattern alignment with OWL Time ontology

Alignment with OWL Time ontology

```
Prefix: time: <http://www.w3.org/2006/time#>
Prefix: ecglvs: <http://schema.geolink.org/dev/vessel#>
Ontology: <http://schema.geolink.org/dev/vessel-to-owltime>
Class: ecglvs:TimeEntity, time:TemporalEntity
```

$$\mathsf{ecglvs{:}TimeEntity} \sqsubseteq \mathsf{time{:}TemporalEntity} \tag{A.375}$$

Alignment with Platform pattern

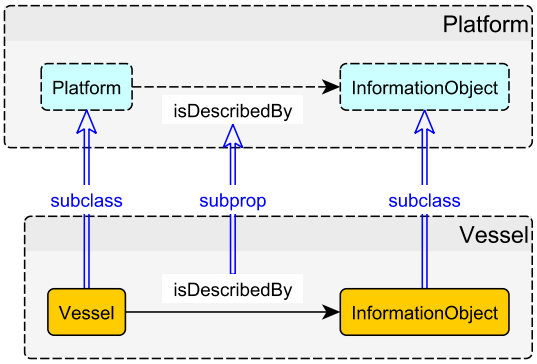


Figure A.64: The Vessel pattern alignment with Platform pattern

```
Prefix: ecglpf: <http://schema.geolink.org/dev/platform#>
Prefix: ecglvs: <http://schema.geolink.org/dev/vessel#>
Ontology: <http://schema.geolink.org/dev/vessel-to-informationobject>
ObjectProperty: ecglvs:isDescribedBy, ecglpf:isDescribedBy
Class: ecglvs:Vessel, ecglvs:InformationObject, ecglpf:Platform,
       ecglpf:InformationObject
```

$$\text{ecglvs:Vessel} \sqsubseteq \text{ecglpf:Platform} \tag{A.376}$$

$$\text{ecglvs:InformationObject} \sqsubseteq \text{ecglpf:InformationObject} \tag{A.377}$$

$$\text{ecglvs:isDescribedBy} \sqsubseteq \text{ecglpf:isDescribedBy} \tag{A.378}$$

Bibliography

[1] Serge Abiteboul, Richard Hull, and Victor Victor. *Foundations of Databases*. Addison-Wesley, 1995. URL: http://webdam.inria.fr/Alice/.

[2] Serge Abiteboul, Ioana Manolescu, Philippe Rigaux, Marie-Christine. Rousset, and Pierre Senellart. *Web Data Management*. Cambridge University Press, 2011. URL: https://books.google.com/books?id=-yKCPHemQ0sC.

[3] Marcelo Arenas, Alexandre Bertails, Eric Prud'hommeaux, and Juan Sequeda, eds. *A Direct Mapping of Relational Data to RDF*. Available at http://www.w3.org/TR/rdb-direct-mapping/. W3C Recommendation, 27 September 2012.

[4] Yigal Arens, Chun-Nan Hsu, and Craig A Knoblock. "Query processing in the SIMS information mediator". In: *Advanced Planning Technology* 32 (1996), pp. 78–93.

[5] Sören Auer, Sebastian Dietzold, Jens Lehmann, Sebastian Hellmann, and David Aumueller. "Triplify: light-weight linked data publication from relational databases". In: *Proceedings of the 18th International Conference on World Wide Web, WWW 2009, Madrid, Spain, April 20-24, 2009*. Ed. by Juan Quemada, Gonzalo León, Yoëlle S. Maarek, and Wolfgang Nejdl. ACM, 2009, pp. 621–630. URL: http://doi.acm.org/10.1145/1526709.1526793.

[6] Franz Baader, Diego Calvanese, Deborah L. McGuinness, Daniele. Nardi, and Peter F. Patel-Schneider. *The Description Logic Handbook: Theory, Implementation and Applications*. 2, illustrated, revised. Cambridge University Press, 2010. URL: https://books.google.com/books?id=c35WRQAACAAJ.

[7] Chahinez Bachtarzi and Fouzia Benchikha. "View-OD: a view model for ontology-based databases". In: *International Journal of Intelligent Information and Database Systems* 7.4 (2013), pp. 295–323.

[8] Ladjel Bellatreche, Nguyen Xuan Dung, Guy Pierra, and Dehainsala Hondjack. "Contribution of ontology-based data modeling to automatic integration of electronic catalogues within engineering databases". In: *Computers in Industry* 57.8 (2006), pp. 711–724.

[9] Tim Berners-Lee. *Linked Data - Design Issues*. July 27, 2006. URL: http://www.w3.org/DesignIssues/LinkedData.html.

[10] Tim Berners-Lee. *The World Wide Web: Past, Present and Future*. Online; Last accessed: December 7, 2015. Available at http://www.w3.org/People/Berners-Lee/1996/ppf.html. Aug. 1996.

[11] Tim Berners-Lee, Roy T. Fielding, and Larry Masinter. *Uniform Resource Identifier (URI): Generic Syntax*. STD 66. RFC Editor, Jan. 2005. URL: http://www.rfc-editor.org/rfc/rfc3986.txt.

[12] Tim Berners-Lee, James Hendler, and Ora Lassila. "The Semantic Web". In: *Scientific American* 284.5 (2001), pp. 28–37.

[13] Christian Bizer, Tom Heath, and Tim Berners-Lee. "Linked Data - The Story So Far". In: *Int. J. Semantic Web Inf. Syst.* 5.3 (2009), pp. 1–22. URL: http://dx.doi.org/10.4018/jswis.2009081901.

[14] Eva Blomqvist, Aldo Gangemi, and Valentina Presutti. "Experiments on pattern-based ontology design". In: *Proceedings of the fifth international conference on Knowledge capture*. ACM. 2009, pp. 41–48.

[15] Eva Blomqvist and Kurt Sandkuhl. "Patterns in Ontology Engineering: Classification of Ontology Patterns". In: *ICEIS 2005, Proceedings of the Seventh International Conference on Enterprise Information Systems, Miami, USA, May 25-28, 2005*. Ed. by Chin-Sheng Chen, Joaquim Filipe, Isabel Seruca, and José Cordeiro. 2005, pp. 413–416.

[16] Harold Boley, Gary Hallmark, Michael Kifer, Adrian Paschke, Axel Polleres, and Dave Reynolds, eds. *RIF Core Dialect (Second Edition)*. Available at http://www.w3.org/TR/rif-core/. W3C Recommendation, 5 February 2013.

[17] Harold Boley and Michael Kifer, eds. *RIF Basic Logic Dialect (Second Edition)*. Available at http://www.w3.org/TR/rif-bld/. W3C Recommendation, 5 February 2013.

[18] Harold Boley and Michael Kifer, eds. *RIF Framework for Logic Dialects (Second Edition)*. Available at http://www.w3.org/TR/rif-fld/. W3C Recommendation, 5 February 2013.

[19] Tim Bray, Dave Hollander, Andrew Layman, and Richard Tobin, eds. *Namespaces in XML 1.1 (Second Edition)*. Available at http://www.w3.org/TR/xml-names11/. W3C Recommendation, 16 August 2006.

[20] Tim Bray, Jean Paoli, C. M. Sperberg-McQueen, Eva Maler, and François Yergeau, eds. *Extensible Markup Language (XML) 1.0 (Fifth Edition)*. Available at http://www.w3.org/TR/xml/. W3C Recommendation, 26 November 2008.

[21] Tim Bray, Jean Paoli, C. M. Sperberg-McQueen, Eva Maler, François Yergeau, and John Cowan, eds. *Extensible Markup Language (XML) 1.1 (Second Edition)*. Available at http://www.w3.org/TR/xml11/. W3C Recommendation, 16 August 2006, edited in place 29 September 2006.

[22] Dan Brickley, Ramanathan V. Guha, and Brian McBride, eds. *RDF Schema 1.1*. Available at http://www.w3.org/TR/rdf-schema/. W3C Recommendation, 25 February 2014.

[23] Agustina Buccella, Alejandra Cechich, and Pablo Fillottrani. "Ontology-driven geographic information integration: A survey of current approaches". In: *Computers & Geosciences* 35.4 (2009), pp. 710–723.

[24] Andrea Calì, Diego Calvanese, Giuseppe De Giacomo, and Maurizio Lenzerini. "On the Expressive Power of Data Integration Systems". In: *Conceptual Modeling - ER 2002, 21st International Conference on Conceptual Modeling, Tampere, Finland, October 7-11, 2002, Proceedings*. Ed. by Stefano Spaccapietra, Salvatore T. March, and Yahiko Kambayashi. Vol. 2503. Lecture Notes in Computer Science. Springer, 2002, pp. 338–350. URL: http://dx.doi.org/10.1007/3-540-45816-6_33.

[25] Diego Calvanese, Giuseppe De Giacomo, Maurizio Lenzerini, and Riccardo Rosati. "Logical Foundations of Peer-To-Peer Data Integration". In: *Proceedings of the Twenty-third ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems, June 14-16, 2004, Paris, France*. Ed. by Catriel Beeri and Alin Deutsch. ACM, 2004, pp. 241–251. URL: http://doi.acm.org/10.1145/1055558.1055593.

[26] Gavin Carothers, ed. *RDF 1.1 N-Quads: A line-based syntax for an RDF datasets*. Available at http://www.w3.org/TR/n-quads/. W3C Recommendation, 25 February 2014.

[27] Gavin Carothers and Andy Seaborne, eds. *RDF 1.1 N-Triples: A line-based syntax for an RDF graph*. Available at http://www.w3.org/TR/n-triples/. W3C Recommendation, 25 February 2014.

[28] Gavin Carothers and Andy Seaborne, eds. *RDF 1.1 Trig: RDF Dataset Language*. Available at http://www.w3.org/TR/trig/. W3C Recommendation, 25 February 2014.

[29] Ana Maria de Carvalho Moura, Fabio Porto, Vânia Maria Ponte Vidal, Regis Pires Magalhães, Macedo Maia, Maira Poltosi, and Daniele C. Palazzi. "A semantic integration approach to publish and retrieve ecological data". In: *IJWIS* 11.1 (2015), pp. 87–119. URL: http://dx.doi.org/10.1108/IJWIS-08-2014-0028.

[30] Surajit Chaudhuri, Umeshwar Dayal, and Vivek Narasayya. "An overview of business intelligence technology". In: *Communications of the ACM* 54.8 (2011), pp. 88–98.

[31] Eun-Sun Cho, Yun-Sam Kim, Manpyo Hong, and We-Duke Cho. "Fine-Grained View-Based Access Control for RDF Cloaking". In: *Computer and Information Technology, 2009. CIT'09. Ninth IEEE International Conference on*. Vol. 1. IEEE. 2009, pp. 336–341.

[32] Joonmyun Cho, Soonhung Han, and Hyun Kim. "Meta-ontology for automated information integration of parts libraries". In: *Computer-Aided Design* 38.7 (2006), pp. 713–725.

[33] Nitishal Chungoora, A.-F. Cutting-Decelle, R.I.M. Young, G. Gunendran, Zahid Usman, Jennifer A Harding, and Keith Case. "Towards the ontology-based consolidation of production-centric standards". In: *International Journal of Production Research* 51.2 (2013), pp. 327–345.

[34] Peter Clark, John A. Thompson, and Bruce W. Porter. "Knowledge Patterns". In: *KR 2000, Principles of Knowledge Representation and Reasoning Proceedings of the Seventh International Conference, Breckenridge, Colorado, USA, April 11-15, 2000.* Ed. by Anthony G. Cohn, Fausto Giunchiglia, and Bart Selman. Morgan Kaufmann, 2000, pp. 591–600.

[35] Christine Collet, Michael N. Huhns, and Wei-Min Shen. "Resource Integration Using a Large Knowledge Base in Carnot". In: *IEEE Computer* 24.12 (1991), pp. 55–62. URL: http://doi.ieeecomputersociety.org/10.1109/2.116889.

[36] Michael Compton, Payam Barnaghi, Luis Bermudez, RaúL GarcíA-Castro, Oscar Corcho, Simon Cox, John Graybeal, Manfred Hauswirth, Cory Henson, Arthur Herzog, et al. "The SSN ontology of the W3C semantic sensor network incubator group". In: *Web Semantics: Science, Services and Agents on the World Wide Web* 17 (2012), pp. 25–32.

[37] World Wide Web Consortium. *Help and FAQ*. Online; Last accessed: December 7, 2015. Available at http://www.w3.org/Help/#invention. 2009.

[38] Joyce Cooper, Michael Noon, Chris Jones, Ezra Kahn, and Peter Arbuckle. "Big data in life cycle assessment". In: *Journal of Industrial Ecology* 17.6 (2013), pp. 796–799.

[39] Richard Cyganiak, David Wood, and Markus Lanthaler, eds. *RDF 1.1 Concepts and Abstract Syntax*. Available at http://www.w3.org/TR/rdf11-concepts/. W3C Recommendation, 25 February 2014.

[40] Souripriya Das, Seema Sundara, and Richard Cyganiak, eds. *R2RML: RDB to RDF Mapping Language*. Available at http://www.w3.org/TR/r2rml/. W3C Recommendation, 27 September 2012.

[41] Christian de Sainte Marie, Gary Hallmark, and Adrian Paschke, eds. *RIF Production Rule Dialect (Second Edition)*. Available at http://www.w3.org/TR/rif-prd/. W3C Recommendation, 5 February 2013.

[42] A.H. Doan, A. Halevy, and Z. Ives. *Principles of Data Integration*. Elsevier Science, 2012. URL: https://books.google.com/books?id=s2YCKGrO10YC.

[43] M. Duerst and M. Suignard. *Internationalized Resource Identifiers (IRIs)*. RFC 3987. RFC Editor, Jan. 2005. URL: http://www.rfc-editor.org/rfc/rfc3987.txt.

[44] Ronald Fagin, Phokion G. Kolaitis, Renée J. Miller, and Lucian Popa. "Data exchange: semantics and query answering". In: *Theoretical Computer Science* 336.1 (2005), pp. 89–124. URL: http://dx.doi.org/10.1016/j.tcs.2004.10.033.

[45] R. Fielding and J. Reschke. *Hypertext Transfer Protocol (HTTP/1.1): Message Syntax and Routing*. RFC 7230. RFC Editor, June 2014. URL: http://www.rfc-editor.org/rfc/rfc7230.txt.

[46] Daniel Fitzpatrick, François Coallier, and Sylvie Ratté. "A Holistic Approach for the Architecture and Design of an Ontology-Based Data Integration Capability in Product Master Data Management". English. In: *Product Lifecycle Management. Towards Knowledge-Rich Enterprises*. Ed. by Louis Rivest, Abdelaziz Bouras, and Borhen Louhichi. Vol. 388. IFIP Advances in Information and Communication Technology. Springer Berlin Heidelberg, 2012, pp. 559–568. URL: http://dx.doi.org/10.1007/978-3-642-35758-9_50.

[47] Michel Gagnon. "Ontology-based integration of data sources". In: *Information Fusion, 2007 10th International Conference on*. IEEE. 2007, pp. 1–8.

[48] Fabien Gandon and Guus Schreiber, eds. *RDF 1.1 XML Syntax*. Available at http://www.w3.org/TR/rdf-syntax-grammar/. W3C Recommendation, 25 February 2014.

[49] Aldo Gangemi. "Ontology Design Patterns for Semantic Web Content". In: *The Semantic Web - ISWC 2005, 4th International Semantic Web Conference, ISWC 2005, Galway, Ireland, November 6-10, 2005, Proceedings*. Ed. by Yolanda Gil, Enrico Motta, V. Richard Benjamins, and Mark A. Musen. Vol. 3729. Lecture Notes in Computer Science. Springer, 2005, pp. 262–276. URL: http://dx.doi.org/10.1007/11574620_21.

[50] Martin Giese, Ahmet Soylu, Guillermo Vega-Gorgojo, Arild Waaler, Peter Haase, Ernesto Jiménez-Ruiz, Davide Lanti, Martín Rezk, Guohui Xiao, Özgür L. Özçep, and Riccardo Rosati. "Optique: Zooming in on Big Data". In: *IEEE Computer* 48.3 (2015), pp. 60–67. URL: http://dx.doi.org/10.1109/MC.2015.82.

[51] Birte Glimm, Aidan Hogan, Markus Krötzsch, and Axel Polleres. "OWL: Yet to arrive on the Web of Data?" In: *WWW2012 Workshop on Linked Data on the Web, Lyon, France, 16 April, 2012*. Ed. by Christian Bizer, Tom Heath, Tim Berners-Lee, and Michael Hausenblas. Vol. 937. CEUR Workshop Proceedings. CEUR-WS.org, 2012. URL: http://ceur-ws.org/Vol-937/ldow2012-paper-16.pdf.

[52] Cheng Hian Goh. "Representing and Reasoning about Semantic Conflicts in Heterogeneous Information Sources". PhD thesis. Massachusetts Institute of Technology, 1997.

[53] David Gomez-Cabrero, Imad Abugessaisa, Dieter Maier, Andrew Teschendorff, Matthias Merkenschlager, Andreas Gisel, Esteban Ballestar, Erik Bongcam-Rudloff, Ana Conesa, and Jesper Tegnér. "Data integration in the era of omics: current and future challenges". In: *BMC systems biology* 8.Suppl 2 (2014), p. I1.

[54] Thomas R. Gruber. "A translation approach to portable ontology specifications". In: *Knowledge Acquisition* 5.2 (1993), pp. 199–220. URL: http://www.sciencedirect.com/science/article/pii/S1042814383710083.

[55] Thomas R Gruber. "Toward principles for the design of ontologies used for knowledge sharing". In: *International journal of human-computer studies* 43.5 (1995), pp. 907–928.

[56] Peter Haase, Tobias Mathäß, and Michael Ziller. "An evaluation of approaches to federated query processing over linked data". In: *Proceedings the 6th Inter-*

national Conference on Semantic Systems, I-SEMANTICS 2010, Graz, Austria, September 1-3, 2010. Ed. by Adrian Paschke, Nicola Henze, and Tassilo Pellegrini. ACM International Conference Proceeding Series. ACM, 2010. URL: http://doi.acm.org/10.1145/1839707.1839713.

[57] Peter Haase, Michael Schmidt, and Andreas Schwarte. "The Information Workbench as a Self-Service Platform for Linked Data Applications". In: *Proceedings of the Second International Workshop on Consuming Linked Data (COLD2011), Bonn, Germany, October 23, 2011*. Ed. by Olaf Hartig, Andreas Harth, and Juan Sequeda. Vol. 782. CEUR Workshop Proceedings. CEUR-WS.org, 2011. URL: http://ceur-ws.org/Vol-782/HaaseEtAl_COLD2011.pdf.

[58] Willem Robert van Hage, Véronique Malaisé, Roxane Segers, Laura Hollink, and Guus Schreiber. "Design and use of the Simple Event Model (SEM)". In: *J. Web Sem.* 9.2 (2011), pp. 128–136. URL: http://dx.doi.org/10.1016/j.websem.2011.03.003.

[59] Alon Y. Halevy, Zachary G. Ives, Dan Suciu, and Igor Tatarinov. "Schema Mediation in Peer Data Management Systems". In: *Proceedings of the 19th International Conference on Data Engineering, March 5-8, 2003, Bangalore, India*. Ed. by Umeshwar Dayal, Krithi Ramamritham, and T. M. Vijayaraman. IEEE Computer Society, 2003, pp. 505–516. URL: http://dx.doi.org/10.1109/ICDE.2003.1260817.

[60] Anthony Halog and Yosef Manik. "Advancing integrated systems modelling framework for life cycle sustainability assessment". In: *Sustainability* 3.2 (2011), pp. 469–499.

[61] Patrick J. Hayes and Peter F. Patel-Schneider, eds. *RDF 1.1 Semantics*. Available at http://www.w3.org/TR/rdf11-mt/. W3C Recommendation, 25 February 2014.

[62] Wu He and Li Da Xu. "Integration of Distributed Enterprise Applications: A Survey". In: *Industrial Informatics, IEEE Transactions on* 10.1 (Feb. 2014), pp. 35–42.

[63] Ivan Herman, Ben Adida, Manu Sporny, and Mark Birbeck, eds. *RDFa 1.1 Primer - Third Edition: Rich Structured Data Markup for Web Documents*. Available at http://www.w3.org/TR/rdfa-primer/. W3C Working Group Note, 17 March 2015.

[64] Pascal Hitzler and Frank van Harmelen. "A reasonable Semantic Web". In: *Semantic Web* 1.1 (2010), pp. 39–44.

[65] Jerry R. Hobbs and Feng Pan, eds. *Time Ontology in OWL*. Available at http://www.w3.org/TR/owl-time/. W3C Working Draft, 27 September 2006.

[66] Robert Hoehndorf, Frank Loebe, Janet Kelso, and Heinrich Herre. "Representing default knowledge in biomedical ontologies: Application to the integration of anatomy and phenotype ontologies". In: *BMC bioinformatics* 8.1 (2007), p. 377.

[67] Rinke J. Hoekstra. "Ontology Representation: Design Patterns and Ontologies that Make Sense". PhD thesis. University of Amsterdam, Sept. 2009.

[68] Yingjie Hu, Krzysztof Janowicz, David Carral, Simon Scheider, Werner Kuhn, Gary Berg-Cross, Pascal Hitzler, Mike Dean, and Dave Kolas. "A Geo-ontology Design Pattern for Semantic Trajectories". In: *Spatial Information Theory - 11th International Conference, COSIT 2013, Scarborough, UK, September 2-6, 2013. Proceedings*. 2013, pp. 438–456. URL: http://dx.doi.org/10.1007/978-3-319-01790-7_24.

[69] Edward Hung, Yu Deng, and Venkatramanan S Subrahmanian. "RDF aggregate queries and views". In: *Data Engineering, 2005. ICDE 2005. Proceedings. 21st International Conference on*. IEEE. 2005, pp. 717–728.

[70] ISO/IEC 10646:2014. *Information technology – Universal Coded Character Set (UCS)*. Standard. Geneva, CH: International Organization for Standardization, Aug. 2014.

[71] Saïd Izza. "Integration of industrial information systems: from syntactic to semantic integration approaches". In: *Enterprise Information Systems* 3.1 (2009), pp. 1–57.

[72] Prateek Jain, Pascal Hitzler, Peter Z Yeh, Kunal Verma, and Amit P Sheth. "Linked Data is Merely More Data". In: *AAAI Spring Symposium: linked data meets artificial intelligence*. Vol. 11. 2010.

[73] Yannis Kalfoglou and Marco Schorlemmer. "Ontology mapping: the state of the art". In: *The knowledge engineering review* 18.01 (2003), pp. 1–31.

[74] Michael Kifer and Harold Boley, eds. *RIF Overview (Second Edition)*. Available at http://www.w3.org/TR/rif-overview/. W3C Working Group Note, 5 February 2013.

[75] Gang-Hoon Kim, Silvana Trimi, and Ji-Hyong Chung. "Big-data applications in the government sector". In: *Communications of the ACM* 57.3 (2014), pp. 78–85.

[76] Graham Klyne and Jeremy J. Carroll, eds. *Resource Description Framework (RDF): Concepts and Abstract Syntax*. Available at http://www.w3.org/TR/2004/REC-rdf-concepts-20040210/. W3C Recommendation, 25 February 2004.

[77] Holger Knublauch and Arthur Ryman, eds. *Shapes Constraint Language (SHACL)*. Available at http://www.w3.org/TR/shacl/. W3C First Public Working Draft, 8October 2015.

[78] Phokion G. Kolaitis. "Schema mappings, data exchange, and metadata management". In: *Proceedings of the Twenty-fourth ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems, June 13-15, 2005, Baltimore, Maryland, USA*. Ed. by Chen Li. ACM, 2005, pp. 61–75. URL: http://doi.acm.org/10.1145/1065167.1065176.

[79] Adila Alfa Krisnadhi, Frederick Maier, and Pascal Hitzler. "OWL and Rules". In: *Reasoning Web. Semantic Technologies for the Web of Data. 7th International Summer School 2011, Galway, Ireland, August 23-27, 2011*. Ed. by A. Polleres, C. d'Amato, M. Arenas, S. Handschuh, P. Kroner, S. Ossowski, and P.F. Patel-Schneider. Vol. 6848. Lecture Notes in Computer Science. Springer, Heidelberg, 2011, pp. 382–415.

[80] Adila Krisnadhi, Robert Arko, Suzanne Carbotte, Cynthia Chandler, Michelle Cheatham, Tim Finin, Pascal Hitzler, Krzysztof Janowicz, Thomas Narock, Lisa Raymond, Adam Shepherd, and Peter Wiebe. "Ontology Pattern Modeling for Cross-Repository Data Integration in the Ocean Sciences: The Oceanographic Cruise Example". In: *The Semantic Web in Earth and Space Science: Current Status and Future Directions*. Ed. by Thomas Narock and Peter Fox. Vol. 20. Studies on the Semantic Web. IOS Press, 2015. Chap. 11, pp. 185–203.

[81] Adila Krisnadhi, Pascal Hitzler, and Krzysztof Janowicz. "On the Capabilities and Limitations of OWL Regarding Typecasting and Ontology Design Pattern Views". In: *Proceedings of the 12th International Workshop on OWL: Experiences and Directions (OWLED 2015) co-located with 14th International Semantic Web Conference on (ISWC 2015), Bethlehem, PA, USA, October 9-10, 2015*. To appear. 2015.

[82] Adila Krisnadhi, Yingjie Hu, Robert Arko, Suzanne Carbotte, Cynthia Chandler, Michelle Cheatham, Douglas Fils, Timothy Finin, Pascal Hitzler, Krzysztof Janowicz, Matthew Jones, Audrey Mickle, Thomas Narock, Margaret O'Brien, Lisa Raymond, Mark Schildhauer, Adam Shepherd, and Peter Wiebe. *GeoLink Core Ontology Design Patterns. Superseding OceanLink Technical Report 2014.2*. GeoLink Technical Report 2014.12. The GeoLink Project,

May 2015. 86 pp. URL: http://schema.geolink.org/docs/0.1/main-pattern-collections.pdf (visited on 07/30/2015).

[83] Adila Krisnadhi, Yingjie Hu, Krzysztof Janowicz, Pascal Hitzler, Robert A. Arko, Suzanne Carbotte, Cynthia Chandler, Michelle Cheatham, Douglas Fils, Timothy W. Finin, Peng Ji, Matthew B. Jones, Nazifa Karima, Kerstin Lehnert, Audrey Mickle, Thomas W. Narock, Margaret O'Brien, Lisa Raymond, Adam Shepherd, Mark Schildhauer, and Peter Wiebe. "The GeoLink Modular Oceanography Ontology". In: *The Semantic Web - ISWC 2015 - 14th International Semantic Web Conference, Bethlehem, PA, USA, October 11-15, 2015, Proceedings, Part II*. Ed. by Marcelo Arenas, Óscar Corcho, Elena Simperl, Markus Strohmaier, Mathieu d'Aquin, Kavitha Srinivas, Paul T. Groth, Michel Dumontier, Jeff Heflin, Krishnaprasad Thirunarayan, and Steffen Staab. Vol. 9367. Lecture Notes in Computer Science. Springer, 2015, pp. 301–309. URL: http://dx.doi.org/10.1007/978-3-319-25010-6_19.

[84] Adila Krisnadhi, Yingjie Hu, Krzysztof Janowicz, Pascal Hitzler, Robert Arko, Suzanne Carbotte, Cynthia Chandler, Michelle Cheatham, Douglas Fils, Timothy Finin, Peng Ji, Matthew Jones, Nazifa Karima, Kerstin Lehnert, Audrey Mickle, Thomas Narock, Margaret O'Brien, Lisa Raymond, Adam Shepherd, Mark Schildhauer, and Peter Wiebe. "The GeoLink Framework for Pattern-based Linked Data Integration". In: *Proceedings of the ISWC 2015 Posters and Demonstrations Track*. To appear. 2015.

[85] Adila Krisnadhi, Víctor Rodríguez-Doncel, Pascal Hitzler, Michelle Cheatham, Nazifa Karima, Reihaneh Amini, and Ashley Coleman. "An Ontology Design Pattern for Chess Games". In: *Proceedings of the 6th Workshop on Ontology and Semantic Web Patterns (WOP 2015) co-located with the 14th International Semantic Web Conference (ISWC 2015), Bethlehem, Pensylvania, USA, October 11, 2015*. Ed. by Eva Blomqvist, Pascal Hitzler, Adila Krisnadhi, Tom Narock, and Monika Solanki. Vol. 1461. CEUR Workshop Proceedings. CEUR-WS.org, 2015. URL: http://ceur-ws.org/Vol-1461/WOP2015_pattern_abstract_2.pdf.

[86] Fenareti Lampathaki, Spiros Mouzakitis, George Gionis, Yannis Charalabidis, and Dimitris Askounis. "Business to business interoperability: A current review of XML data integration standards". In: *Computer Standards & Interfaces* 31.6 (2009), pp. 1045–1055.

[87] T. A. Landers and Ronni Rosenberg. "An Overview of MULTIBASE". In: *Symposium on Distributed Data Bases*. 1982, pp. 153–184.

[88] Ora Lassila and Ralph R. Swick, eds. *Resource Description Framework (RDF) Model and Syntax Specification*. Available at http://www.w3.org/TR/1999/REC-rdf-syntax-19990222/. W3C Recommendation, 22 February 1999.

[89] Maurizio Lenzerini. "Data Integration: A Theoretical Perspective". In: *Proceedings of the Twenty-first ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems, June 3-5, Madison, Wisconsin, USA*. Ed. by Lucian Popa, Serge Abiteboul, and Phokion G. Kolaitis. ACM, 2002, pp. 233–246. URL: http://doi.acm.org/10.1145/543613.543644.

[90] Kevin M Livingston, Michael Bada, William A Baumgartner, and Lawrence E Hunter. "KaBOB: ontology-based semantic integration of biomedical databases". In: *BMC bioinformatics* 16.1 (2015), p. 126.

[91] Deborah L. McGuinness and Frank van Harmelen, eds. *OWL Web Ontology Language Overview*. Available at http://www.w3.org/TR/owl-features/. W3C Recommendation, 10 February 2004.

[92] Eduardo Mena, Arantza Illarramendi, Vipul Kashyap, and Amit P Sheth. "OBSERVER: An approach for query processing in global information systems based on interoperation across pre-existing ontologies". In: *Distributed and parallel Databases* 8.2 (2000), pp. 223–271.

[93] Hua Min, Frank J Manion, Elizabeth Goralczyk, Yu-Ning Wong, Eric Ross, and J Robert Beck. "Integration of prostate cancer clinical data using an ontology". In: *Journal of biomedical informatics* 42.6 (2009), pp. 1035–1045.

[94] Thomas Moser and Stefan Biffl. "Semantic integration of software and systems engineering environments". In: *Systems, Man, and Cybernetics, Part C: Applications and Reviews, IEEE Transactions on* 42.1 (2012), pp. 38–50.

[95] Boris Motik, Peter F. Patel-Schneider, and Bernardo Cuenca Grau, eds. *OWL 2 Web Ontology Language: Direct Semantics (Second Edition)*. Available at http://www.w3.org/TR/owl2-direct-semantics/. W3C Recommendation, 11 December 2012.

[96] Boris Motik, Peter F. Patel-Schneider, and Bijan Parsia, eds. *OWL 2 Web Ontology Language: Structural Specification and Functional-Style Syntax (Second Edition)*. Available at http://www.w3.org/TR/owl2-syntax/. W3C Recommendation, 11 December 2012.

[97] Robert Neches, Richard Fikes, Timothy W. Finin, Thomas R. Gruber, Ramesh S. Patil, Ted E. Senator, and William R. Swartout. "Enabling Technology for

Knowledge Sharing". In: *AI Magazine* 12.3 (1991), pp. 36–56. URL: http://www.aaai.org/ojs/index.php/aimagazine/article/view/902.

[98] Khai Nguyen, Ryutaro Ichise, and Bac Le. "SLINT: a schema-independent linked data interlinking system". In: *Proceedings of the 7th International Workshop on Ontology Matching, Boston, MA, USA, November 11, 2012*. Ed. by Pavel Shvaiko, Jérôme Euzenat, Anastasios Kementsietsidis, Ming Mao, Natasha Fridman Noy, and Heiner Stuckenschmidt. Vol. 946. CEUR Workshop Proceedings. CEUR-WS.org, 2012. URL: http://ceur-ws.org/Vol-946/om2012_Tpaper1.pdf.

[99] Natalya F Noy. "Semantic integration: a survey of ontology-based approaches". In: *ACM Sigmod Record* 33.4 (2004), pp. 65–70.

[100] Natalya Fridman Noy and Michel C. A. Klein. "Ontology Evolution: Not the Same as Schema Evolution". In: *Knowl. Inf. Syst.* 6.4 (2004), pp. 428–440. URL: http://www.springerlink.com/index/10.1007/s10115-003-0137-2.

[101] Daniel Oberle. *Semantic Management of Middleware*. Vol. 1. Semantic Web and Beyond: Computing for Human Experience. Springer, 2006.

[102] Daniel Oberle, Anupriya Ankolekar, Pascal Hitzler, Philipp Cimiano, Michael Sintek, Malte Kiesel, Babak Mougouie, Stephan Baumann, Shankar Vembu, Massimo Romanelli, Paul Buitelaar, Ralf Engel, Daniel Sonntag, Norbert Reithinger, Berenike Loos, Hans-Peter Zorn, Vanessa Micelli, Robert Porzel, Christian Schmidt, Moritz Weiten, Felix Burkhardt, and Jianshen Zhou. "DOLCE ergo SUMO: On foundational and domain models in the SmartWeb Integrated Ontology (SWIntO)". In: *Web Semantics: Science, Services and Agents on the World Wide Web* 5.3 (2007), pp. 156–174. URL: http://www.sciencedirect.com/science/article/pii/S1570826807000273.

[103] W3C OWL Working Group, ed. *OWL 2 Web Ontology Language: Document Overview (Second Edition)*. Available at http://www.w3.org/TR/owl2-overview/. W3C Recommendation, 11 December 2012.

[104] Hendrik Paasche, Detlef Eberle, Sonali Das, Antony Cooper, Pravesh Debba, Peter Dietrich, Nontembeko Dudeni-Thlone, Cornelia Gläßer, Andrzej Kijko, Andreas Knobloch, Angela Lausch, Uwe Meyer, Ansie Smit, Edgar Stettler, and Ulrike Werban. "Are Earth Sciences lagging behind in data integration methodologies?" English. In: *Environmental Earth Sciences* 71.4 (2014), pp. 1997–2003. URL: http://dx.doi.org/10.1007/s12665-013-2931-9.

[105] David Peterson, Shudi (Sandy) Gao, Ashok Malhotra, C. M. Sperberg-McQueen, Henry S. Thompson, and Paul V. Biron, eds. *W3C XML Schema*

Definition Language (XSD) 1.1 Part 2: Datatypes. Available at http://www.w3.org/TR/xmlschema11-2/. W3C Recommendation, 5 April 2012.

[106] Stephan Philippi and Jacob Köhler. "Using XML technology for the ontology-based semantic integration of life science databases". In: *Information Technology in Biomedicine, IEEE Transactions on* 8.2 (2004), pp. 154–160.

[107] Guy Pierra. "The PLIB ontology-based approach to data integration". In: *Building the Information Society*. Springer, 2004, pp. 13–18.

[108] Valentina Presutti, Enrico Daga, Aldo Gangemi, and Eva Blomqvist. "eXtreme Design with Content Ontology Design Patterns". In: *Proceedings of the Workshop on Ontology Patterns (WOP 2009), collocated with the 8th International Semantic Web Conference (ISWC 2009), Washington D.C., USA, 25 October, 2009.* 2009. URL: http://ceur-ws.org/Vol-516/pap21.pdf.

[109] Valentina Presutti and Aldo Gangemi. "Content Ontology Design Patterns as Practical Building Blocks for Web Ontologies". In: *Conceptual Modeling - ER 2008, 27th International Conference on Conceptual Modeling, Barcelona, Spain, October 20-24, 2008. Proceedings*. Ed. by Qing Li, Stefano Spaccapietra, Eric S. K. Yu, and Antoni Olivé. Vol. 5231. Lecture Notes in Computer Science. Springer, 2008, pp. 128–141. URL: http://dx.doi.org/10.1007/978-3-540-87877-3_11.

[110] Valentina Presutti, Aldo Gangemi, Stefano David, Guadalupe Aguado de Cea, Mari Carmen Suárez-Figueroa, Elena Montiel-Ponsoda, and María Poveda. *A Library of Ontology Design Patterns: reusable solutions for collaborative design of networked ontologies*. Tech. rep. Deliverable D2.5.1, NeOn Project, 2008.

[111] Eric Prud'hommeaux and Gavin Carothers, eds. *RDF 1.1 Turtle: Terse RDF Triple Language*. Available at http://www.w3.org/TR/turtle/. W3C Recommendation, 25 February 2014.

[112] Víctor Rodríguez-Doncel, Adila Alfa Krisnadhi, Pascal Hitzler, Michelle Cheatham, Nazifa Karima, and Reihaneh Amini. "Pattern-Based Linked Data Publication: The Linked Chess Dataset Case". In: *Proceedings of the 6th International Workshop on Consuming Linked Data co-located with 14th International Semantic Web Conference (ISWC 2105), Bethlehem, Pennsylvania, US, October 12th, 2015.* Ed. by Olaf Hartig, Juan Sequeda, and Aidan Hogan. Vol. 1426. CEUR Workshop Proceedings. CEUR-WS.org, 2015. URL: http://ceur-ws.org/Vol-1426/paper-05.pdf.

[113] Sebastian Rudolph, Markus Krötzsch, and Pascal Hitzler. "All Elephants are Bigger than All Mice". In: *Proceedings of the 21st International Workshop on*

Description Logics (DL2008), Dresden, Germany, May 13-16, 2008. Ed. by Franz Baader, Carsten Lutz, and Boris Motik. Vol. 353. CEUR Workshop Proceedings. CEUR-WS.org, 2008. URL: http://ceur-ws.org/Vol-353/RudolphKraetzschHitzler.pdf.

[114] Max Schmachtenberg, Christian Bizer, and Heiko Paulheim. "Adoption of the Linked Data Best Practices in Different Topical Domains". In: *The Semantic Web - ISWC 2014 - 13th International Semantic Web Conference, Riva del Garda, Italy, October 19-23, 2014. Proceedings, Part I*. Ed. by Peter Mika, Tania Tudorache, Abraham Bernstein, Chris Welty, Craig A. Knoblock, Denny Vrandecic, Paul T. Groth, Natasha F. Noy, Krzysztof Janowicz, and Carole A. Goble. Vol. 8796. Lecture Notes in Computer Science. Springer, 2014, pp. 245–260. URL: http://dx.doi.org/10.1007/978-3-319-11964-9_16.

[115] Andreas Schwarte, Peter Haase, Katja Hose, Ralf Schenkel, and Michael Schmidt. "FedX: Optimization Techniques for Federated Query Processing on Linked Data". In: *The Semantic Web - ISWC 2011 - 10th International Semantic Web Conference, Bonn, Germany, October 23-27, 2011, Proceedings, Part I*. Ed. by Lora Aroyo, Chris Welty, Harith Alani, Jamie Taylor, Abraham Bernstein, Lalana Kagal, Natasha Fridman Noy, and Eva Blomqvist. Vol. 7031. Lecture Notes in Computer Science. Springer, 2011, pp. 601–616. URL: http://dx.doi.org/10.1007/978-3-642-25073-6_38.

[116] Khouri Selma, Boukhari Ilyès, Bellatreche Ladjel, Sardet Eric, Jean Stéphane, and Baron Michael. "Ontology-based structured web data warehouses for sustainable interoperability: requirement modeling, design methodology and tool". In: *Computers in Industry* 63.8 (2012). Special Issue on Sustainable Interoperability: The Future of Internet Based Industrial Enterprises, pp. 799–812. URL: http://www.sciencedirect.com/science/article/pii/S0166361512001200.

[117] W3C SPARQL Working Group, ed. *SPARQL 1.1 Overview*. Available at http://www.w3.org/TR/sparql11-overview/. W3C Recommendation, 27 March 2013.

[118] Sebastian Speiser and Andreas Harth. "Integrating Linked Data and Services with Linked Data Services". In: *The Semantic Web: Research and Applications - 8th Extended Semantic Web Conference, ESWC 2011, Heraklion, Crete, Greece, May 29-June 2, 2011, Proceedings, Part I*. Ed. by Grigoris Antoniou, Marko Grobelnik, Elena Paslaru Bontas Simperl, Bijan Parsia, Dimitris Plexousakis, Pieter De Leenheer, and Jeff Z. Pan. Vol. 6643. Lecture Notes in Computer Science. Springer, 2011, pp. 170–184. URL: http://dx.doi.org/10.1007/978-3-642-21034-1_12.

[119] Manu Sporny, Gregg Kellogg, and Markus Lanthaler, eds. *JSON-LD 1.0: A JSON-based Serialization for Linked Data*. Available at http://www.w3.org/TR/json-ld/. W3C Recommendation, 16 January 2014.

[120] GeoLink Team. *GeoLink Meeting Notes - August 19-20, 2015 - Woods Hole*. Online with Access Restriction at https://docs.google.com/document/d/1SpDQcug34UshCenHHxfz_UyE4PeHsP_NbUk8WhdoOcc/. Aug. 2015.

[121] Thanh Tran, Haofen Wang, and Peter Haase. "Hermes: Data Web search on a pay-as-you-go integration infrastructure". In: *J. Web Sem.* 7.3 (2009), pp. 189–203. URL: http://dx.doi.org/10.1016/j.websem.2009.07.001.

[122] U.S. Department of Transportation - Federal Highway Administration. *Data Integration Primer - Challenges to Data Integration*. Apr. 29, 2015. URL: http://www.fhwa.dot.gov/asset/dataintegration/if10019/dip06.cfm.

[123] Harry T Uitermark, Peter JM van Oosterom, Nicolaas JI Mars, and Martien Molenaar. "Ontology-based geographic data set integration". In: *Spatio-temporal database management*. Springer. 1999, pp. 60–78.

[124] Jeffrey D. Ullman. "Information integration using logical views". In: *Theoretical Computer Science* 239.2 (2000), pp. 189–210. URL: http://dx.doi.org/10.1016/S0304-3975(99)00219-4.

[125] Michael Uschold and Michael Gruninger. "Ontologies and semantics for seamless connectivity". In: *ACM SIGMod Record* 33.4 (2004), pp. 58–64.

[126] Charles Vardeman, Adila Krisnadhi, Michelle Cheatham, Krzysztof Janowicz, Holly Ferguson, Pascal Hitzler, and Aimee Buccellato. "An Ontology Design Pattern and Its Use Case for Modeling Material Transformation". In: *Semantic Web* (2015). Accepted with minor revision.

[127] Charles Vardeman, Adila Krisnadhi, Michelle Cheatham, Krzysztof Janowicz, Holly Ferguson, Pascal Hitzler, Aimee Buccellato, Krishnaprasad Thirunarayan, Gary Berg-Cross, and Torsten Hahmann. "An Ontology Design Pattern for Material Transformation". In: *Proceedings of the 5th Workshop on Ontology and Semantic Web Patterns (WOP2014) co-located with the 13th International Semantic Web Conference (ISWC 2014), Riva del Garda, Italy, October 19, 2014*. Ed. by Victor de Boer, Aldo Gangemi, Krzysztof Janowicz, and Agnieszka Lawrynowicz. Vol. 1302. CEUR Workshop Proceedings. CEUR-WS.org, 2014, pp. 73–77.

[128] Stijn Verstichel, Femke Ongenae, Leanneke Loeve, Frederik Vermeulen, Pieter Dings, Bart Dhoedt, Tom Dhaene, and Filip De Turck. "Efficient data in-

tegration in the railway domain through an ontology-based methodology". In: *Transportation Research Part C: Emerging Technologies* 19.4 (2011), pp. 617–643. URL: http://www.sciencedirect.com/science/article/pii/S0968090X10001609.

[129] Vânia Maria P. Vidal, João C. Pinheiro, Eveline R. Sacramento, José Antônio Fernandes de Macêdo, and Bernadette Farias Lóscio. "An Ontology-Based Framework for Heterogeneous Data Sources Integration". In: *RITA* 16.2 (2009), pp. 61–64. URL: http://www.seer.ufrgs.br/index.php/rita/article/view/rita_v16_n2_p61.

[130] Ubbo Visser, Heiner Stuckenschmidt, Holger Wache, and Thomas Vögele. "Enabling technologies for interoperability". In: *Workshop on the 14th International Symposium of Computer Science for Environmental Protection*. Bonn, Germany, pp. 35–46.

[131] Raphael Volz, Daniel Oberle, and Rudi Studer. "Towards views in the semantic web". In: *2nd International Workshop on Databases, Documents and Information Fusion (DBFUSION02)*. 2002.

[132] Holger Wache, Th Scholz, Helge Stieghahn, and Birgitta König-Ries. "An integration method for the specification of rule-oriented mediators". In: *Database Applications in Non-Traditional Environments, 1999.(DANTE'99) Proceedings. 1999 International Symposium on*. IEEE. 1999, pp. 109–112.

[133] Holger Wache, Thomas Voegele, Ubbo Visser, Heiner Stuckenschmidt, Gerhard Schuster, Holger Neumann, and Sebastian Hübner. "Ontology-based integration of information-a survey of existing approaches". In: *Proceedings of IJCAI-01 Workshop: Ontologies and Information Sharing, Seattle, WA, 2001*. 2001, pp. 108–117.

[134] Kun Wang, Xiaoying Bai, Jing Li, and Cong Ding. "A service-based framework for pharmacogenomics data integration". In: *Enterprise Information Systems* 4.3 (2010), pp. 225–245.

[135] Gio Wiederhold. "Mediators in the architecture of future information systems". In: *Computer* 25.3 (1992), pp. 38–49.

[136] Andreas Wiesner, Jan Morbach, and Wolfgang Marquardt. "Information integration in chemical process engineering based on semantic technologies". In: *Computers & Chemical Engineering* 35.4 (2011), pp. 692–708.

[137] Patrick Ziegler and Klaus R. Dittrich. "Data Integration – Problems, Approaches, and Perspectives". English. In: *Conceptual Modelling in Informa-*

tion Systems Engineering. Ed. by John Krogstie, Andreas Lothe Opdahl, and Sjaak Brinkkemper. Springer Berlin Heidelberg, 2007, pp. 39–58. URL: http://dx.doi.org/10.1007/978-3-540-72677-7_3.